# Reference Manual

# Belpic V1.7 eID card

# Table of content

# Table of figures

# 1 Introduction

## 1.1 Purpose

This document is the public reference manual of Belpic electronic identity card, deployed to Belgian citizens. This card can be used in Belgium's electronic Government (eGov) applications and Public Key Infrastructure (PKI) system.

This reference manual provides a detailed description of some useful security objects, security protocols and applicative commands managed by the card in its operational life.

## 1.2 Who should read this book

This reference manual is designed for developers building software applications based on this eID card. Typical applications may feature check of identity and / or personal information, PIN management (verification or change), user authentication, and digital signature creation.

To fully benefit from this document's contents, knowledge of the following is helpful:

- Smart card technology
- Java cards
- Cryptography
- PKI systems

## 1.3 References

The following table gives the references of the documents which content is applied in this document:

| Ref | Document Title | Description |
|---|---|---|
| **[ER1]** | ISO CEI 7816-3 | Integrated circuits cards with contacts – Electronic signals and transmission protocols – 1997 |
| **[ER2]** | ISO CEI 7816-4 | Integrated circuits cards with contacts – Inter-industry commands for interchange – 1995 + amendment 1- 1997 |
| **[ER3]** | ISO CEI 7816-5 | Integrated circuits cards with contacts – Inter-industry commands for interchange – 1996 |
| **[ER4]** | ISO CEI 7816-8 | Integrated circuits cards with contacts – Security related inter-industry commands – 1999 |
| **[ER5]** | ISO CEI 7816-9 | Integrated circuits cards with contacts – Additional inter-industry commands and security attributes – 2000 |
| **[ER6]** | Java Card 2.2.1 | Sun Java Card™ 2.2.1 Application Programming Interface - October 2003 |
| **[ER7]** | Global Platform | Card Specification – version 2.1.1 - March 2003 |
| **[ER8]** | PKCS#15 1.1 | Cryptographic Token Information Standard, version 1.1, RSA laboratories, June 2000<br>URL:ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-15/pkcs-15v1_1.pdf |
| **[ER9]** | ISO CEI 9564-1 | Banking – Personal Identification Number (PIN) management and security – 2002 |
| **[ER10]** | PKCS#1 2.1 | RSA Cryptography Standard, version 2.1, RSA laboratories, June 2002<br>URL: ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf |

## 1.4 Terminology

| | |
|---|---|
| Nibble | Half part of a byte (4 bits) |
| Digit | Nibble taking values between 0 and 9 |
| Temporary mute state | Mute until next reset |

## 1.5 Acronyms and Abbreviations

For the purposes of this document, the following abbreviations apply:

| | |
|---|---|
| **AID** | Application IDentifier |
| **APDU** | Application Protocol Data Unit |
| **CA** | Certification Authority |
| **Cert** | Certificate |
| **CHL** | Challenge |
| **CHV** | Card Holder Verification |
| **CLA** | APDU Class byte |
| **DF** | Dedicated File |
| **DO** | Data Object |
| **DST** | Digital Signature Template |
| **DTBS** | Data To Be Signed |
| **EID, eID** | Electronic Identity Device |
| **EF** | Elementary File |
| **FCI** | File Control Information |
| **GP** | Global Platform (see [ER7]) |
| **IFD** | Interface device (e.g. smart card reader) |
| **ISO** | International Organization for Standardization |
| **MF** | Master File |
| **MSB** | Most Significant Byte |
| **MSE** | Manage Security Environment |
| **OS** | Operating System (of smart card) |
| **PIN** | Personal Identification Number |
| **PKCS** | Public Key Cryptography Standard |
| **PKI** | Public Key Infrastructure |
| **PSO** | Perform Security Operation |
| **PSO:CDS** | PSO:Compute Digital Signature |
| **PuK** | RSA public key |
| **PrK** | RSA private key |
| **RES** | Result |
| **RFU** | Reserved for Future Use |
| **RSA** | Rivest, Shamir, Adleman (creators of crypto- algorithm) |
| **SE** | Security Environment |
| **SHA** | Secure Hash Algorithm |
| **SW** | Status Word |
| **TLV** | Tag Length Value |

## 1.6 Conventions

### 1.6.1 General

- All the values between quotes are in hexadecimal notation, unless otherwise specified.
- The values are presented with MSB first.
- The symbol || represents a concatenation.

### 1.6.2 Symbols

ℹ Information field

STOP Important note

✓ feature supported

✗ feature not supported

## 1.7 Remarks on standards used

***PKCS#1*** version used is 2.1 (see [ER10] for algorithms description).
Following signature schemes are available:

- ***RSASSA-PKCS1-v1.5 using SHA1 or MD5 or SHA256***
- ***RSASSA-PSS using SHA1 or SHA256*** defined in ***PKCS#1-v2.1***

ℹ The salt lengths, used in encoding method EMSA-PSS, are 20 bytes for SHA1 and 32 bytes for SHA256 (see [ER10] §9.1 EMSA-PSS, Notes n°4, p34 : «*Typical salt lengths in octets are hLen (the length of the output of the hash function Hash) …*»

ℹ MD5 hashing is provided for backward compatibility only, it should not be used anymore.

Encryption scheme is***:***

- ***RSAES-PKCS1-v1.5 using SHA1***

## 1.8 Product version

This document describes the version 1.7 of the eID card product.

Product version is given by the “**applet version**” byte returned by the command “***Get Card Data***” (see §6.9) set to ‘**17**’.

# 2 Belpic eID card main features

The Belpic eID card mainly features two applications:

- Electronic signature
- Electronic identification

Both these applications are powered by PKI mechanisms.

Digital signatures are used to ensure the integrity and authenticity of a message and / or authenticate the card holder. These signatures are made using RSA private keys securely stored within the card. The card also securely stores digital certificates, issued by a trusted body known as a certification authority (CA) and typically used in PKI authentication schemes.

The file system of Belpic eID card, containing files and data objects (such as PINs and keys), is described in a separate document. Some card features are mentioned hereafter:

- The MF is the selected DF after a reset
- It is not possible to extend the existing file system
- It is not possible to delete a file within the existing file system
- The card does not process the PKCS#15 information contained in the Belpic file system; this information is only managed and used by the external application (middleware)
- Only the transparent elementary files are supported
- RSA keys supported have 2,048 bits length
- In all signature algorithms, the hash computation is done off card

## 2.1 Applet AID

| | |
|---|---|
| **Applet Instance AID** | A0 00 00 00 30 29 05 70 00 AD 13 10 01 01 FFh |

This AID can be used to select the Belpic applet.

Notes:

- After card reset, Belpic applet is implicitly selected because the applet is installed with the default-selected privilege
- Belpic applet is only selectable on logical channel 0

## 2.2 Security conditions

Access to Belpic data and features is submitted to control & verification of some security conditions. The following table lists the different security conditions managed by the eID card:

| Security Condition | Meaning |
|---|---|
| **NEV** | The operation is never allowed |
| **ALW** | The operation is always allowed |
| **CHV** | The operation is only allowed during the operational phase after a successful PIN verification (refer to §5.1) |

STOP **Only one security condition can be fulfilled at a time.**

# 3 PIN code

## 3.1 Description

The applet manages a user PIN:

| PIN | PIN reference | Max. number of attempts | Application |
|---|---|---|---|
| **PIN$_{\text{cardholder}}$** | ‘01’ | 3 | ❒ Change PIN$_{\text{cardholder}}$<br>❒ Non-repudiation signature with non-repudiation key<br>❒ Signature with authentication key |

**Table 1 - Belpic user PIN**

**Remark:**

**1)** One PIN (PIN$_{\text{cardholder}}$) for both signature keys (authentication key and non-repudiation key).

**2)** The PIN$_{\text{cardholder}}$ behaviour is different for authentication key (“PIN once”) and non-repudiation key (“PIN just before”, see §5.1).

## 3.2 Format

The PIN is 8-bytes string with the following format (by nibble) as defined in [ER7] and [ER9]:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | L | P | P | P | P | P/’F’ | P/’F’ | P/’F’ | P/’F’ | P/’F’ | P/’F’ | P/’F’ | P/’F’ | ‘F’ | ‘F’ |

| Nibble | Signification |
|---|---|
| C | Control parameter, contains always ‘2’ |
| L | Length of the PIN (in nibbles) ; from ‘4’ to ‘C’ |
| P | Four mandatory digits |
| P/’F’ | The rest of the PIN digits depending on the length, ‘F’ else |
| ‘F’ | Padding always contains ‘F’ |

**Table 2 - PIN format**

| PIN | Minimum number of digits |
|---|---|
| PIN$_{\text{cardholder}}$ | 4 |

**Table 3 - PIN length (in digits)**

# 4 Keys Description

## 4.1 RSA Private Keys

All private RSA keys are 2048 bit-long (in former product version V1.1, they were 1024 bit-long).

The 2 following RSA private keys are present in Belpic card:

| ***RSA Key*** | ***Key Reference*** |
|---|---|
| Authentication key | ‘82’ |
| Non-Repudiation key | ‘83’ |

**Table 4 – RSA private keys**

- **Authentication key**

This key is a private RSA key with reference **‘82’**.
This Citizen key is a **signature key** used in **PSO:CDS** command.
The key access conditions are the following (See access condition coding in chapter §2.2):

| | ***PSO:CDS*** |
|---|---|
| **Access Conditions** | CHV<br>PIN$_{\text{cardholder}}$ «once»<br>(see §5.1) |

**Table 5 – Authentication key access conditions**

- **Non-Repudiation key**

This key is a private RSA key with reference **‘83’**.
This Citizen key is a **signature key** used in **PSO:CDS** command.
The key access conditions are the following (See access condition coding in chapter §2.2):

| | ***PSO:CDS*** |
|---|---|
| **Access Conditions** | CHV<br>PIN$_{\text{cardholder}}$ «just before», (see §5.1) |

**Table 6 – Non-Repudiation key access conditions**

# 5 Authentication processes

## 5.1 PIN Verification

The PIN verification consists in verifying a PIN code from an external application against the reference data stored into the EID card. If this verification process succeeds then the external application can get access to the authorized data and functions in the Belpic EID card.

The CHV process uses the ***MVP: Verify (ISO 7816-4)*** APDU command:

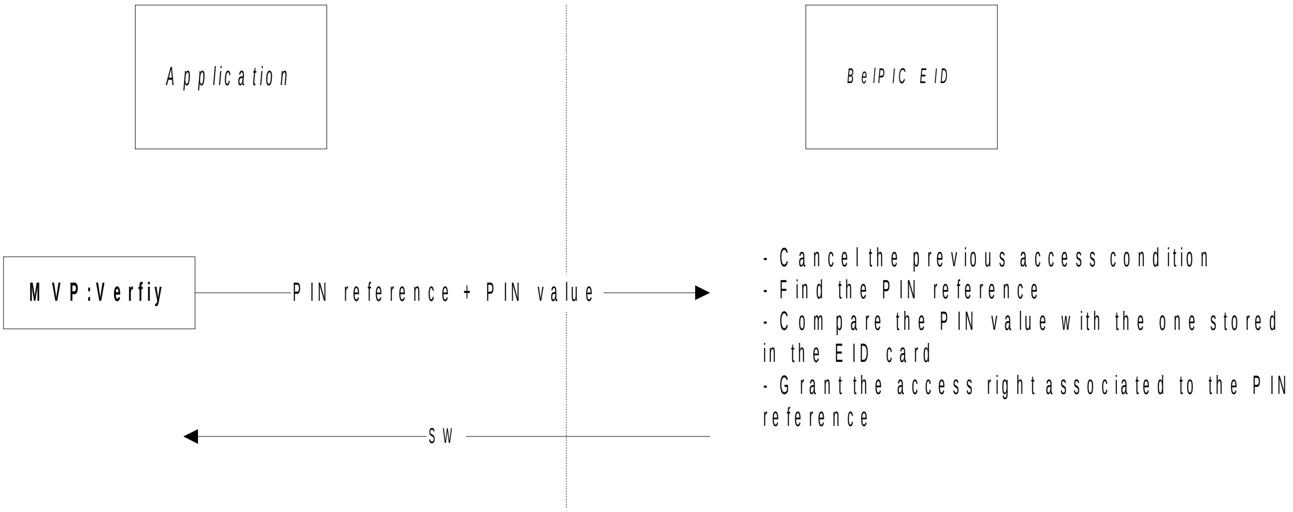
Application
BelPIC EID
MVP:Verfiy
PIN reference + PIN value
- Cancel the previous access condition
- Find the PIN reference
- Compare the PIN value with the one stored in the EID card
- Grant the access right associated to the PIN reference
SW

**Figure 1 - PIN verification process**

### *User PIN behaviour*

User PIN can be used in 2 modes:

- “PIN once”: the PIN access right must have been granted once, at any time before calling a command
- “PIN just before”: the PIN access right must be granted immediately before calling a command – each time

## ***User PIN and Signature keys***

The user PIN shall be verified before signing with non-repudiation and authentication keys.

We use the following conventions in examples described in this chapter:

- **A yellow Verify(PIN) is a Verify(PIN) for non-repudiation signature sequence.**

A non-repudiation signature sequence is a Signature just after a Verify PIN in a non-repudiation Security Environment (MSE: SET command with non-repudiation key).

Examples of non-repudiation signature sequence:

- ❒ Reset, …,MSE($\text{Key}_{\text{non-rep}}$), Verify(PIN), Sign($\text{Key}_{\text{non-rep}}$)
- ❒ Reset, …,MSE($\text{Key}_{\text{non-rep}}$),…,[1] Verify(PIN), Sign($\text{Key}_{\text{non-rep}}$)

- **A blue Verify(PIN) is a Verify(PIN) for authentication signature sequence.**

An authentication signature sequence is a Signature in an authentication Security Environment (MSE: SET command with authentication key) with a Verify PIN performed at any time before the signature and outside a non-repudiation signature sequence.

Examples of Verify(PIN) for authentication sequence:

- ❒ Reset, …,Verify(PIN), …, MSE($\text{Key}_{\text{auth}}$), …,Sign($\text{Key}_{\text{auth}}$)
- ❒ Reset, …,MSE($\text{Key}_{\text{auth}}$), …, Verify(PIN), …,Sign($\text{Key}_{\text{auth}}$)
- ❒ Reset, …,Verify(PIN), …
- ❒ Reset, …,Verify(PIN), …, Read, …

- **In following examples the MSE:SET commands are implicitly performed before the signature.**

[1] It is assumed that ", …," means any Belpic commands except MSE, Verify, Sign (and commands resetting PIN or SE)

### “PIN once” mode

This mode is used when signing with the **authentication key**: a successful ***MVP:Verify(PIN)*** APDU command has to be executed at any time before performing the signature. The next times, the **MVP:Verify(PIN)** APDU command does not have to be executed anymore.
More generally, this mode is used each time the “CHV” access is required.

**Examples of required sequences:**

- Verify(PIN), Sign(Key$_{auth}$), …,[1] Sign(Key$_{auth}$)

**Examples of incorrect sequences:**
- No previous Verify(PIN)

- Sign(Key$_{auth}$) KO

### “PIN just before” mode

This mode is used when signing with the **non-repudiation key**: a successful ***MVP:Verify(PIN)*** APDU command has to be executed just before performing the signature. Every time the signature with this key has to be performed, the ***MVP:Verify(PIN)*** APDU command must be executed again.

**Examples of required sequences:**

- Verify(PIN), Sign(Key$_{non\text{-}rep}$), …, Verify(PIN), Sign(Key$_{non\text{-}rep}$)

**Examples of incorrect sequences:**
**-** No previous Verify(PIN) just before the signature

- Verify(PIN), …, Sign(Key$_{non\text{-}rep}$) KO
- Verify(PIN), …, Sign(Key$_{auth}$), Sign(Key$_{non\text{-}rep}$) KO
- Verify(PIN), Sign(Key$_{non\text{-}rep}$), Sign(Key$_{non\text{-}rep}$) KO
- Sign(Key$_{non\text{-}rep}$) KO

[1] it is assumed that “, …,” means any Belpic commands except MSE, Verify, Sign (and commands resetting PIN or SE)

## “PIN Mixed” mode

STOP **New PIN behavior when using both signature keys during the same card session:**
After a successful non-repudiation signature sequence (see definition above), a signature with authentication key is not allowed, unless a Verify PIN for using the authentication key has been done before.

**Examples of required sequences:**

In yellow colour Verify(PIN) for non-repudiation key.
In blue colour Verify(PIN) for authentication key.

- ❐ Verify(PIN), …, Verify(PIN), Sign(Key$_{\text{non-rep}}$), …, Sign(Key$_{\text{auth}}$)
- ❐ Verify(PIN), Sign(Key$_{\text{auth}}$), Verify(PIN), Sign(Key$_{\text{non-rep}}$), …, Sign(Key$_{\text{auth}}$)
- ❐ Verify(PIN), …, Sign(Key$_{\text{auth}}$), Verify(PIN), Sign(Key$_{\text{non-rep}}$), …, Sign(Key$_{\text{auth}}$), …, Sign(Key$_{\text{auth}}$), Verify(PIN), Sign(Key$_{\text{non-rep}}$)
- ❐ Verify(PIN), Sign(Key$_{\text{non-rep}}$), … Verify(PIN), …, Sign(Key$_{\text{auth}}$)

Example with a Read Binary command protected by PIN once access conditions

- ❐ Verify(PIN), …, Verify(PIN), Sign(Key$_{\text{non-rep}}$), …, Sign(Key$_{\text{auth}}$), Read

**Examples of incorrect sequences:**

**-** No Verify(PIN) for authentication key during the session

- ❐ Verify(PIN), Sign(Key$_{\text{non-rep}}$), …, Sign(Key$_{\text{auth}}$) KO
- ❐ Verify(PIN), Wong Sign(Key$_{\text{non-rep}}$) KO, …, Sign(Key$_{\text{auth}}$) KO[1]
- ❐ Verify(PIN), …, Wrong Verify(PIN) KO, Sign(Key$_{\text{non-rep}}$) KO, …, Sign(Key$_{\text{auth}}$) KO[2]

Example with a Read binary command protected by PIN once access conditions

- ❐ Verify(PIN), Sign(Key$_{\text{non-rep}}$), …, Sign(Key$_{\text{auth}}$) KO, Read KO
- ❐ Verify(PIN), Sign(Key$_{\text{non-rep}}$), Read KO
- ❐ Verify(PIN), Read, Sign(Key$_{\text{non-rep}}$) KO…, Sign(Key$_{\text{auth}}$) OK

[1] Verify (PIN) only for non-repudiation signature because the signature is just after the Verify.

[2] A wrong PIN verification resets PIN access conditions. As it is the same PIN for 2 signature keys, the access is denied for authentication and non-repudiation signatures.

## 5.2 User Authentication

The user authentication is the process whereby the external application authenticates the cardholder.
The algorithm must use the ***PKCS#1 / RSASSA-PSS PKCS1-v2_1*** format (see [ER10]) to disable any attack based on a malformed message. SHA-256 is highly recommended but is not mandatory.

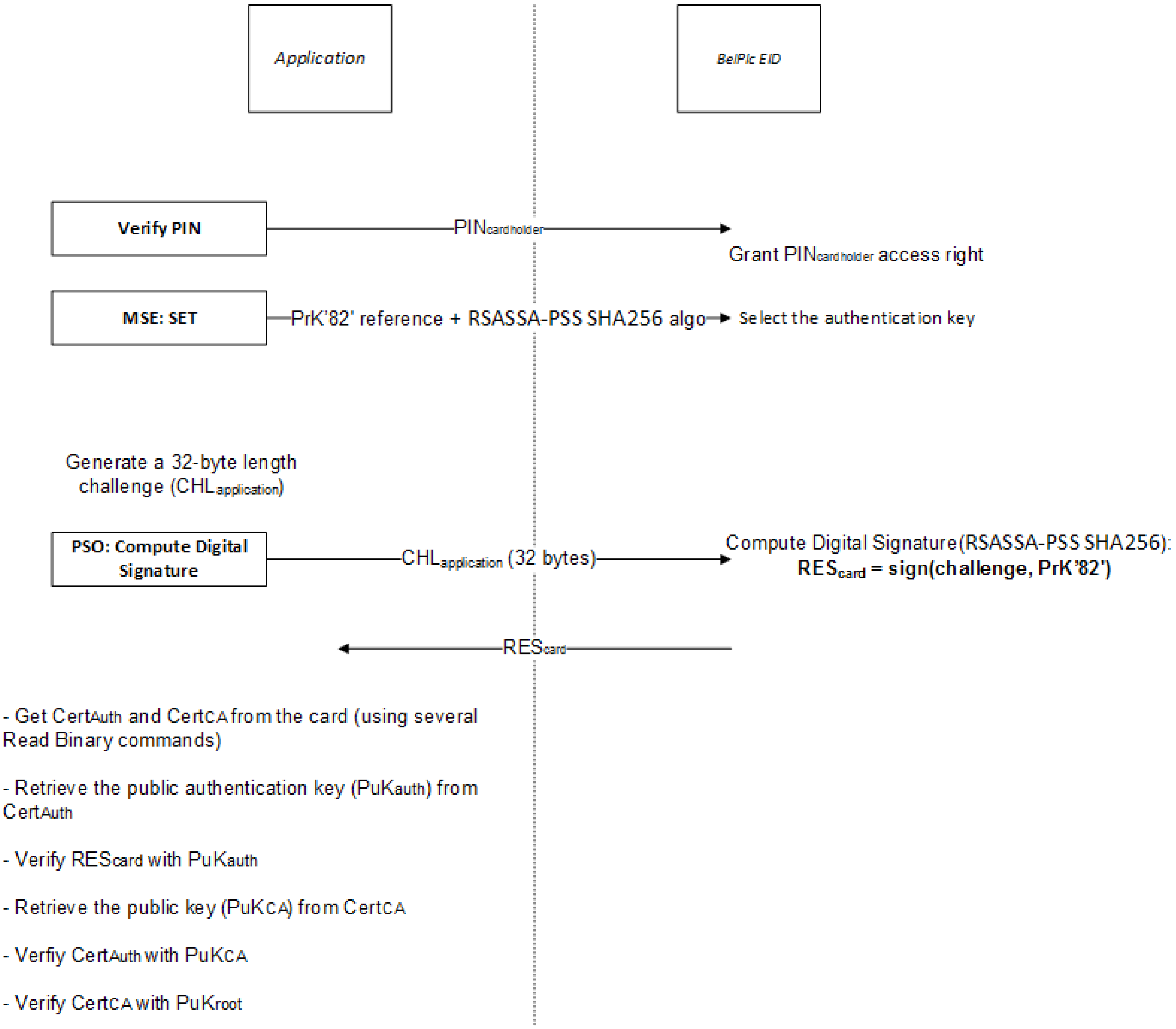
Application
BelPic EID
Verify PIN
PIN$_{cardholder}$
Grant PIN$_{cardholder}$ access right
MSE: SET
PrK'82' reference + RSASSA-PSS SHA256 algo
Select the authentication key
Generate a 32-byte length challenge (CHL$_{application}$)
PSO: Compute Digital Signature
CHL$_{application}$ (32 bytes)
Compute Digital Signature (RSASSA-PSS SHA256):
RES$_{card}$ = sign(challenge, PrK'82')
RES$_{card}$
- Get Cert$_{Auth}$ and Cert$_{CA}$ from the card (using several Read Binary commands)
- Retrieve the public authentication key (PuK$_{auth}$) from Cert$_{Auth}$
- Verify RES$_{card}$ with PuK$_{auth}$
- Retrieve the public key (PuK$_{CA}$) from Cert$_{CA}$
- Verfiy Cert$_{Auth}$ with PuK$_{CA}$
- Verify Cert$_{CA}$ with PuK$_{root}$

**Figure 2 - User Authentication process**

# 6 Command interface

During the operational phase, the EID card offers the following APDU interface:

| Instruction | CLA | INS | P1 | P2 | Lc | Input Data field | Le | Output Data field |
|---|---|---|---|---|---|---|---|---|
| GET RESPONSE | ‘00’ | ‘C0’ | ‘00’ | ‘00’ | - | - | Length | Response from previous command |
| SELECT FILE | ‘00’ | ‘A4’ | ‘02’ or ‘04’ or ‘08’ | ‘0C’ | Length | File ID or Absolute path or AID | - | - |
| READ BINARY | ‘00’ | ‘B0’ | OFF_H | OFF_L | - | - | Length | Read data |
| MVP: VERIFY | ‘00’ | ‘20’ | ‘00’ | Data Ref. | Length | Verification Data | - | - |
| MVP: CHANGE REFERENCE DATA | ‘00’ | ‘24’ | ‘00’ | Data Ref. | Length | Existing PIN$_{cardholder}$ \|\| New PIN$_{cardholder}$ | - | - |
| MSE: SET | ‘00’ | ‘22’ | ‘41’ | ‘B6’ | Length | Digital signature template | - | - |
| PSO: COMPUTE DIGITAL SIGNATURE | ‘00’ | ‘2A’ | ‘9E’ | ‘9A’ | Length | Data to be signed | Length | Signature |
| GET CARD DATA | ‘80’ | ‘E4’ | ‘00’ | ‘00’ | - | - | Length | Card information |
| LOG OFF | ‘80’ | ‘E6’ | ‘00 | ‘00’ | - | - | - | - |

**Table 7 - APDU Commands**

## 6.1 Get Response

The card currently only implements the protocol “***T=0***”, which does not support input and output data in the same command (cf. ISO 7816-3). Such commands – referred as ***case 4*** commands – must be called without the ***Le*** parameter and return a Status Word ’***61 xx***’ where ‘***xx***’ is the length of the output data to retrieve in an additional command. This protocol-level only command to use is ***Get Response***.

### Description

This command retrieves the data output by a ***case 4*** command.

No security conditions are required to perform this command.

### Command structure

| Instruction | CLA | INS | P1 | P2 | Le | Case |
|---|---|---|---|---|---|---|
| GET RESPONSE | ‘00’ | ‘C0’ | ‘00’ | ‘00’ | Length | 2 |

**Command ADPU**

| Field | Value |
|---|---|
| CLA | ‘00’ |
| INS | ‘C0’ |
| P1 | ‘00’ |
| P2 | ‘00’ |
| Le | Length of the data to retrieve |

**Table 8 – GET RESPONSE Command APDU**

**Response APDU**

| Field | Value |
|---|---|
| Data | Data to retrieve |
| SW1-SW2 | Status Bytes |

**Table 9 – GET RESPONSE Response APDU**

## Status bytes

| Value | Meaning |
|---|---|
| ’61 **xx**’ | **‘xx’** remaining bytes to retrieve from the card, through subsequent Get Response command |
| ’6C **xx**’[1] | ***Le*** too long, only ‘**xx’** bytes available, retrievable through subsequent Get Response command |
| ‘6D 00’ | Command not available within the current life cycle |
| ‘6E 00’ | CLA not supported |
| ’69 85’ | There are no data to retrieve |
| ’90 00’ | Normal ending of the command |

**Table 10 – GET RESPONSE Status bytes**

[1] *After a Status Word ‘6C XX”, if Get Response Le is different from ‘XX’, the Get Response chaining mechanism is not aborted*

## 6.2 Coding of algorithm

In the Belpic application, the algorithm and data object references used by the command interface shall be coded as follow:

| $b_7$ | $b_6$ | $b_5$ | $b_4$ | $b_3$ | $b_2$ | $b_1$ | $b_0$ | Meaning |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | RSASSA-PKCS1 (no predefined padding) |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | RSASSA-PKCS1-v1.5 using SHA1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | RSASSA-PKCS1-v1.5 using MD5 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | RSASSA-PKCS1-v1.5 using SHA256 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | RSASSA-PSS PKCS1-v2.1 using SHA1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | RSASSA-PSS PKCS1-v2.1 using SHA256 |
| X | X | 0 | 0 | 0 | 0 | 0 | 0 | 00 (others are RFU) |

**Table 11 - Algorithm references**

## 6.3 SELECT FILE Command (ISO 7816-4)

### Description

This command shall be used to select a file from the file system according to:

1. A file identifier (EF selection)

2. A path from MF (EF selection)

3. An application identifier (DF selection)

### Conditions of Use

No security conditions are required to perform this command.

### Command structure

| Instruction | CLA | INS | P1 | P2 | Lc | Le | Case |
|---|---|---|---|---|---|---|---|
| SELECT FILE | ‘00’ | ‘A4’ | ‘02’ or ‘04’ or ‘08’ | ‘0C’ | Length | - | 3 |

**Command ADPU**

| Field | Value |
|---|---|
| CLA | ‘00’ |
| INS | ‘A4’ |
| P1 | 1. ‘02’ (the data field shall contain a File ID)<br>2. ‘08’ (the data field shall contain an absolute path)<br>3. ‘04’ (the data field shall contain an AID) |
| P2 | ‘0C’: (No FCI to be returned) |
| Lc | Length of the subsequent data |
| Data | 1. File ID (2 bytes)<br>2. Absolute path without the identifier of the MF<br>3. Full AID (between 5 and 16 bytes) |
| Le | Empty |

**Table 12 – SELECT FILE Command APDU**

**Response APDU**

| Field | Value |
|---|---|
| Data | Empty |
| SW1-SW2 | Status Bytes |

**Table 13 – SELECT FILE Response APDU**

## Status bytes

| Value | Meaning |
|---|---|
| ’62 83’ | Selected file not activated |
| ’64 00’ | No precise diagnostic |
| ‘65 81’ | EEPROM corrupted (followed by a temporary mute state) |
| ‘6A 82’ | File not found |
| ‘6A 86’ | Wrong parameter P1-P2 |
| ‘6A 87’ | Lc inconsistent with P1-P2 |
| ’69 99’ / ’69 85’ | Attempt to select forbidden logical channel |
| ‘6D 00’ | Command not available within the current life cycle |
| ‘6E 00’ | CLA not supported |
| ’90 00’ | Normal ending of the command |

**Table 14 – SELECT FILE Status bytes**

## 6.4 READ BINARY Command (ISO 7816-4)

### Description

This command shall be used to read the content of a transparent EF.

### Conditions of Use

The security conditions to fulfil before performing this command depend on the currently selected file.

### Command structure

| Instruction | CLA | INS | P1 | P2 | Lc | Le | Case |
|---|---|---|---|---|---|---|---|
| READ BINARY | ‘00’ | ‘B0’ | OFF_H | OFF_L | - | Length | 2 |

**Command ADPU**

| Field | Value |
|---|---|
| CLA | ‘00’ |
| INS | ‘B0 |
| P1 | OFF_H: Higher byte of the offset (bit 8 =0) |
| P2 | OFF_L: Lower byte of the offset |
| Lc | Empty |
| Data | Empty |
| Le | Length of the data to read |

**Table 15 – READ BINARY Command APDU**

Note: If ***Le*** is equal to 0, then it is interpreted as 256 bytes.

**Response APDU**

| Field | Value |
|---|---|
| Data | Read data |
| SW1-SW2 | Status Bytes |

**Table 16 – READ BINARY Response APDU**

## Status bytes

| Value | Meaning |
|---|---|
| ’64 00’ | No precise diagnostic |
| ‘65 81’ | EEPROM corrupted (followed by a temporary mute state) |
| ’69 82’ | Security status not satisfied |
| ’69 85’ | Condition of use not satisfied (File not activated) |
| ’69 86’ | Command not allowed (no current EF) |
| ‘6B 00’ | Wrong parameter P1-P2 (offset outside the EF) |
| ‘6C’ XX | ***Le*** incorrect, ***XX*** indicates the expected length (hexadecimal value) |
| ‘6D 00’ | Command not available within the current life cycle |
| ’6E 00’ / ’68 81’ | Wrong coding of class byte |
| ’90 00’ | Normal ending of the command |

**Table 17 – READ BINARY Status bytes**

## 6.5 MVP:VERIFY Command (ISO 7816-4)

### Description

This command shall be used to fulfil a PIN access right. The EID card discards the previous access condition and then verifies data given by the external application against the referenced PIN:

- If the verification is successful the corresponding access right is granted
- If the verification is not successful the corresponding retry counter is decreased

This command is usually defined as a “PIN verification” procedure.

**MVP:VERIFY with Lc=0 management:**

- Presenting an empty PIN will NOT decrease the PIN retry counter.
- This feature can be used by middleware to get the PIN verification status:
  1. Verify PIN command with Lc=0 returns ’9000‘ if the referenced PIN is verified.
  2. Verify PIN command with Lc=0 returns ‘63Cx’ (with x = PIN tries remaining) if the referenced PIN is not verified.
  3. Verify PIN command with Lc=0 returns ‘6983’ if the referenced PIN is blocked.

**Compatibility note**

In V1.1 of the product, a MVP:VERIFY with an empty PIN buffer counts as a wrong PIN and so the retry counter is decremented.

### Conditions of Use

The PIN to verify must not be blocked.

### Command structure

| Instruction | CLA | INS | P1 | P2 | Lc | Le | Case |
|---|---|---|---|---|---|---|---|
| MVP:VERIFY | ‘00’ | ‘20’ | ‘00’ | ‘01’ | ‘00’ or ‘08’ | - | 3 |

**Command APDU**

| Field | Value |
|---|---|
| CLA | ‘00’ |
| INS | ‘20’ |
| P1 | ‘00’ |
| P2 | ‘01’ (PIN$_{cardholder}$ reference, see **§3.1**) |
| Lc | Length of verification data |
| Data | Verification data |
| Le | Empty |

**Table 18 – MVP:VERIFY Command APDU**

**Response APDU**

| Field | Value |
|---|---|
| Data | Empty |
| SW1-SW2 | Status Bytes |

**Table 19 – MVP:VERIFY Response APDU**

## Status bytes

| Value | Meaning |
|---|---|
| ’63 C**x**’ | Verification failed, ‘**x**’ retries remaining |
| ’64 00’ | No precise diagnostic |
| ‘65 81’ | EEPROM corrupted (followed by a temporary mute state) |
| ’69 83’ | Authentication method blocked (PIN counter null) |
| ‘6A 86’ | Wrong parameter P1-P2 |
| ‘6A 88’ | Referenced PIN not found |
| ‘6D 00’ | Command not available within the current life cycle |
| ’6E 00’ / ’68 81’ | Wrong coding of class byte |
| ’90 00’ | Normal ending of the command |

**Table 20 – MVP:VERIFY Status bytes**

## 6.6 MVP:CHANGE REFERENCE DATA Command (ISO 7816-8)

### Description

This command is used to replace an existing PIN value with a new one.

The new value shall be presented with the same format, as it exists within the card.

When the user changes his PIN value, the previous access condition is discarded, the current PIN is presented and compared with the one stored in the EID card.
If the comparison fails, the PIN retry counter is decreased and the PIN value is not changed.
If the verification is successful, the PIN value is modified with the new PIN value and the associated access right granted.

This command is usually defined as a “PIN updating” procedure.

### Conditions of Use

The PIN to verify must not be blocked.

### Command structure

| Instruction | CLA | INS | P1 | P2 | Lc | Le | Case |
|---|---|---|---|---|---|---|---|
| MVP:CHANGE REFERENCE DATA | ‘00’ | ‘24’ | ‘00’ | ‘01’ | ’10’ | - | 3 |

**Command APDU**

| Field | Value |
|---|---|
| CLA | ‘00’ |
| INS | ‘24’ |
| P1 | ‘00’ |
| P2 | ‘01’ (PIN$_{\text{cardholder}}$ reference, see **§3.1**) |
| Lc | Length of subsequent data field |
| Data | Existing PIN **\|\|** New PIN |
| Le | Empty |

**Table 21 – MVP:CHANGE REFERENCE DATA Command APDU**

**Response APDU**

| Field | Value |
|---|---|
| Data | Empty |
| SW1-SW2 | Status Bytes |

**Table 22 – MVP:CHANGE REFERENCE DATA Response APDU**

## Status bytes

| Value | Meaning |
|---|---|
| ’63 CX’ | Verification failed, ‘***X***’ retries remaining |
| ’64 00’ | No precise diagnostic |
| ‘65 81’ | EEPROM corrupted (followed by a temporary mute state) |
| ’67 00’ | Wrong length |
| ’69 83’ | Authentication method blocked (PIN$_{\text{cardholder}}$ blocked) |
| ‘6A 80’ | Incorrect parameter in data field (e.g. wrong PIN$_{\text{cardholder}}$ format) |
| ‘6A 86’ | Wrong parameter P1-P2 |
| ‘6A 88’ | Referenced data not found (PIN$_{\text{cardholder}}$ not found) |
| ‘6D 00’ | Command not available within the current life cycle |
| ’6E 00’ / ’68 81’ | Wrong coding of class byte |
| ’90 00’ | Normal ending of the command |

**Table 23 – MVP:CHANGE REFERENCE DATA Status bytes**

## 6.7 MSE:SET Command (ISO 7816-4)

### Description

The MSE: SET command is used to set attributes in the Belpic Security Environment (SE):

- Selecting the algorithm used to perform a signature via the algorithm reference (see Table 11 - Algorithm references)
- Selecting the RSA Private Key (by using a key reference inside the Digital Signature Template) that shall be used in the digital signature creation process (see signature key references in **§4.1**).

**See the available signature schemes** in Table 27 – Signature Schemes.

### Conditions of Use

No security conditions are required to perform this command.

### Command structure

| Instruction | CLA | INS | P1 | P2 | Lc | Le | Case |
|---|---|---|---|---|---|---|---|
| MSE:SET | '00' | '22' | '41' | 'B6' | '05' | - | 3 |

**Command ADPU**

| Field | Value |
|---|---|
| CLA | '00' |
| INS | '22': Manage Security Environment |
| P1 | '41': Set the signature mode |
| P2 | 'B6': Value of the DST in data field |
| Lc | Length of subsequent data field |
| Data | Length of following data ='04' **\|\|**<br>Tag for Algorithm reference ='80' **\|\|**<br>**Algorithm reference** (refer to Table 11) **\|\|**<br>Tag for private key reference ='84' **\|\|**<br>**Private key reference** = either '82' or '83', refer to **§4.1** |
| Le | Empty |

**Table 24 – MSE:SET command APDU**

**Response ADPU**

| Field | Value |
|---|---|
| Data | Empty |
| SW1-SW2 | Status Bytes |

**Table 25 – MSE:SET Response APDU**

### Status bytes

| Value | Meaning |
|---|---|
| ’64 00’ | No precise diagnostic |
| ‘65 81’ | EEPROM corrupted (followed by a temporary mute state) |
| ’67 00’ | Wrong length |
| ’69 85’ | Condition of use not satisfied (Key not activated or algorithm locked by Lock Algorithm command) |
| ‘6A 80’ | Incorrect parameter in the data field (e.g. wrong tag, algorithm reference not supported, P2 doesn’t refer to a private key). |
| ‘6B 00’ | Wrong parameter P1-P2 |
| ‘6A 88’ | Referenced key not found (e.g. Authentication key, Non-repudiation key not found) |
| ‘6D 00’ | Command not available within the current life cycle |
| ’6E 00’ / ’68 81’ | Wrong coding of class byte |
| ’90 00’ | Normal ending of the command |

**Table 26 – MSE:SET Status bytes**

## 6.8 PSO:COMPUTE DIGITAL SIGNATURE command (ISO 7816-8)

### Description

The PSO:CDS command shall initiate the computation of a digital signature. The private key and the algorithm to be used shall be previously specified by a MSE: SET command.
Available “Signature keys” are (see §4.1):

- Authentication key
- Non-Repudiation key

**Key length supported:**

- The command supports only RSA 2048 key length.

**Algorithms used**:

- The following 6 signature schemes are supported:

| Signature Schemes (PKCS#1 v2.1) | | |
|---|---|---|
| **Scheme** | **RSA key length** | **Hash algorithm** |
| RSASSA-PKCS1_v15 | 2048 | Not specified [1] |
| | | MD5 |
| | | SHA1 |
| | | SHA256 |
| RSASSA-PSS | 2048 | SHA1 |
| | | SHA256 |

**Table 27 – Signature Schemes**

[1] For the signature scheme RSASSA-PKCS1-v15 it is possible to leave the hash algorithm unspecified (“Not specified”) to support any hash algorithm but the applet will not be able to check the length of the hash to be signed.

## Conditions of Use

The Access Condition of the referenced key related to this command must be fulfilled prior using the command (refer to ***Table 5*** and Table 6 for keys access conditions):

- The ***PIN*** $_{cardholder}$ access right must have been granted at any time before using the signature authentication key.
- Each time the signature non-repudiation key is used, a successful ***MVP: Verify (PIN*** $_{cardholder}$***)*** APDU command has to be executed just before performing this command.

## Command structure

| Instruction | CLA | INS | P1 | P2 | Lc | Le | Case |
|---|---|---|---|---|---|---|---|
| PSO:CDS | '00' | '2A' | '9E' | '9A' | Length | Length | 4 |

### Command APDU

| Field | Value |
|---|---|
| CLA | '00' |
| INS | '2A': Perform Security Operation |
| P1 | '9E': PSO:CDS |
| P2 | '9A' (Data field contains data to be signed) |
| Lc | Length of the data to be signed |
| Data | Data to be signed (not padded) |
| Le | Length of the signature :<br>- '00' (256 bytes) for RSA 2048 |

**Table 28 – PSO:CDS command APDU**

**Note:** When PKCS#1 is used, the card will pad the data to be signed (DTBS) according to PKCS#1 version 2.1.

- RSA PKCS#1 algorithm selected (hashing format not specified):
  The length of the DTBS must be inferior or equal to 245 bytes (RSA 2048)

- RSA MD5 PKCS#1 algorithm selected:
  The length of the DTBS must be equal to 16 bytes.

- RSA SHA-1 PKCS#1 algorithm selected:
  The length of the DTBS must be equal to 20 bytes.

- RSA SHA-256 PKCS#1 algorithm selected:
  The length of the DTBS must be equal to 32 bytes.

### Response ADPU

| Field | Value |
|---|---|
| Data | Signature |
| SW1-SW2 | Status Bytes |

**Table 29 – PSO:CDS Response APDU**

## Status bytes

| Value | Meaning |
|---|---|
| ’64 00’ | No precise diagnostic |
| ‘65 81’ | EEPROM corrupted (followed by a mute state) |
| ’67 00’ | Wrong length |
| ’69 82’ | Security status not satisfied (e.g. PIN access right not granted) |
| ’69 85’ | Condition of use not satisfied (e.g. security environment not set or algorithm locked by Lock Algorithm command) |
| ‘6B 00’ | Wrong parameter P1-P2 |
| ‘6D 00’ | Command not available within the current life cycle |
| ‘61 xy’ | Process completed normally (‘xy’ encoded the number of extra data bytes still available). The card expects a GET RESPONSE command with Le=’xy’ |
| ’6E 00’ / ’68 81’ | Wrong coding of class byte |
| ’90 00’ | Normal ending of the command |

**Table 30 – PSO:CDS Status bytes**

## 6.9 GET CARD DATA Command

### Description

This command is used to retrieve some useful information about the card.

The content of the response gives information about chip, OS and applet (version for instance).

### Conditions of Use

No security conditions are required to perform this command.

### Command structure

| Instruction | CLA | INS | P1 | P2 | Lc | Le | Case |
|---|---|---|---|---|---|---|---|
| GET CARD DATA | ‘80’ | ‘E4’ | ‘00’ | ‘00’ | - | Length | 2 |

**Command ADPU**

| Field | Value |
|---|---|
| CLA | ‘80’ |
| INS | ‘E4’ |
| P1 | ‘00’ |
| P2 | ‘00’ |
| Lc | Empty |
| Data | Empty |
| Le | Length of the response = ‘1C’ |

**Table 31 - GET CARD DATA Command APDU**

**Response APDU**

| Response fields | Length (byte) | Description / Value |
|---|---|---|
| Serial Number | 16 | The serial number is composed of 2 bytes reserved for Gemalto, 2 bytes identifying the chip manufacturer, and 12 bytes identifying uniquely the chip among all chips from this manufacturer |
| Component code | 1 | Chip dependent |
| OS number | 1 | ‘xx’ (platform specific) |
| OS version | 1 | ‘xx’ (platform specific) |
| Softmask number | 1 | ‘xx’ (platform specific) |
| Softmask version | 1 | ‘xx’ (platform specific) |
| Applet version | 1 | ‘17’ = Applet version 1.7 |
| Global OS Version | 2 | ‘0003’ = Belpic V1.7 |
| Applet interface version | 1 | ‘00’ |
| PKCS#1 support | 1 | ‘21’ = PKCS#1 version 2.1 |
| Key Exchange version | 1 | ‘01’ |
| Applet Life cycle | 1 | ‘07’ = SELECTABLE state<br>‘0F’ = PERSONALIZED state |

**Table 32 – GET CARD DATA Detailed Response**

## Status bytes

| Value | Meaning |
|---|---|
| ’64 00’ | No precise diagnostic |
| ‘65 81’ | EEPROM corrupted (followed by a mute state) |
| ‘6C’ XX | ***Le*** incorrect, ***XX*** indicates the expected length (hexadecimal value) |
| ‘6B 00’ | Wrong parameter P1-P2 |
| ’6E 00’ / ’68 81’ | Wrong coding of class byte |
| ’90 00’ | Normal ending of the command |

**Table 33 – GET CARD DATA Status bytes**

## 6.10 LOG OFF Command

### Description

This command shall be used to discard the current fulfilled access condition.

### Conditions of Use

No security conditions are required to perform this command.

### Command structure

| Instruction | CLA | INS | P1 | P2 | Lc | Le | Case |
|---|---|---|---|---|---|---|---|
| LOG OFF | ‘80’ | ‘E6’ | ‘00’ | ‘00’ | - | - | 1 |

**Command ADPU**

| Field | Value |
|---|---|
| CLA | ‘80’ |
| INS | ‘E6’ |
| P1 | ‘00’ |
| P2 | ‘00’ |
| Lc | Empty |
| Data | Empty |
| Le | Empty |

**Table 34 - LOG OFF Command APDU**

**Response APDU**

| Field | Value |
|---|---|
| Data | Empty |
| SW1-SW2 | Status Bytes |

**Table 35 – LOG OFF Response APDU**

### Status bytes

| Value | Meaning |
|---|---|
| ’64 00’ | No precise diagnostic |
| ‘65 81’ | EEPROM corrupted (followed by a mute state) |
| ’67 00’ | Wrong length |
| ‘6B 00’ | Wrong parameter P1-P2 |
| ‘6D 00’ | Command not available within the current life cycle |
| ’6E 00’ / ’68 81’ | Wrong coding of class byte |
| ’90 00’ | Normal ending of the command |

**Table 36 – LOG OFF Status bytes**

**- END OF DOCUMENT -**