MyEID
PKI JavaCard Applet

Reference manual
Ver. 2.4.1

Contents

Version history

Date	Version	Description
28.1.2011	1.7.7	
17.12.2014	2.0.0	MyEID 4.x functionality added
29.1.2015	2.0.1	Corrections in references
5.2.2015	2.0.2	AES and DES, added response SW values
5.4.2015	2.0.3	Clarification on EC key lengths, added some error codes
9.4.2015	2.0.4	Added information about applet states
15.4.2015	2.0.5	Incorrect parameters in the data field / Wrong data (6A80)
4.5.2015	2.0.6	Init PIN flag default settings clarified. Moved General authenticate and clarified the command data structure.
15.5.2015	2.0.8	Specified how setting PIN and PUK retry counters work. Added AID in Activate Applet command's spec.
23.6.2015	2.0.9	Added PSO: ENCIPHER
10.3.2016	2.1.0	Added GET DATA: MyEID CARD CAPABILITIES
22.3.2016	2.1.1	Added symmetric keys to PUT DATA: LOAD KEY
12.4.2016	2.1.3	RESET RETRY COUNTER, specified C/R pins
28.4.2016	2.1.4	Corrections to EC related commands
7.9.2017	2.1.5	Corrected some references to e.g. tables
13.12.2017	2.1.6	Correction to P1 and P2 values in MSE:SET command when used before ECDH operation.
2.11.2018	2.2.0	Added key wrapping/unwrapping and session objects
25.3.2019	2.3.0	Added PUT DATA: Initialise PIV emulation
14.2.2020	2.4.0	Added description of 4096 bit RSA and APDU chaining. Added information about ATR.

Introduction

This document describes the MyEID PKI JavaCard applet, context and command interface.

The command set of the applet is based on ISO/IEC 7816-4 and ISO/IEC 7816-8 standards. The application related data is in files according to the PKCS #15 v1.0 specification. The applet implements the FINEID S1 v1.12 specification and it can be configured to include all or any subset of the features specified in FINEID S4-1 or S4-2 specifications. Starting from version 2.2.0 the applet supports both 1024 and 2048 bit RSA keys. From version 3.0.0 (MyEID3) the applet supports keys from 512 bit up to 2048 bit in increments of 64 bits and in version 4.5.5 maximum RSA key length has been increased to 4096 bits. Version 4 added support for Elliptic Curve Cryptography.

The applet is fully compatible with JavaCard 2.1.1 and GlobalPlatform 2.0.1 specifications. The applet can be set as the default applet on the card and in that case it will be automatically selected after the JavaCard has been powered up.

MyEID version 3 (MyEID3) uses the JavaCard 2.2.1 and GlobalPlatform 2.1.1 platform. It supports RSA keys from 512 up to 2048 bits in 64 bit increments. MyEID3 supports file sizes up to 32767 bytes and 14 different PIN-codes can be created and used. The number of RSA and ECC keys is only limited by the available memory and maximum numbers of files (see *PUT DATA: INITIALISE APPLET*).

MyEID versions 4.x (MyEID4) use JavaCard 2.2.2 or 3.0.x and GlobalPlatform 2.x.x platforms. It supports RSA and ECC asymmetric algorithms and DES and AES symmetric algorithms. MyEID4 is backwards compatible. As new features MyEID4 natively supports all requirements of the Microsoft Windows Minidriver Specification version 7, with e.g. administrator PIN, global unblocker, dynamic file size and deauthenticate command. A more detailed list of new features can be found in “New in MyEID 4” section and in the feature list table.

References

The most relevant specifications and standards are:

- ISO/IEC 7816-4
- ISO/IEC 7816-8
- ISO/IEC 7816-9
- ISO/IEC 7816-15
- JavaCard 2.1.1, MyEID3: 2.2.1
- GlobalPlatform 2.0.1 ' (Open Platform), MyEID3: GlobalPlatform 2.1.1
- FINEID S1 and S4 documentation
- PIV/CIV

1.1.1. Abbreviations

AC	Access Condition
AID	Application Identifier
APDU	Application Protocol Data Unit
ASN.1	Abstract Syntax Notation One
CRDO	Control Reference Data Object
CIV	Commercial Identity Verification
CLA	Class Byte
C/R	Challenge/Response
DF	Dedicated File
EF	Elementary File
FCI	File Control Information
FID	File Identifier
LC	Length of APDU command data
MF	Master File
MSE	Manage Security Environment
P1/P2	APDU command parameter 1 and 2
PIN	Personal Identification Number
PIV	Personal Identity Verification
PSO	Perform Security Operation
RFU	Reserved for Future Use
SE	Security Environment
SW	Status Word

New in MyEID 4.5

MyEID version 4.5 introduces key wrapping and unwrapping, and session objects.

Key wrapping and unwrapping

This feature enables transferring keys from and to the card in encrypted form. In key wrapping, card encrypts key material from a key file using another key, and outputs the resulting cryptogram. In key unwrapping, a cryptogram is input into the card. The card decrypts the cryptogram using a selected key and places the resulting plaintext into a key file.

The following usage scenarios are supported:

- Unwrap symmetric keys using RSA keys
- Unwrap symmetric keys using symmetric keys
- Wrap symmetric keys using symmetric keys.

Session objects

Session objects are temporary objects. In terms of PKCS#11 specification, they are objects with CKA_TOKEN attribute set to CK_FALSE. They are cleared from the card automatically next time the card is reset. This mechanism ensures that if the software that is using the card has no possibility to clear the session objects before the card is ejected, they are cleared when the card is powered up next time.

4096 bit RSA keys

MyEID 4.5 adds support to RSA keys up to 4096 bit length. Crypto-operations with 2048 bit and larger keys require data to be split into multiple APDUs. This is handled with APDU chaining. In command APDUs, the CLA byte is set to 10h when there is more data to follow. In response APDU, SW1/SW2 value 61xx indicates that there is more response data waiting, and the next block can be requested using the GET RESPONSE command. SW2 indicates the number of bytes remaining, 00 indicates a full block. The old method of splitting a 2048 bit cryptogram in PSO: Decipher operations remains available for backwards compatibility.

New in MyEID 4

MyEID version 4.0 introduces several new features including Elliptic Curve Cryptography (ECC) support, challenge/response PINs and many new options for creating files and PINs. Some of the features are implemented specially to make the applet fulfil Microsoft’s Smart Card Minidriver specification completely and make MyEID card a Windows Certified device.

ECC support

The applet supports ECDSA and ECDH algorithms with the following named prime curves, which are defined by NIST in document “RECOMMENDED ELLIPTIC CURVES FOR FEDERAL GOVERNMENT USE”: P-192, P-224, P-256, P-384 and P-521. Some of the MyEID4 platforms do not support 384 and 521 bit EC keys.

Challenge/response PINs

When authenticating a challenge/response PIN the card generates a random challenge. The user encrypts the challenge with a 3DES key that is also stored on the card, and sends the response to the card using VERIFY command APDU. The card decrypts the response and compares it with the original challenge.

Auto size files

Files created with the “auto size” flag set grow dynamically, when they are updated past their initial size.

Admin state

In previous applet versions each operation for each card object could be only accessed by one specified PIN. In some situations it is practical that an object can be accessed with different PINs. The admin state has been implemented to fulfil this requirement.

A PIN can be create with the administrator PIN flag set. When such PIN is verified, the admin state is enabled. Admin state gives special access to files and keys. If for example the normal user PIN is PIN #1 and PIN #3 is given admin flag, the administrator could be allowed to unblock PIN #1 or write to a file which has PIN#1 defined for writing in security attributes list. By default admin state does not allow special access to any file or PIN. To prevent unintentional access, the operations that are allowed in admin state must be defined separately for each card object.

PIV / CIV emulation

The MyEID4 card can emulate a PIV/CIV card by mapping the ISO 7816-15 (PKCS#15) structure to the PIV/CIV command interface. The PIV/CIV emulation can be switched on and off with a single command. The MyEID4 is not fully PIV/CIV compliant; the aim is to emulate the functionality to the extent of interoperability on command interface level, and to allow using MyEID cards in PIV compatible devices, in situations where native MyEID middleware cannot be installed.

All command parameters defined in the PIV specification are not be supported. Returned data and error codes do not comply with the specification in all situations.

With the PIV/CIV emulation enabled the card will be identified as a PIV/CIV card and with the relevant AID.

Cards cannot be personalised using the PIV/CIV interface, it is read-only in this sense. The cards must be personalised using the MyEID command interface, and keys and certificates are mapped to the PIV/CIV objects using FIDs, by executing a PUT DATA: INITIALISE PIV EMULATION command.

Feature list table

In the following table is shown in which version of the applet specific features have been introduced.

Feature	MyEID 3	MyEID 4	MyEID 4.5
Challenge/response PINs		*	*
Elliptic Curve Cryptography		*	*
Automatically growing files		*	*
Admin state		*	*
Global unblocker state		*	*
3DES encryption	(*) [1]	*	*
AES encryption	(*) [1]	*	*
Key wrapping			*
Generic Secret key EF’s			*
Session objects			*
4096 bit RSA key length			*

1) On MyEID 3 symmetric encryption is a special feature that has to be ordered separately.

Contents and File Structure

The applet follows the ISO 7816-15 standard (based on PKCS #15 specification) and implements all or any subset of files and contents of FINEID S4-1, S4-2, or ISO 7816-15 specifications.

The applet can contain the following objects:

- private RSA and ECC keys,
- public RSA and ECC keys,
- authentication objects,
- card holder certificates,
- trusted certificates, and
- any data objects

Applet states

To simplify card personalisation, MyEID applet includes a concept of Creation State.
An empty MyEID card is in Creation State, where access conditions are not effective. This way the whole card personalisation can be completed without verifying PINs required to access the files included in the process. After personalisation ACTIVATE APPLET command must be issued, which switches the applet to Operational State. The only way to switch the

applet back to Creation State is to reinitialise the applet, which empties the applet's contents (files, keys and PINs) completely.

MyEID4 introduces more states: Admin State and Global Unblocker state which allow changing and unblocking other PINs. These features are optional properties of PINs and they are explained in PUT DATA: INITIALISE PIN command's description.

Answer to Reset

MyEID cards up to version 3.3.3 return the following ATR:

```
3B F5 18 00 00 81 31 FE 45 4D 79 45 49 44 9A
```

Since version 4.0.0, most cards have been shipped with this ATR:

```
3B F5 96 00 00 81 31 FE 45 4D 79 45 49 44 14
```

In the new ATR, TA1 byte has been changed to enable faster communication with smart card readers. To avoid compatibility issues, it is recommended to identify MyEID cards with only the historical characters instead of the whole ATR string. Cards with the old ATR are still available upon request.

Command Interface

This chapter describes the commands of the applet with their parameters.

The commands are split into two categories:

- regular commands
- personalisation and management commands

The regular commands are compatible with the FINEID S1 v1.12 specifications that comply with the ISO/IEC 7816-4 and 7816-8 standards.

The personalisation and management commands comply with the ISO/IEC 7816-4 and 7816-9 standards.

3.1 Command lists

Regular commands:

Table 1 - Regular usage related commands of the MyEID applet.

Command	Standard	Functionality
GET DATA	ISO/IEC 7816-4	Retrieves applet version and other information
SELECT FILE	ISO/IEC 7816-4	Selects the applet or a file within the applet.
GET RESPONSE	ISO/IEC 7816-4	Reads card response in T=0 protocol mode.
READ BINARY	ISO/IEC 7816-4	Reads contents from a transparent (binary) file.

UPDATE BINARY	ISO/IEC 7816-4	Updates contents in a transparent (binary) file.
ERASE BINARY	ISO/IEC 7816-4	Erases contents in a transparent (binary) file.
VERIFY	ISO/IEC 7816-4	Verifies reference data presented by the user (PIN) with the reference data stored in the applet. The current verification status can be queried with this command.
CHANGE REFERENCE DATA	ISO/IEC 7816-8	Changes the current reference data (PIN).
RESET RETRY COUNTER	ISO/IEC 7816-8	Unblocks a blocked reference data (PIN).
MANAGE SECURITY ENVIRONMENT: **RESTORE**	ISO/IEC 7816-8	Restores a predefined or empty security environment.
MANAGE SECURITY ENVIRONMENT: **SET**	ISO/IEC 7816-8	Sets security environment parameters that will be used with subsequent PSO command(s).
PERFORM SECURITY OPERATION: **COMPUTE DIGITAL SIGNATURE**	ISO/IEC 7816-8	Computes a digital signature using a private key. The algorithm and the key are selected with the preceding MSE command.
PERFORM SECURITY OPERATION: **DECIPHER**	ISO/IEC 7816-8	Decrypts data with a private key. The algorithm and the key are selected with the preceding MSE command.
GET CHALLENGE	ISO/IEC 7816-4	Retrieves a challenge for authenticating a challenge/response PIN. Can be used to request random bytes from the card's RNG also for other purposes.
GENERAL AUTHENTICATE	ISO/IEC 7816-4	Calculates and returns a shared secret using the Elliptic Curve Diffie-Hellman (ECDH) key agreement protocol.
DEAUTHENTICATE	-	Deauthenticates selected access conditions without resetting the card.

Personalisation and management commands:

Table 2 – MyEID applet personalisation and management related commands.

Command	**Standard**	**Functionality**
PUT DATA: **LOAD AUID**	ISO/IEC 7816-4	Stores a unique identifier in the applet.

PUT DATA: **INITIALISE APPLET**	ISO/IEC 7816-4	Initialises the file system and pre-creates the permanent files (MF and DF 5015 (PKCS#15).
PUT DATA: **INITIALISE PIN**	ISO/IEC 7816-4	Initialises the requested PIN reference data and its unblocking reference data (PUK).
ACTIVATE APPLET	ISO/IEC 7816-9	Activates the applet. Sets files to Activated State.
CREATE FILE	ISO/IEC 7816-9	Creates a file in the applet's file system under the current DF.
DELETE FILE	ISO/IEC 7816-9	Deletes the current file.
GENERATE KEY PAIR	ISO/IEC 7816-8	Generates an RSA or ECC key pair in the current key EF. *
PUT DATA: **LOAD KEY PAIR**	ISO/IEC 7816-4	Loads an externally generated RSA or ECC key pair to a key EF. *
PUT DATA: **LOAD KEY**	ISO/IEC 7816-4	Loads a DES or AES key to key EF

* Applet version 2.2.0 onwards also supports 2048 bit keys. Earlier versions support only 1024 bit keys. MyEID3 supports key sizes from 512 to 2048 bit in 64 bit increments and MyEID 4.5 supports RSA keys up to 4096 bit key length.

3.2 Command response status bytes

Table 3 – General status bytes

SW1 / SW2	Functionality
9000h	Command successful
61xxh	Bytes remaining, xx = number of bytes
63Cxh	PIN verification trials left, x = trials left
6700h	Incorrect data length or wrong length
6981h	Incorrect file type
6982h	Security status not satisfied
6983h	PIN blocked or PIN not initialised or File invalid
6984h	Invalid data
6985h	Conditions not satisfied
6986h	Command not allowed
6A80h	Incorrect parameters in the data field / Wrong data
6A81h	Function not supported
6A82h	File or tag not found
6A83h	PIN not initialised
6A86h	Incorrect P1 and/or P2 parameters
6B00h	Offset exceeds file length
6Cxxh	Correct length, xx bytes
6D00h	Unsupported command instruction

Regular Commands

4.1 GET DATA

The GET DATA command retrieves information about the applet and its current state. The type of the returned information depends on parameter P2.

Table 4 - GET DATA command APDU

Bytes	Value
CLA	00h
INS	CAh
P1	01h
P2	Get Key File information (a key file must be selected): 00h: algorithm identifier, see Table 6. 01h: modulus (RSA keys only), see Table 7. 02h: public exponent (RSA keys only), see Table 8. 81h-85h: Elliptic curve parameters, see Table 9. 86h: ECC key's public point in uncompressed format Get other information: A0h: MyEID Applet information, see Table 10. A1h: current DF file list, see Table 12. A2h: same as A1h, but only includes EF's A3h: same as A1h, but only includes DF's A4h: same as A1h, but only includes RSA key EF's A5h: same as A1h, but only includes ECC key EF's A6h: same as A1h, but only includes symmetric key EF's A8h: Get full path of currently selected DF A9h: Get full path of currently selected EF AAh: MyEID card capabilities, see Table 11. * Get PIN information: BXh: X defines the PIN reference number. Get access condition states: ACh: see Table 15
LC	Empty
Data	Empty
LE	Empty or length of requested data

* P2 value AAh: MyEID card capabilities is available beginning from MyEID 4.0.0

Table 5 - GET DATA response APDU

Bytes	Value
Data	Requested data, see details in tables below

SW1 – SW2	Status Bytes

The GET DATA response command APDU data holds specific information about the applet and the current DF.

Table 6 – Algorithm Identifier

Bytes	Length	Value
Algorithm Identifier	2	9200h: RSA CRT
Modulus size	2	XXXXh: length of modulus in bits
Public exponent size	2	XXXXh: length of public exponent in bits

Table 7 – Modulus

Bytes	Length	Value
Modulus	N	N = length of modulus in bytes

Table 8 – Public Exponent

Bytes	Length	Value
Public Exponent	N	N = length of public exponent in bytes

Table 9 – Elliptic Curve Parameters

P2	Value
81h	Prime (a number denoted as p coded on z bytes)
82h	First coefficient (a number denoted as b coded on z bytes)
83h	Second coefficient (a number denoted as b coded on z bytes)
84h	Generator (a point denoted as PB on the curve, coded on 2z bytes)
85h	Order (a prime number denoted as q, order of the generator PB, coded on z bytes)

Table 10 – Applet Information

Bytes	Length	Value
Name	5	"MyEID"
Major version	1	XXh: major version number
Minor version	1	XXh: minor version number
Version revision	1	XXh: revision number
Unique identifier	10	Unique identifier, different from card to card

Change counter	2	XXXXh: Incremented after every successful command that changes the contents of the applet.

Table 11 – MyEID card capabilities

Bytes	Length	Value
Structure version	1	1: MyEID card capabilities – version 1. Version number can be increased in future to indicate that new information is added to the end of the structure
MyEID card capabilities	2	Specifies capabilities supported in this version of MyEID. bit flags, bytes in big endian order: bit 0 (0001h): RSA algorithm bit 1 (0002h): DES/3DES algorithm bit 2 (0004h): AES algorithm bit 3 (0008h): ECDSA and ECDH algorithms bit 4 (0010h): GridPIN bit 5 (0020h): PIV emulation bits 6-15: RFU
Maximum RSA key length	2	Maximum key size in bits
Maximum DES/3DES key length	2	Maximum key size in bits
Maximum AES key length	2	Maximum key size in bits
Maximum ECC key length	2	Maximum key size in bits

Table 12 – Current DF file list

Bytes	Length	Value
First FID	2	XXXXh: FID of the first file in current DF
Second FID	2	XXXXh: FID of the second file in current DF
…	…	…
Nth FID	2	XXXXh: FID of the Nth file in current DF

Table 13 – Current DF or EF path

Response data
The path is returned in reverse order, current file ID first. For example: 4B0150153F00

Table 14 – PIN information

Bytes	Length	Value
PIN tries remaining	1	Number of verification tries remaining
PUK tries remaining	1	Number of unblocking tries remaining
PIN tries	1	Number of verification tries initially or after successful VERIFY or RESET RETRY COUNTER command
PUK tries	1	Number of verification tries initially or after successful RESET RETRY COUNTER command.
Status byte	1	Status byte containing PIN flags, see Initialise PIN command
PIN type	1	00h = Normal PIN, 01h = Grid PIN, 02h = Challenge / response PIN
Grid size	1	Grid size for Grid PINs or 00h
Minimum length of PIN	1	Minimum length of PIN reference data
Minimum length of PUK	1	Minimum length of PUK reference data

Table 15 – Access condition states – two bytes of bit flags

Value	Meaning
0x8000	Admin state active
0x4000	Global unblocker state active
0x2000	PIN 14 verified
...	PINs 2-13
0x0001	PIN 1 verified

4.2 SELECT FILE

The SELECT FILE command selects the applet itself with the Application Identifier (AID) or a file within the applet. A file can be selected by a single File Identifier (FID) or by relative or absolute path. A path consists of a sequence of FID’s.

Table 16 - SELECT FILE command APDU

Byte	Value
CLA	00h
INS	A4h
P1	00h: select EF, DF or MF by single FID 04h: select applet by AID or a DF by its name 08h: select file by absolute path from MF 09h: select file by relative path current DF
P2	00h: FCI returned in response
LC	00h or length of data

Data	P1 = 00h:
	EF, DF or MF FID (or empty for MF)
	P1 = 04h:
	AID value (for applet or DF name)
	P1 = 08h:
	absolute path from MF without the FID for MF
	P1 = 09h:
	relative path from the current DF without the FID of current DF

LE	Empty

Table 17 - SELECT FILE response APDU

Byte	Value
Data	File Control Information (FCI), empty if protocol is T=0, or on error
SW1 – SW2	Status Bytes 0x6A82 – File not found

The SELECT FILE response command APDU data holds specific information about the selected file.

Table 18 - MF, DF, and Applet File Control Information data

Byte	Length	Value
FCI tag	1	6Fh
Length	1	17h – 2Fh
File Size tag	1	81h

Length	1	02h
File Size	2	XXXXh: available free space for additional files
File Description tag	1	82h
Length	1	01h
File Descriptor	1	38h: DF or MF
File Identifier tag	1	83h
Length	1	02h
File Identifier	2	XXXXh: FID
Security Attributes tag	1	86h
Length	1	03h
Security Attributes	3	XXXXXXh: See Table 21 for details of MF See Table 22 for details of DF
Proprietary Information tag	1	85h: File status byte
Length	1	02h
Proprietary Information	2	0000h: normal file 0002h: permanent file; DF cannot be deleted wit DELETE FILE command.
Life Cycle Status tag	1	8Ah
Length	1	01h
Life Cycle Status	1	X1h: creation state X7h: operational state – activated The high nibble is RFU
DF Name tag	1	84h: Optional tag
Length	1	01h – 10h
DF Name	1-16	

Table 19 - EF File Control Information data

Byte	Length	Value
FCI tag	1	6Fh
Length	1	17h
File Size tag	1	80h
Length	1	02h

File Size	2	Transparent EF’s: XXXXh: File size in bytes RSA key EF’s: 0400h to 1000h* for a private RSA key EF (=key size in bits, or modulus, of the key EF) MyEID3 accepts values from 0200h up to 0800h in 64 bit increments, MyEID 4.5 accepts values up to 1000h. ECC key EF’s: XXXXh: key length in bits (field size): 192, 224, 256, 384** or 521** DES key EF’s: 0040h: 64 bit DES key 0080h: 128 bit 3DES key (two key) 00C0h: 192 bit 3DES key (three key) AES key EF’s: 0080h: 128 bit key 0100h: 256 bit key
File Description tag	1	82h
Length	1	01h
File Descriptor	1	01h: transparent EF 11h: private RSA key EF 19h: DES key EF 22h: EC key EF 29h: AES key EF 41h: Generic secret key EF ***
File Identifier tag	1	83h
Length	1	02h
File Identifier	2	XXXXh: FID
Security Attributes tag	1	86h
Length	1	03h
Security Attributes	3	XXXXXXh: See Table 23 for details of transparent EF See Table 24 for details of private key EF
Proprietary Information tag	1	85h: File status byte
Length	1	02h

Proprietary Information	2	First status byte: Transparent EF's 00h: RFU Key EF's X0h: key not valid X1h: key valid X3h: key valid, key generated on card X = AC to be cleared after RSA operation. The remaining bits are RFU for EF Key, and for the other file types the whole byte is RFU. Second status byte: Bit flags: bit 0 (01h): Session object: The key EF is automatically removed during next reset. bit 3 (08h): Extractable: The key can be wrapped. 00h = key is not extractable and not a session object.
Life Cycle Status tag	1	8Ah
Length	1	01h
Life Cycle Status	1	X1h: creation state X7h: operational state – activated The high nibble is RFU

* 0800h is supported from MyEID applet version 2.2.0 onwards.
** Only supported on some of the MyEID platforms
*** a Generic Secret is a secret key which is not associated with an algorithm and cannot perform crypto-operations. I.e. it is a secret file that is treated like a key.

File Security Attributes

The operations that can be performed on a file, such as reading and updating, are controlled by File Security Attributes. Every file has six Security Attributes, coded in four bits each, and totalling three bytes. A given Security Attribute indicates which Access Condition must be fulfilled before the operation in question can be performed. The Access Conditions can be fulfilled by executing the VERIFY command with the correct data.

The exact meaning of the six File Security Attributes depends on the file type, as listed in tables 14 to 17. Each Security Attribute is coded in four bits as follows:

Table 20 - File Security Attribute values

Value	Meaning
0h	The operation is allowed at all times
1h…Eh	Indicates the reference number of the PIN that must be valid before the operation can be performed
Fh	The operation is forbidden

Table 21 - MF Security Attributes

Position	Meaning
X00000h	Create DF’s
0X0000h	Create EF’s
00X000h	Recreate MF (removes all existing files)
000X00h	RFU
0000X0h	RFU
00000Xh	RFU

Table 22 - DF Security Attributes

Position	Meaning
X00000h	Create DF’s
0X0000h	Create EF’s
00X000h	Delete File (this file)
000X00h	RFU
0000X0h	RFU
00000Xh	RFU

Table 23 - Transparent EF Security Attributes

Position	Meaning
X00000h	Read
0X0000h	Update / Erase
00X000h	Delete File (this file)
000X00h	RFU
0000X0h	RFU
00000Xh	RFU

Table 24 – RSA or ECC Key EF Security Attributes

Position	Meaning
X00000h	Use

0X0000h	Put Data
00X000h	Delete File (this file)
000X00h	Generate
0000X0h	RFU
00000Xh	RFU

4.3 GET RESPONSE

The GET RESPONSE command returns the response of the APDU when T=0 protocol is used. The response can only be retrieved once, and the GET RESPONSE command must be executed immediately after the command that generated the response.

This command retrieves the response data of the following commands:

- SELECT FILE,
- PSO: COMPUTE DIGITAL SIGNATURE, and
- PSO: DECIPHER

Table 25 - GET RESPONSE command APDU

Byte	Value
CLA	00h
INS	C0h
P1	00h
P2	00h
LC	Empty
Data	Empty
LE	Data length of the response to retrieve

Table 26 - GET RESPONSE response APDU

Byte	Value
Data	Previous APDU commands response data. Empty on error
SW1 – SW2	Status Bytes

4.4 READ BINARY

The READ BINARY command reads data from transparent EF’s. The file to read must be selected first with the SELECT FILE command.

Table 27 - READ BINARY command APDU

Byte	Value
CLA	00h
INS	B0h
P1	00-7Fh: MSB of the 15 bit offset to the first byte to read
P2	00-FFh: LSB of the 15 bit offset to the first byte to read
LC	Empty
Data	Empty
LE	Number of bytes to read.

Table 28 - READ BINARY response APDU

Byte	Value
Data	Bytes read, or empty on error
SW1 – SW2	Status Bytes

4.5 UPDATE BINARY

The UPDATE BINARY command updates data in transparent EF's. The file to update must be selected first with the SELECT FILE command.

Table 29 - UPDATE BINARY command APDU

Byte	Value
CLA	00h
INS	D6h
P1	00-7Fh: MSB of the 15 bit offset to the first byte to update
P2	00-FFh: LSB of the 15 bit offset to the first byte to update
LC	Length of data to update
Data	Data to be written
LE	Empty

Table 30 - UPDATE BINARY response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

4.6 ERASE BINARY

The ERASE BINARY command erases bytes starting at the requested offset until the end of the transparent EF. The file to erase must be selected first with the SELECT FILE command.

Table 31 - ERASE BINARY command APDU

Byte	Value
CLA	00h
INS	0Eh
P1	00-7Fh: MSB of the 15 bit offset to the first byte to erase
P2	00-FFh: LSB of the 15 bit offset to the first byte to erase
LC	Empty
Data	Empty
LE	Empty

Table 32 - ERASE BINARY response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

Table 33 - Response Status Byte values

SW1 – SW2	Description
0x9000	Verification successful and/or AC acquired
0x6985	PIN locked – *CONDITIONS NOT SATISFIED*
0x6983	Verification failed and no number of retries left PIN blocked.
0x63CX	Verification failed and X number of retries left.

4.7 VERIFY

The VERIFY command is used to authenticate the user. The verification data (PIN) is compared internally with the reference data stored in the applet. A successful verification sets the Access Condition (AC) in parameter P2, which enables the execution of commands that are restricted by that AC in the File Security Attributes of the file in question. This command can also be used just to inquire the status of the given Access Condition. This is achieved by executing the command without any reference data.

Verification can fail if the presented PIN is incorrect, or the PIN is blocked or locked.

Blocked PIN

Each time the PIN is verified with this command, a retry counter connected to this PIN is updated. If the verification was successful, the counter is reset to its original value of 5 (or optionally a value that has been configured when the PIN was created). If the verification fails, the counter will be decremented. If the counter reaches zero, the PIN will be blocked.

To unblock it, a Reset Retry Counter command must be executed with the correct PUK (PIN Unblocking Code) and a new PIN. This also resets the retry counter to the original value.

PIN locking

Besides blocking, it is also possible to configure the PIN in such a way that it can be locked. The locking can be set to occur in two situations: After the creation of the PIN, and after unblocking the PIN with the Reset Retry Counter command. A locked PIN can be unlocked by changing it with the Change Reference Data command. This optional feature can be used to enforce the changing of an initial PIN when user receives his card. This can be desirable if all the cards initially have a constant PIN. A locked PIN will cause the Verify command to fail with the error code SW1SW2 = 0x6985 (*CONDITIONS NOT SATISFIED*).

Table 34 - VERIFY command APDU

Byte	Value
CLA	00h
INS	20h
P1	00h
P2	PIN reference number 01h to 03h MyEID3: 01h to 0Eh
LC	00h: No data, just query the status of the PIN 08h: PIN reference data present in the data field 10h: Response for a challenge present in the data field
Data	Normal PINs: Empty or PIN reference data (verification data) padded to 8 bytes. Byte value 00h or FFh can be used for padding. C/R PINs: 16 bytes of response data.
LE	Empty

Table 35 - VERIFY response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

Table 36 - Response Status Byte values

SW1 – SW2	Description
0x9000	Verification successful and/or AC acquired
0x6985	PIN locked – *CONDITIONS NOT SATISFIED*
0x6983	Verification failed and no number of retries left PIN blocked.
0x63CX	Verification failed and X number of retries left.

4.8 CHANGE REFERENCE DATA

The CHANGE REFERENCE DATA command replaces the current internal reference data with a new value. The current reference data is first validated with the verification data. Successful validation also removes optional reference data locking.

Table 37 - CHANGE REFERENCE DATA command APDU

Byte	Value
CLA	00h
INS	24h
P1	00h: exchange reference data
P2	PIN reference number 01h to 03h MyEID3: 01h to 0Eh
LC	PIN change using verification reference data: 10h PIN change using admin state: 08h
Data	PIN change using verification reference data: Verification reference data padded to 8 bytes, followed by the new reference data padded to 8 bytes. PIN change using admin state: Only 8 bytes of new reference data. A PIN with admin state flag must be verified for the operation to succeed. Byte value 00h or FFh can be used for padding.
LE	Empty

Table 38 - CHANGE REFERENCE DATA response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes: 0x9000 – Command successful

	0x6983 – Verification failed and/or no number of retries left PIN blocked. 0x63CX – Verification failed and/or X number of retries left.

4.9 RESET RETRY COUNTER

The RESET RETRY COUNTER command unblocks a PIN which has been blocked after too many unsuccessful verification trials. Successful unblocking requires the correct unblocking reference data (PUK).

Successful execution of this command sets the reference data into locked state if the relocking after reset retry counter option is used. The relocking flag is optionally set with the PUT DATA: INITIALISE PIN command.

Table 39 - RESET RETRY COUNTER command APDU

Byte	Value
CLA	00h
INS	2Ch
P1	00h: reset retry counter and set reference data
P2	PIN reference number 01h to 03h MyEID3: 01h to 0Eh
LC	00h: query status of unblocking reference data 10h: reset retry counter 08h: reset retry counter using admin state 20h: reset retry counter of a challenge/response PIN
Data	Using PUK: Empty or unblocking reference data padded to 8 bytes, followed by the new reference data padded to 8 bytes (alphanumeric PIN) or 24 bytes 3DES key (challenge/response PIN). Using admin state: New reference data padded to 8 bytes (alphanumeric PIN) or 24 bytes 3DES key (challenge/response PIN). Byte value 00h or FFh can be used for padding.
LE	Empty

Table 40 - RESET RETRY COUNTER response APDU

Byte	Value

Data	Empty
SW1 – SW2	Status Bytes

Table 41 - Response Status Byte values

SW1 – SW2	Description
0x9000	Command successful
0x6985	PIN locked – *CONDITIONS NOT SATISFIED*
0x6983	Verification failed and no number of retries, PUK blocked.
0x63CX	Verification failed and X number of retries left.

4.10 DEAUTHENTICATE

Table 42 – DEAUTHENTICATE command APDU

Byte	Value
CLA	00h
INS	2Eh
P1	00h: Deauthenticate PIN A0h: Deauthenticate admin state B0h: Deauthenticate global unblocker state
P2	P1 = 00h: Pin reference (01h-0Eh) or 00h to deauthenticate all PINs and states P1 <> 00h: P2 must be 00h
LC	Length of data field
Data	Empty
LE	Empty

Table 43 – DEAUTHENTICATE response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

4.11 MANAGE SECURITY ENVIRONMENT: RESTORE

The MANAGE SECURITY ENVIRONMENT: RESTORE command restores an empty or predefined Security Environment (SE).

Table 44 - MANAGE SECURITY ENVIRONMENT: RESTORE command APDU

Byte	Value

CLA	00h
INS	22h
P1	F3h: Restore SE
P2	00h: SE reference number (00h denotes an empty SE)
LC	Empty
Data	Empty
LE	Empty

Table 45 - MANAGE SECURITY ENVIRONMENT: RESTORE response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

4.12 MANAGE SECURITY ENVIRONMENT: SET

The MANAGE SECURITY ENVIRONMENT: SET command sets attributes in the current Security Environment (SE).

Table 46 - MANAGE SECURITY ENVIRONMENT: SET command APDU

Byte	Value
CLA	00h
INS	22h
P1	41h: computation, decipherment or key agreement 81h: encipher
P2	B6h: attributes of DST in data field B8h: attributes of CT in data field A4h: authentication template in the data field (use before GENERAL AUTHENTICATE)
LC	Empty or length of Data field
Data	Empty or concatenation of Control Reference Data Objects (CRDO)
LE	Empty

Table 47 - MANAGE SECURITY ENVIRONMENT: SET response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

Table 48 - Control Reference Data Objects (CRDO)

Tag	Meaning
80h	XXh: Algorithm Reference (see Table 49 for details). Use 00h for symmetric key operations, algorithm is selected according to key type.
81h	XXXXh: File Reference; File Identifier (FID) - only FIDs shall be used - identifies RSA EF to be used in PSO commands value according to PKCS#15: private key objects PKCS15Path.path
83h	P1=41h: Key unwrapping: XXXXh: Target File Reference; File Identifier (FID) - only FIDs shall be used - identifies the file which the unwrapped key material will be placed into. Other use cases: Not used, must not be present P1=81h: 00h: Symmetric key reference MyEID supports only one key in each key file
84h	P1=41h: 00h: Key Reference (references the private key in a key file) MyEID supports only one key in each key file P1=81h: 00h: Not used, must not be present
87h	P1=41h: Not used, must not be present. P1=81h: Initialisation vector

Table 49 - Values for Algorithm Reference

Algorithm Reference	Meaning
0Xh	No hash algorithm
1Xh	SHA-1 hash algorithm
2Xh	RFU (RIPEMD-160)
3Xh	RFU (SHA-224)
4Xh	RFU (SHA-256)
5Xh	RFU (SHA-384)
6Xh	RFU (SHA-512)
8Xh	Symmetric operations: PKCS#7 padding

X0h	RSA keys: Raw RSA operation * No input or output data formatting, padding, or hash encapsulation is applied. PSO: COMPUTE DIGITAL SIGNATURE: 1. Input data must be equal to the RSA key modulus length. 2. PKCS#1 RSASP1 signature primitive is applied (private key operation). PSO: DECIPHER 1. PKCS#1 RSADP decryption primitive is applied (private key operation).
X1h	RFU
0Ah	Key wrapping or unwrapping
X2h	RSASSA-PKCS1-v1-5 signature scheme According to PKCS#1 v2.0 with RSA algorithm, compatible with PKCS#15 v1.5 PSO: COMPUTE DIGITAL SIGNATURE: - The data (hash code) is encapsulated into a DigestInfo ASN.1 structure according to the selected hash algorithm. If no algorithm is selected (algorithm reference = 02h) then no encapsulation is done by the applet. - DigestInfo is padded to RSA key modulus length according to PKCS#1 v1.5, block type 01h. Size of the DigestInfo must not exceed 40% of the RSA key modulus length. - PKCS#1 RSASP1 signature primitive is applied (private key operation). RSAAES-PKCS1-v1-5 encryption scheme According to PKCS#1 v2.0 with RSA algorithm, compatible with PKCS#15 v1.5 PSO: DECIPHER: - PKCS#1 RSADP decryption primitive is applied (private key operation). PKCS#1 v1.5 padding is removed
X3h	RFU (DSA)
X4h	Elliptic curve operation (ECDSA or ECDH)

* Raw RSA operation is not supported on 2048 bit keys when computing a digital signature.

4.13 PERFORM SECURITY OPERATION: COMPUTE DIGITAL SIGNATURE

The PSO: COMPUTE DIGITAL SIGNATURE command computes a digital signature. The Secure Environment (SE) attributes Algorithm and File Reference must be previously set with MSE: SET command.

Table 50 - PSO: COMPUTE DIGITAL SIGNATURE command APDU

Byte	Value
CLA	00h: single or last data block 10h: chained APDU, more data to follow
INS	2Ah
P1	9Eh: return digital signature data

P2	9Ah: data field contains data to be signed
LC	Length of data field
Data	SE Algorithm Reference = 00h: * - Raw data, data length must be equal to the RSA key modulus length. SE Algorithm Reference = 02h: - DigestInfo data, data is padded by the applet to match the length of the RSA key modulus. SE Algorithm Reference = 12h: - SHA-1 Hash data, the hash is encapsulated into a DigestInfo structure and then padded to the full length of the RSA key modulus. SE Algorithm Reference = 04h: - Raw data, data length must be equal to the ECC key length.
LE	Empty or length of response data

* 00h algorithm reference is not supported on 2048 bit keys.

Table 51 - PSO: COMPUTE DIGITAL SIGNATURE response APDU

Byte	Value

Data	Digital signature
	
	With ECC keys, the digital signature is returned in the following format:
	
	EC Signature:: = SEQUENCE OF {
	r INTEGER
	s INTEGER
	}

SW1 – SW2	Status Bytes

4.14 PERFORM SECURITY OPERATION: ENCIPHER

The PSO: ENCIPHER operation enciphers data transmitted in the command data field using symmetric encryption. The Secure Environment (SE) attributes Algorithm and File Reference must be previously set with MSE: SET command to select an AES or DES key file. The PSO: ENCIPHER command can also be used for key wrapping. In key wrapping the command data field is left empty and the card encrypts key material from a key file defined in SE's Target File attribute.

Table 52 - PSO: ENCIPHER command APDU

Byte	Value
CLA	00h: DO FINAL (finalises the cipher) 10h: UPDATE CIPHER (more data is coming in subsequent APDUs)

INS	2Ah
P1	84h: return encrypted data
P2	80h: data field contains plaintext 00h: key wrapping: the data field is absent
LC	Length of data field according to key algorithm and length
Data	Plain text data to encipher
LE	Empty or length of response data

Table 53 - PSO: ENCIPHER response APDU

Byte	Value
Data	Encrypted data
SW1 – SW2	Status Bytes

4.15 PERFORM SECURITY OPERATION: DECIPHER

The PSO: DECIPHER command decrypts an encrypted data. The Secure Environment (SE) attributes Algorithm and File Reference must be set previously with MSE: SET command. PSO: DECIPHER command can also be used for unwrapping keys into the card. In this case, the command does not return the deciphered data, but places it into a key EF defined in SE's Target File attribute.

The cryptogram must be presented in two parts when a 2048 bit RSA key is used.

Table 54 - PSO: DECIPHER command APDU

Byte	Value
CLA	RSA and ECC keys: 00h: single or final block of data 10h: chained APDU, more data to follow AES and DES keys: 00h: DO FINAL (finalises the cipher) AES and DES keys: 10h: UPDATE CIPHER (indicates that more data is coming)
INS	2Ah
P1	80h: return decrypted data 00h: the response data field is absent
P2	86h: data field contains padding indicator concatenated with the data to be decrypted (RSA +keys) 84h: data field contains encrypted data (AES and DES keys)
LC	81h: Length of data field
Data	Data for smaller than 2048 bit keys: 00h (pad indicator) || cryptogram

	Data for 2048 bit keys: 81h (pad and first half indicator) || first half of cryptogram 82h (pad and second half indicator) || second half of cryptogram MyEID > 4.5, 2048 bit and longer keys: 00 (pad indicator) || cryptogram following subsequent parts of the cryptogram split into APDUs with maximum 255 bytes of data. * AES and DES keys: encrypted data
LE	Empty or length of response data

* The old method of splitting a 2048 bit cryptogram can also be used with MyEID 4.5.

Table 55 - PSO: DECIPHER response APDU

Byte	Value
Data	SE Algorithm Reference = 00h: - Decrypted data is returned including formatting. SE Algorithm Reference = 02h: - Decrypted data, PKCS#1 v1.5 padding is removed by the applet.
SW1 – SW2	Status Bytes

4.16 GENERAL AUTHENTICATE

In MyEID applet GENERAL AUTHENTICATE command from ISO 7816 is used to perform Elliptic Curve Diffie-Hellman (ECDH) key agreement. It takes the other party's public point as input and returns the shared secret in the response APDU.

Table 56– GENERAL AUTHENTICATE command APDU

Byte	Value
CLA	00h
INS	86h
P1	00h
P2	00h
LC	Length of data field
Data	Data Objects in the Dynamic Authentication Template (Tag '7C'), see Table 58.
LE	Empty

Table 57 – GENERAL AUTHENTICATE response APDU

Byte	Value
Data	Shared secret
SW1 – SW2	Status Bytes

Table 58 – Dynamic Authentication Template

Byte	Length	Value
7Ch	1	Dynamic Authentication Template tag
Length	1	Length of the structure
80h	0-1	Optional "Witness" tag. (RFU for nonce)
Length	0-1	RFU, must be 00h if tag 80h is present.
Nonce	0	RFU, must be empty
85h	1	Exponential tag
Length	1	Length of public point
Public point	1	PUBLIC POINT in uncompressed format [04h, x, y]

Personalisation and Management Commands

5.1 PUT DATA: INITIALISE APPLET

The PUT DATA: INITIALISE APPLET command initialises the applet's file system. The first time this command is called after the applet has been installed; it also allocates the requested memory space for the file system.

After a successful execution, the applet will be in the personalisation state, and all files will be in the Created State. In the personalisation state all Access Conditions are disabled, i.e. the applet does not check them against the File Security Attributes in the files. In other words, the applet behaves as if all the Security Attributes were set to ALWAYS (00h).

Table 59 – PUT DATA: INITIALISE APPET command APDU

Byte	Value
CLA	00h
INS	DAh
P1	01h
P2	E0h
LC	08h
Data	File system initialisation parameters, see Table 61 for

	details.
LE	Empty

Table 60 – PUT DATA: INITIALISE APPLET response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

Table 61 – File system initialisation parameters

Bytes	Length	Value
File system size MyEID3: Maximum number of files	2	XXXXh MyEID3: 0080h to 0200h (128 to 2048 files), values outside the range will be changed to the closest allowed value by the applet before execution.
MF ACL	3	See SELECT FILE command for details.
DF 5015 ACL	3	See SELECT FILE command for details.

5.2 PUT DATA: INITIALISE PIN

The PUT DATA: INITIALISE PIN command initialises the requested PIN and its unblocking reference data (PUK). This command is allowed only in Creation State.

Table 62 – PUT DATA: INITIALISE PIN command APDU

Byte	Value
CLA	00h
INS	DAh
P1	01h
P2	01h: PIN #1 02h: PIN #2 03h: PIN #3 MyEID3: 01h to 0Eh
LC	10h to 13h
Data	PIN initialisation parameters, see Table 64 for details.
LE	Empty

Table 63 – PUT DATA: INITIALISE PIN response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

Table 64 – PIN initialisation parameters

Bytes	Length	Value
PIN reference data	8	Padded with 00h or FFh
Unblocking reference data	8	Padded with 00h or FFh
* PIN retry counter	1	Lower nibble indicates the number of failed PIN verifications before the PIN is blocked. For example: X3h = 3 trials. X = RFU. Default value: 05h = 5 trials
* PUK retry counter	1	Lower nibble indicates the number of failed PUK verifications before the PUK is blocked. For example: XAh = 10 trials. X = RFU. Default value: 0Ah = 10 trials
* PIN flags	1	bit0: 0 = normal mode (default), 1 = PIN is locked after initialisation. PIN will be unlocked after successful execution of the Change Reference Data command. bit1: 0 = normal mode (default), 1 = PIN will be locked after a Reset Retry Counter command. bit2: Allow global unblocking (default = 0, not allowed) bit3: Global unblocker (default = 0, no global unblocker) bit4: Allow admin changing and unblocking (default = 0, not allowed) bit5: Administrator PIN (default = 0, is not admin PIN) bit7: Allow PIN type change (default = 0, not allowed)
* PIN type	1	00h = Alphanumeric PIN 01h = Grid PIN 02h = Challenge / response PIN
* GridPIN grid size	1	
* Minimum length of PIN	1	1-8. Takes effect when creating new PINs and when a PIN is changed.
* Minimum length of PUK	1	1-8. Default = 4
C/R pins only: Challenge / response key	24	Triple-DES key (192 bits)

* Optional field

5.3 PUT DATA: INITIALISE PIV EMULATION

The PUT DATA: INITIALISE PIV EMULATION command activates the PIV/CIV interface and maps selected keys and certificates to corresponding PIV objects.

Before issuing this command, the keys and certificates to be used must be loaded on the card using MyEID command interface. Mapping objects is optional. For example, if only authentication key and certificate are needed, other FIDs can be set to 0000 in the PIV Initialisation Parameters structure.

After execution of this command, PIV compatible applications can recognize the cards as a PIV card. The PIV interface can be selected with the SELECT command, using the following AID: A00000030800001000.

Table 65 – PUT DATA: INITIALISE PIV EMULATION command APDU

Byte	Value
CLA	00h
INS	DAh
P1	01h
P2	50h: Initialise PIV emulation
LC	14h
Data	PIV Initialisation Parameters, see Table 66 for details.
LE	Empty

Table 66 – PIV Initialisation Parameters

Bytes	Length	Value
State	1	80h = PIV emulation activated 00h = PIV emulation deactivated 0x7F = RFU
ACL	3	Access Control List
PIV Authentication Key FID	2	FID of the key EF that is accessible as PIV Authentication Key via the PIV/CIV interface
PIV Authentication Certificate FID	2	FID of the EF that is accessible as PIV Authentication Certificate via the PIV/CIV interface
Card Authentication Key FID	2	FID of the key EF that is accessible as Card Authentication Key via the PIV/CIV interface
Card Authentication Certificate FID	2	FID of the EF that is accessible as Card Authentication Certificate via the PIV/CIV interface
Signature Key FID	2	FID of the key EF that is accessible as Signature Key via the PIV/CIV interface
Signature Certificate FID	2	FID of the EF that is accessible as Signature Certificate via the PIV/CIV interface

Management Key FID	2	FID of the key EF that is accessible as Management Key via the PIV/CIV interface
Management Certificate FID	2	FID of the EF that is accessible as Management Certificate via the PIV/CIV interface.

5.4 ACTIVATE APPLET

The ACTIVATE APPLET command causes the applet to exit the Personalisation State and enter the Use State. All the files are changed from Created Status to Activated Status. All Access Conditions become active and fully functional.

The PIN that has been associated with MF's Recreate MF access condition must be initialised before executing the ACTIVATE APPLET command.

Table 67 – ACTIVATE APPLET command APDU

Byte	Value
CLA	00h
INS	44h
P1	04h: applet AID in data field
P2	00h
LC	01h – 10h: Length of AID (data field)
Data	Applet AID (A000000063504B43532D3135)
LE	Empty

Table 68 – ACTIVATE APPLET response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

5.5 CREATE FILE

The CREATE FILE command creates EF's and DF's into the current DF. After a successful creation the created file is set as the current file. If the applet is in the Use State, the applet is also set to Temporary Personalisation State and the created file is set to Created State.

Table 69 – CREATE FILE command APDU

Byte	Value

CLA	00h
INS	E0h
P1	00h
P2	00h
LC	19h – 31h: Length of data field
Data	File Control Parameters (FCP), see Table 69 for details.
LE	Empty

Table 70 – CREATE FILE response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes: 0x6A89 – File already exists

The following table shows the mandatory position and length of each parameter; they must not be rearranged and the only optional parameter is the DF name that can be used when creating DF's.

Table 71 – File Control Parameters (FCP)

Byte	Length	Value
FCP tag	1	62h
Length	1	17h: EF's 17h - 2Fh: DF's
File Size tag	1	80h: transparent EF 81h: DF and key EF
Length	1	02h

File Size	2	Transparent EF's: XXXXh: File size in bytes RSA key EF's: 0400h or 0800h*: size of modulus in bits (in 0040h increments) ECC key EF's: 0080h to 0209h: key size in bits (field size) DES key EF's: 0040h: 56 bit DES key 0080h: 128 bit 3DES key 00C0h: 196 bit 3DES key AES key EF's: 0080h: 128 bit key 0100h: 256 bit key DF's: 0000h: Not applicable
File Description tag	1	82h
Length	1	01h
File Descriptor	1	01h: transparent EF 11h: RSA key EF 19h: DES key EF 22h: ECC key EF 29h: AES key EF 38h: DF
File Identifier tag	1	83h
Length	1	02h
File Identifier	2	XXXXh: FID
Security Attributes Tag	1	86h
Length	1	03h
Security Attributes	3	See SELECT FILE command for details
Proprietary Information tag	1	85h: File status byte
Length	1	02h

Proprietary Information	2	First status byte: X0h: for private RSA and EC key EF's X = AC to clear after RSA operation. The remaining bits are RFU for EF Key, and for other file types the whole byte is RFU. Second status byte: Transparent EF's: version < 3.5: 00h (RFU) version >= 3.5: Transparent EF's and key EF's: See table 72
Life Cycle Status tag	1	8Ah
Length	1	01h
Life Cycle Status	1	00h: RFU
DF Name tag	1	84h: Optional tag, can only be used when creating DF's.
Length	1	01h – 10h
DF Name	1-16	

* 0800h is supported from MyEID applet version 2.2.0 onwards.

Table 72 – FCP proprietary information flags (bits 0-7)

Bit	Hex value	Description
0	01h	RFU
1	02h	RFU
2	04h	Auto size flag – the file grows dynamically when writing past end of file. New in MyEID 4.
3	08h	RFU
4	10h	Admin generate key (RSA/EC key EF)
5	20h	Admin recreate MF (MF), admin delete file (other file types)
6	40h	Admin create EF's (MF/DF), Admin update (EF), Admin PUT DATA (RSA/EC key EF)
7	80h	Admin create DF's (MF/DF), Admin read (EF), Admin use key (RSA/ECC key EF)

5.6 DELETE FILE

The DELETE FILE command removes the currently selected file if the required Access Condition is fulfilled. In case that the file to be deleted is a DF, it is deleted along with its full sub-tree, including all child DF's and EF's.
After a successful file removal the file space is de-allocated and can be reused for new files.

Table 73 – DELETE FILE command APDU

Byte	Value
CLA	00h
INS	E4h
P1	00h
P2	00h
LC	00h
Data	Empty
LE	Empty

Table 74 – DELETE FILE response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

5.7 GENERATE PUBLIC KEY PAIR

The GENERATE PUBLIC KEY PAIR command generates and stores a new key into an existing key EF. The key length (length of modulus) is given when the key EF is created; see CREATE FILE command for details. Before the command can be executed successfully, the related Access Condition must be fulfilled. The command returns the modulus or public point of the new key.

Table 75 – GENERATE PUBLIC KEY PAIR command APDU

Byte	Value
CLA	00h
INS	46h
P1	00h
P2	00h
LC	RSA keys: 05h or 07h: Length of data field EC keys: 00h
Data	RSA keys: TLV with value of public exponent, see Table 77 for details. ECC keys: Empty
LE	Empty or length of modulus in bytes

Table 76 – GENERATE PUBLIC KEY PAIR response APDU

Byte	Value
Data	RSA keys: Modulus ECC keys: a TLV structure containing the public point in uncompressed format: [tag 86h, length, 04h, X, Y]

SW1 – SW2	Status Bytes

Table 77 – Generate public key pair command data parameters for RSA keys

Bytes	Length	Value
Outer sequence Tag	1	30h
Length	1	03h or 05h
Public exponent tag	1	02h
Length	1	01h or 03h
Public Exponent	1 – 3	03h or 010001h **MyEID3: supports only 010001h**

5.8 PUT DATA: LOAD KEY

The PUT DATA: LOAD KEY command stores an externally generated key to the applet. A key EF must be selected first with the SELECT FILE command.

All the key components must be loaded to create a valid key.

All MyEID versions support the loading of private RSA keys in the CRT format. The MyEID3 version also supports the private exponent and modulus format. Only one private RSA key can be loaded into a key file together with a public key.

Table 78 – PUT DATA command APDU

Byte	Value
CLA	00h: single or final block of data 10h: chained APDU, more data to follow
INS	DAh
P1	01h

P2	RSA key components: 80h: modulus, N *4 81h: public exponent, E 83h: prime 1, P 84h: prime 2, Q 85h: *d* mod (*p* - 1), DP1 86h: *d* mod (*q* - 1), DQ1 87h: *q*-1 mod *p*, PQ 88h: first half of 2048 bit modulus 89h: second half of 2048 bit modulus *1 MyEID3 adds support for: 82h: private exponent, D *3, *4 8Ah: first half of 2048 bit key private exponent, D 8Ah: second half of 2048 bit key private exponent, D ECC key components: 86h: Public key (a point denoted as PP on the curve, equal to x times PB where x is the private key, coded on 2z bytes) 87h: Private key Note: domain parameters are set according to NIST/SEC named curves. Symmetric keys (AES and DES): A0h: load symmetric key. Data field contains the symmetric key data.

LC	Length of Data field
	
	P2 = 80h: **80h**
	P2 = 81h: **01h** or **03h**
	P2 = 83h, 84h, 85h, 86h, 87h: **40h or 80h** *2
	P2 = 88h, 89h: **80h**
	
	MyEID3:
	P2 = 80h, 82h: **40h to C1h**
	P2 = 81h: **01h** or **03h**
	P2 = 83h, 84h, 85h, 86h, 87h: **20h or C1h**
	P2 = 88h, 89h: **81h**
	P2 = 8Ah, 8Bh: 81h
	
	ECC-keys:
	Length of ECC key component
	
	Symmetric keys:
	Key length. Must be equal to the file size set in CREATE FILE command.

Data	Component
LE	Empty

*1 The second half of 2048 bit modulus must be preceded by the first half and the order of the commands must not be reversed.
*2 80h is used for the 2048 bit key components.
*3 If a private key is loaded as a modulus and a private exponent, then only the public exponent is required. The private exponent must not be loaded with a CRT formatted key, loading it will remove the CRT components.
*with MyEID 4.5 and newer, for 2048 bit and longer keys use APDU chaining and P2=80h for modulus and P2=82h for private exponent. For 2048 bit keys also the old MyEID3 method is available for compatibility.

Table 79 – PUT DATA: LOAD KEY response APDU

Byte	Value
Data	Empty
SW1 – SW2	Status Bytes

5.9 GET CHALLENGE

The GET CHALLENGE command can be used to generate a random challenge to be used for authenticating a challenge/response PIN or to generate random data for any purpose by using the card’s random number generator.

Table 80 – GET CHALLENGE command APDU

Byte	Value
CLA	00h

INS	84h
P1	00h: Get random data 02h: Get challenge data for authenticating a challenge/response PIN.

P2	P1=00h: 00h
	P2=02h: PIN number

LC	00h
Data	Empty
LE	Number of bytes to return

Table 81 – GET CHALLENGE response APDU

Byte	Value
Data	Requested number of random bytes
SW1 – SW2	Status Bytes