



# **Reference Manual**

## **Belpic V1.8 eID**

## Table of content

<b>Reference Manual.....</b>	<b>1</b>
<b>Belpic V1.8 eID.....</b>	<b>1</b>
Table of content.....	2
Table of figures.....	5
<b>1 Introduction.....</b>	<b>6</b>
1.1 Purpose.....	6
1.2 Who should read this manual.....	6
1.3 References.....	6
1.4 Terminology.....	7
1.5 Acronyms and Abbreviations.....	7
1.6 Conventions.....	8
1.6.1 General.....	8
1.6.2 Symbols.....	8
1.7 Remarks on standards use.....	9
1.8 Applet version.....	9
1.9 Backward compatibility.....	9
1.9.1 Cryptographic algorithms, keys and protocols.....	9
<b>2 General description of the applet.....</b>	<b>11</b>
2.1 Properties of the Java Card applet.....	12
2.2 Applet AID.....	13
2.3 Applet Life Cycle.....	14
2.4 Security conditions.....	15
<b>3 PIN codes.....</b>	<b>16</b>
3.1 Description.....	16
3.2 Format.....	16
<b>4 Keys Description.....</b>	<b>17</b>
4.1 ECDSA Private Key.....	17
4.2 ECDSA Public Key.....	18
<b>5 Authentication processes.....</b>	<b>19</b>
5.1 PIN Verification.....	19
5.1.1 “PIN once” mode.....	20
5.1.2 “PIN just before” mode.....	21
5.1.3 “PIN Mixed” mode.....	22
5.2 Internal Authentication.....	23
5.3 User Authentication.....	24
<b>6 Command interface.....</b>	<b>25</b>
6.1 Protocol for T=0 (ISO 7816-3).....	26
Get Response Description.....	26
Command structure.....	26

Status bytes.....	26
<b>6.2 Logical Channels (ISO 7816-4).....</b>	<b>27</b>
6.2.1 Introduction.....	27
6.2.2 Description.....	27
6.2.3 Chaining Output Data.....	28
6.2.4 List of commands supporting chaining.....	29
<b>6.3 Coding of algorithm.....</b>	<b>30</b>
<b>6.4 Coding of data object references.....</b>	<b>30</b>
<b>6.5 Coding of the data object number.....</b>	<b>31</b>
<b>6.6 SELECT FILE Command (ISO 7816-4).....</b>	<b>31</b>
Description.....	31
Conditions of Use.....	31
Command structure.....	31
Status bytes.....	32
<b>6.7 READ BINARY Command (ISO 7816-4).....</b>	<b>33</b>
Description.....	33
Conditions of Use.....	33
Command structure.....	33
Status bytes.....	33
<b>6.8 READ RECORD Command.....</b>	<b>35</b>
Description.....	35
Conditions of Use.....	35
Command structure.....	35
Status bytes.....	36
<b>6.9 MVP: VERIFY Command (ISO 7816-4).....</b>	<b>38</b>
Description.....	38
Conditions of Use.....	38
Command structure.....	38
Status bytes.....	39
<b>6.10 MVP: CHANGE REFERENCE DATA Command (ISO 7816-8).....</b>	<b>40</b>
Description.....	40
Conditions of Use.....	40
Command structure.....	40
Status bytes.....	41
<b>6.11 INTERNAL AUTHENTICATE Command (ISO 7816-4).....</b>	<b>42</b>
Description.....	42
Conditions of Use.....	43
Response APDU.....	44
Status bytes.....	44
<b>6.12 MSE: SET Command (ISO 7816-4).....</b>	<b>45</b>
Description.....	45
Conditions of Use.....	45
Command structure.....	45
Status bytes.....	46
<b>6.13 PSO: COMPUTE DIGITAL SIGNATURE command (ISO 7816-8).....</b>	<b>47</b>
Description.....	47
Conditions of Use.....	48
Command structure.....	48
Status bytes.....	49
<b>6.14 GET CARD DATA Command.....</b>	<b>50</b>
Description.....	50
Conditions of Use.....	50
Command structure.....	50
Status bytes.....	52

<b>6.15 LOG OFF Command.....</b>	<b>53</b>
Description.....	53
Conditions of Use.....	53
Command structure.....	53
Status bytes.....	53

## Table of figures

Figure 1 – BelpIC electronic identity card structure.....	11
Figure 2 - Applet Life Cycle.....	14
Figure 3 - PERSONALIZED state.....	14
Figure 4 - PIN verification process.....	19
Figure 5 - Internal Authentication process using Internal Authenticate command with PrK#1(Basic) and ECDSA using SHA-2-256 algorithm.....	23
Figure 6 – Producing an authentication or non-repudiation signature.....	24

# 1 Introduction

## 1.1 Purpose

The purpose of this document is to define the functional specification of the BelPIC electronic identity card.

## 1.2 Who should read this manual

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#1 2.1	RSA Cryptography Standard, version 2.1, RSA laboratories, June 2002 <a href="ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf">ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf</a>
[ER9]	ISO CEI 9564-1	Banking – Personal Identification Number (PIN) management and security – 2002
[ER10]	NIST-FIPS-186-4, ECDSA	ECDSA as specified by NIST in FIPS 186-4, July 2013 <a href="http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf">http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf</a>
[ER11]	BSI Technical Guideline TR-03111, Elliptic Curve Cryptography V2.0	Elliptic curve cryptography, June 2012
[ER12]	NIST-FIPS-202 SHA-3	<a href="https://csrc.nist.gov/publications/detail/fips/202/final">https://csrc.nist.gov/publications/detail/fips/202/final</a>

## 1.4 Terminology

Nibble	Half part of bytes (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:

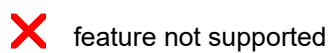
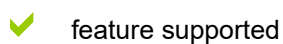
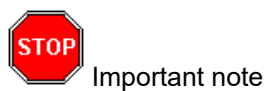
<b>AID</b>	Application provider identifier
<b>AODF</b>	Authentication object directory file
<b>APDU</b>	Application protocol data unit
<b>API</b>	Application programming interface
<b>BCD</b>	Binary-coded decimal
<b>CA</b>	Certification Authority
<b>CBC</b>	Cipher Block Chaining
<b>CC</b>	Cryptographic Checksum
<b>CG</b>	Cryptogram
<b>CDF</b>	Certificate directory file
<b>CLA</b>	Apu Class byte
<b>ISD</b>	Issuer Security Domain (Card Manager)
<b>DF</b>	Dedicated File
<b>DO</b>	Data Object
<b>DODF</b>	Data object directory file
<b>DST</b>	Digital Signature Template
<b>DTBS</b>	Data to be signed
<b>EID</b>	Electronic Identity Device
<b>EF</b>	Elementary file
<b>FCI</b>	File Control Information
<b>FCP</b>	File Control Parameter
<b>GP</b>	Global Platform (see <a href="#">[ER7]</a> )
<b>ICC</b>	Integrated Circuit(s) Card
<b>IFD</b>	Interface device (e.g. reader)
<b>JCVM</b>	Java card virtual machine
<b>MF</b>	Master file
<b>MSB</b>	Main significant byte
<b>ODF</b>	Object directory file
<b>OP</b>	Open Platform
<b>OS</b>	Operating system
<b>PIN</b>	Personal identification number
<b>PSO:CDS</b>	PSO: Compute Digital Signature
<b>PUK</b>	PIN Unblocking Key
<b>PuK</b>	public key
<b>PrK</b>	private key
<b>PrKDF</b>	Private key directory file
<b>PuKDF</b>	Public key directory file
<b>RID</b>	Registered application provider identifier
<b>SE</b>	Security Environment
<b>TLV</b>	Tag Length Value

## 1.6 Conventions

### 1.6.1 General

- All the values between quotations are in hexadecimal notation.
- The values are in MSB first.
- The symbol || represents a concatenation.
- The following syntax is used to describe the requirements:
  - “shall” when it is mandatory,
  - “should” when it is optional.

### 1.6.2 Symbols





## 1.7 Remarks on standards use

**ECDSA over a prime field with a modulus of at least 256 bits** is used (see [ER10] for recommended elliptic curve definitions) in combination with a secure hash function.

## 1.8 Applet version

This document describes the specification of applet V1.8.

Applet V1.8 is recognized by the “**applet version**” byte returned by the command “**Get Card Data**” (see §6.14) set to ‘18’.

All differences from version V1.7 are annotated with the following:

### **Compatibility note**

In applet V1.7 ...

## 1.9 Backward compatibility

Due to new security requirements and new Belpic use-cases, the Applet V1.8 is not fully backward compatible with the previous Applet V1.7.

The following chapters summarize the legacy Applet V1.7 features no more supported by the Applet V1.8.

### 1.9.1 Cryptographic algorithms, keys and protocols

To meet the security requirements up to 2030 and to reduce key lengths, the Applet V1.8 does not support anymore:

- deprecated algorithms and keys
- EID RSA signatures
- EID RSA private keys

legacy features	Applet V1.7	Applet V1.8
MD5 and SHA-1 hash algorithms	✓	✗
TDES secure messaging session keys	✓	✗
RSA encryption schemes (RSAES-PKCS1-v15 with SHA1 and RSAES-OAEP with SHA256)	✓	✗
RSA signature schemes (RSASSA-PKCS1_v15 using SHA1 or MD5 or SHA256, RSASSA-PSS using SHA1 or SHA256)	✓	✗
private RSA key (basic key ‘81’, authentication key ‘82’, non-repudiation key ‘83’)	✓	✗
<b>Internal authenticate</b> command with RSA signatures	✓	✗
<b>PSO:CDS</b> command with RSA signatures	✓	✗
Blocking the EID card after 5000 <b>Internal Authenticate</b>	✓	✗ (only Internal Authenticate is blocked)
Unblocking the EID card after 5000 <b>Internal Authenticate</b> with PIN <sub>activate</sub>	✓	✗

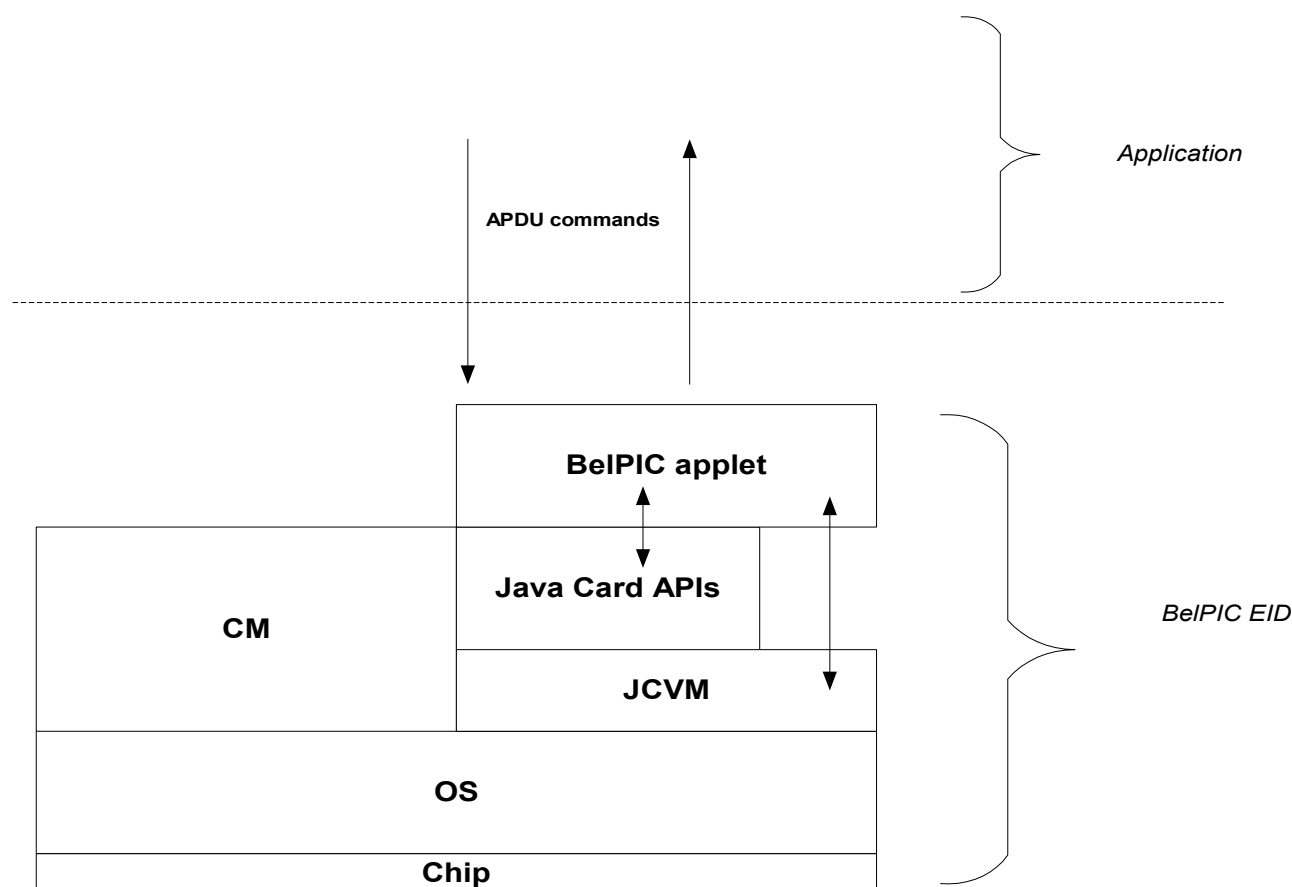
legacy features	Applet V1.7	Applet V1.8
		(PIN <sub>cardholder</sub> replaces PIN <sub>activate</sub> )

## 2 General description of the applet

The BelPIC application is composed of two applications:

- The electronic signature application
- The electronic identification application

Those two applications must be implemented through a Java Card Applet and will share the same security environment.



**Figure 1 – BelPIC electronic identity card structure**

The Issuer Security Domain (ISD) is responsible for loading and installing the BelPIC applet into the card. The BelPIC applet implements the file system associated to the BelPIC application and is installed with the default-selected privilege.

The MF is selected DF after a reset.

The BelPIC security environment must be implicitly set after a reset and must contain the PIN objects and key objects used by the application.

## 2.1 Properties of the Java Card applet

- It is not possible to extend the existing file system
- It is not possible to delete a file.
- The applet shall not process the PKCS#15 information contained by the BelPIC file system; this information shall only be managed and used by the external application (middleware).
- The cryptographic objects (PIN and Keys) shall not be stored into files but into secured Java containers managed by the operating system.
- Only the transparent elementary files are supported.
- ECDSA moduli supported are at least 256 bits long.
- In all signature algorithms, the hash computation is done off card.

## 2.2 Applet AID

Applet Instance AID	A000000003029057000AD13100101FF
---------------------	---------------------------------

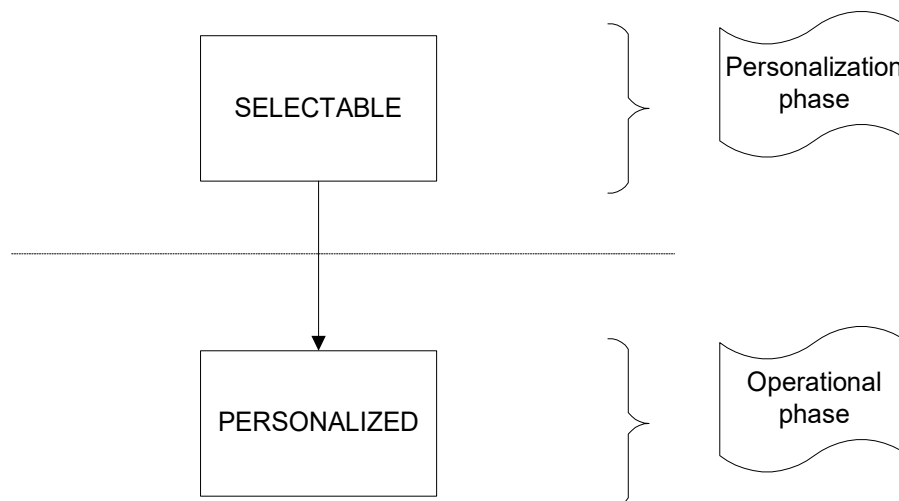
This AID can be used to select the BelPIC applet.



**Note:** After card reset, BelPIC applet is implicitly selected because the applet is installed with the default-selected privilege.

## 2.3 Applet Life Cycle

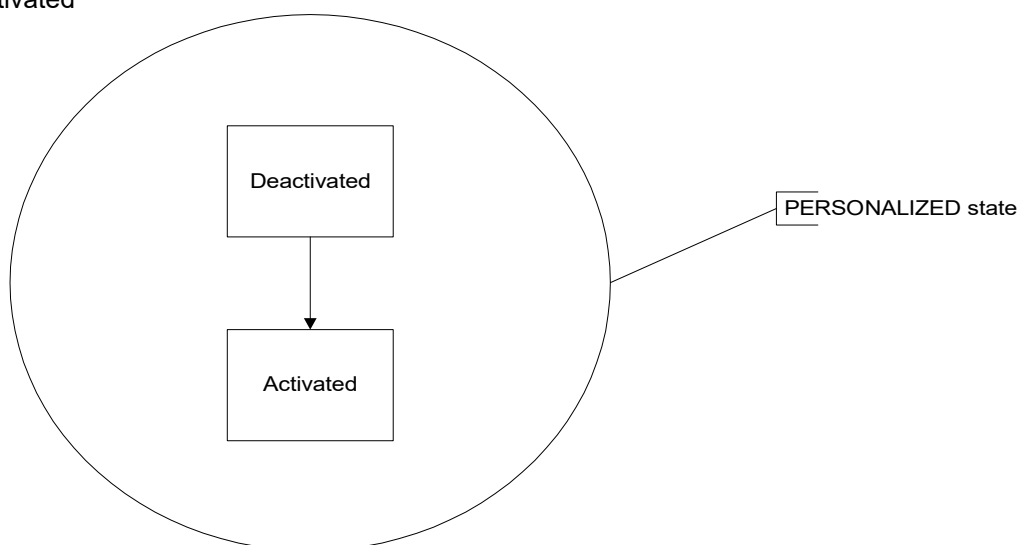
During the personalization phase, the BelpIC applet will be in the SELECTABLE state.  
At the end of the personalization, the Set Status command will put the Applet in the PERSONALIZED state.



**Figure 2 - Applet Life Cycle**

The PERSONALIZED state shall be made of two internal states:

- Deactivated (state after end of personalization)
- Activated



**Figure 3 - PERSONALIZED state**

**Remark:** The private keys can only be used when the card is activated; they cannot be used during the personalisation.

## 2.4 Security conditions

The following table lists the different security conditions of the EID card:

Security Conditions	Type 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).

## 3 PIN codes

### 3.1 Description

The applet shall contain the following PIN Objects:

PIN	Max. number of attempts	Corresponding unblocking mechanism	Application
PIN <sub>activate</sub>	15	No	<input type="checkbox"/> Activate the EID card
PIN <sub>cardholder</sub>	3	External Authenticate with Role <sub>Admin</sub> certificate	<input type="checkbox"/> Change PIN <sub>cardholder</sub> <input type="checkbox"/> Signature with authentication key <input type="checkbox"/> Non-repudiation signature with non-repudiation key <input type="checkbox"/> Unblock the <b>Internal Authenticate</b> mechanism after 5000 <b>Internal Authenticate</b> commands

Table 1 - BelpIC PIN



**New feature V1.8:** The PUK<sub>Unblock</sub> and the PIN<sub>reset</sub> have been replaced by the new administration role.

### 3.2 Format

The PIN is 8-bytes strings 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'	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 contains always 'F'

Table 2 - PIN format

PIN	Minimum number of digits
PIN <sub>cardholder</sub>	4

Table 3 - PIN length (in digits)



## 4 Keys Description

### 4.1 ECDSA Private Key

The modulus of each ECDSA key is at least 256 bits long.

The following ECDSA key length are supported according to **EID personalization**:

- **ECDSA 256 bit-long key**
- **ECDSA 384 bit-long key**
- **ECDSA 512 bit-long key**
- **ECDSA 521 bit-long key**

The 3 following private keys are present in the Belpic card and are generated on-card.

#### **Basic key**

This key is a private key with reference '**81**'.

This card key is used in **Internal Authenticate** command.

The key access conditions are the following (See access condition coding in chapter §2.4):

	<i><b>PSO:CDS</b></i>	<i><b>Internal Authenticate</b></i>
<b>Access Conditions</b>	NEV	ALW

**Table 4 – Basic key '81' access conditions**

#### **Authentication key**

This key is a private ECDSA 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 §2.4):

	<i><b>PSO:CDS</b></i>	<i><b>Internal Authenticate</b></i>
<b>Access Conditions</b>	CHV PIN <sub>cardholder</sub> «once» (see §5.1)	NEV

**Table 5 – Authentication key '82' access conditions**

#### **Non-Repudiation key**

This key is a private ECDSA 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 §2.4):

	<i><b>PSO:CDS</b></i>	<i><b>Internal Authenticate</b></i>
<b>Access Conditions</b>	CHV PIN <sub>cardholder</sub> «just before», (see §5.1)	NEV

**Table 6 – Non-Repudiation key '83' access conditions**



## 4.2 ECDSA Public Key

The modulus of each ECDSA key is at least 256 bits long.

All public ECDSA keys generated in the card match their corresponding private keys.

## 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:

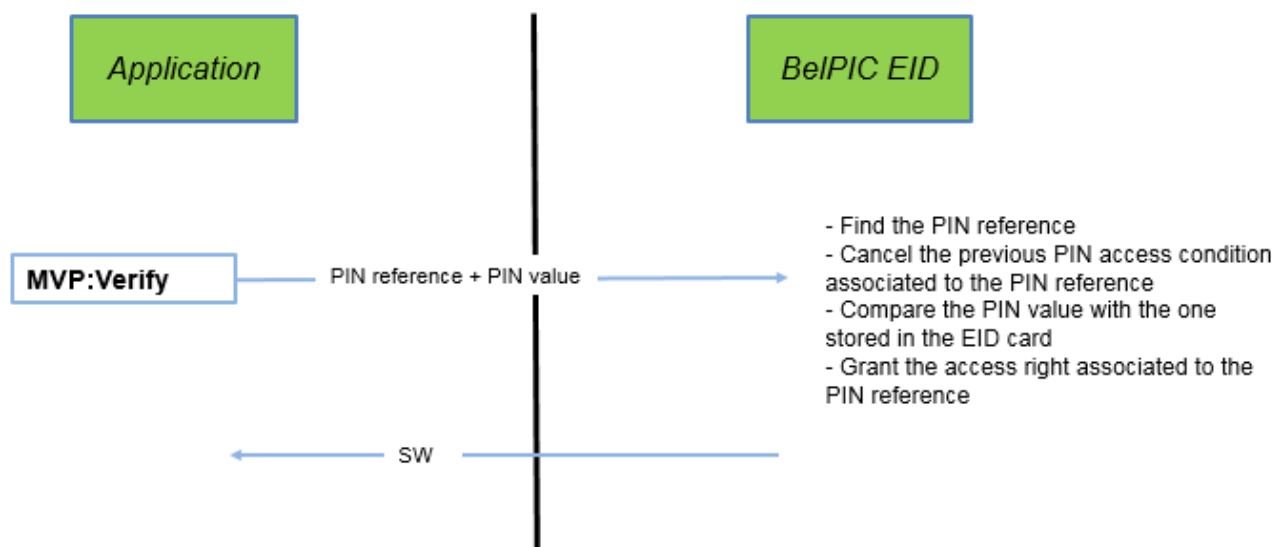


Figure 4 - PIN verification process

### 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 the signature generation command
- ❑ "PIN just before": the PIN access right must be granted at most 1 command before calling the signature generation command.



A wrong PIN verification resets PIN access conditions for ALL PIN-protected commands.

## User PIN and Signature keys

The PIN<sub>authenticate</sub> or PIN<sub>nonrepudiation</sub> shall be verified before signing with the authentication and non-repudiation, respectively.



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

- A **Verify(PIN<sub>nonRepudiation</sub>)** is a Verify PIN for non-repudiation signature sequence.

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

Examples of **Verify(PIN<sub>nonRepudiation</sub>)** for non-repudiation signature sequence:

- ❑ Reset, ..., MSE(Key<sub>non-rep</sub>), **Verify(PIN<sub>nonRepudiation</sub>)**, Sign(Key<sub>non-rep</sub>)
- ❑ Reset, ..., MSE(Key<sub>non-rep</sub>), ..., <sup>1</sup> **Verify(PIN<sub>nonRepudiation</sub>)**, Sign(Key<sub>non-rep</sub>)
- ❑ Reset, ..., MSE(Key<sub>non-rep</sub>), ..., <sup>2</sup> **Verify(PIN<sub>nonRepudiation</sub>)**, one command, Sign(Key<sub>non-rep</sub>)

- A **Verify(PIN<sub>authentication</sub>)** 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<sub>authentication</sub>)** for authentication sequence:

- ❑ Reset, ..., **Verify(PIN<sub>authentication</sub>)**, ..., MSE(Key<sub>auth</sub>), ..., Sign(Key<sub>auth</sub>)
- ❑ Reset, ..., MSE(Key<sub>auth</sub>), ..., **Verify(PIN<sub>authentication</sub>)**, ..., Sign(Key<sub>auth</sub>)
- ❑ Reset, ..., **Verify(PIN<sub>authentication</sub>)**, ...
- ❑ Reset, ..., **Verify(PIN<sub>authentication</sub>)**, ..., Read, ...

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

### 5.1.1 “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.

<sup>1</sup> It is assumed that “...” means any Belpic commands except MSE, Verify, Sign (and commands resetting PIN or SE, like External Authenticate,...)

<sup>2</sup> It is assumed that “...” means any Belpic commands except MSE, Verify, Sign (and commands resetting PIN or SE, like External Authenticate,...)

### Examples of required sequences:

- ❑ **Verify(PIN<sub>authentication</sub>)**, Sign(Key<sub>auth</sub>), ..., <sup>1</sup> Sign(Key<sub>auth</sub>)

### Examples of incorrect sequences:

- No previous Verify(PIN)

- ❑ Sign(Key<sub>auth</sub>) KO

## 5.1.2 "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, or with at most 1 command between the **MVP: Verify(PIN)** and the signature command.

If the number of commands executed between the **MVP: Verify(PIN)** and the **PSO: CDS** commands exceeds 1, the PIN verification status must be reset, and it will not be possible to produce a 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<sub>nonRepudiation</sub>)**, Sign(Key<sub>non-rep</sub>), ..., **Verify(PIN<sub>nonRepudiation</sub>)**, Sign(Key<sub>non-rep</sub>)
- ❑ **Verify(PIN<sub>nonRepudiation</sub>)**, one command<sup>2</sup>, Sign(Key<sub>non-rep</sub>), ..., **Verify(PIN<sub>nonRepudiation</sub>)**, Sign(Key<sub>non-rep</sub>)

### Examples of incorrect sequences:

- No previous Verify(PIN) just before the signature

- ❑ **Verify(PIN<sub>authentication</sub>)**, ..., Sign(Key<sub>non-rep</sub>) KO
- ❑ **Verify(PIN<sub>authentication</sub>)**, ..., Sign(Key<sub>auth</sub>), Sign(Key<sub>non-rep</sub>) KO
- ❑ **Verify(PIN<sub>nonRepudiation</sub>)**, Sign(Key<sub>non-rep</sub>), Sign(Key<sub>non-rep</sub>) KO
- ❑ Sign(Key<sub>non-rep</sub>) KO
- ❑ **Verify(PIN<sub>nonRepudiation</sub>)**, Read, Read, Sign(Key<sub>non-rep</sub>) KO

<sup>1</sup> it is assumed that " , ..., " or "one command" means any Belpic commands except **Verify**, **PSO:CDS** and commands resetting PIN or SE (logout, ...)

<sup>2</sup> it is assumed that " , ..., " means any Belpic commands except **Verify**, **PSO:CDS** (and commands resetting PIN or SE)

### 5.1.3 "PIN Mixed" mode



#### 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<sub>authentication</sub>)**, ..., **Verify(PIN<sub>nonRepudiation</sub>)**, Sign(Key<sub>non-rep</sub>), ..., Sign(Key<sub>auth</sub>)
- ❑ **Verify(PIN<sub>authentication</sub>)**, Sign(Key<sub>auth</sub>), **Verify(PIN<sub>nonRepudiation</sub>)**, Sign(Key<sub>non-rep</sub>), ..., Sign(Key<sub>auth</sub>)
- ❑ **Verify(PIN<sub>authentication</sub>)**, ..., Sign(Key<sub>auth</sub>), **Verify(PIN<sub>nonRepudiation</sub>)**, Sign(Key<sub>non-rep</sub>), ..., Sign(Key<sub>auth</sub>), ..., Sign(Key<sub>auth</sub>), **Verify(PIN<sub>nonRepudiation</sub>)**, Sign(Key<sub>non-rep</sub>)
- ❑ **Verify(PIN<sub>nonRepudiation</sub>)**, Sign(Key<sub>non-rep</sub>), ... **Verify(PIN<sub>authentication</sub>)**, ..., Sign(Key<sub>auth</sub>)

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

- ❑ **Verify(PIN<sub>authentication</sub>)**, ..., **Verify(PIN<sub>nonRepudiation</sub>)**, Sign(Key<sub>non-rep</sub>), ..., Sign(Key<sub>auth</sub>), Read

#### Examples of incorrect sequences:

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

- ❑ **Verify(PIN<sub>nonRepudiation</sub>)**, Sign(Key<sub>non-rep</sub>), ..., Sign(Key<sub>auth</sub>) KO
- ❑ **Verify(PIN<sub>nonRepudiation</sub>)**, Wrong Sign(Key<sub>non-rep</sub>) KO, ..., Sign(Key<sub>auth</sub>) KO<sup>1</sup>
- ❑ **Verify(PIN<sub>authentication</sub>)**, ..., Wrong **Verify(PIN<sub>nonRepudiation</sub>)** KO, Sign(Key<sub>non-rep</sub>) KO, ..., Sign(Key<sub>auth</sub>) KO<sup>2</sup>

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

- ❑ **Verify(PIN<sub>nonRepudiation</sub>)**, Sign(Key<sub>non-rep</sub>), ..., Sign(Key<sub>auth</sub>) KO, Read KO
- ❑ **Verify(PIN<sub>nonRepudiation</sub>)**, Sign(Key<sub>non-rep</sub>), Read KO
- ❑ **Verify(PIN<sub>authentication</sub>)**, Read, Sign(Key<sub>non-rep</sub>) KO..., Sign(Key<sub>auth</sub>) OK

<sup>1</sup> Verify (PIN) only for non-repudiation signature because the signature is just after the Verify.

<sup>2</sup> A wrong PIN verification resets PIN access conditions for ALL PIN-protected commands.

## 5.2 Internal Authentication

The internal authentication process is used by the external application to authenticate the BelPIC EID card. The external application reads the public key associated to the basic key from the EF(PuK#1) and checks the hash of this file using the reference hash value included in the EF(ID#RRN).



**New feature V1.8:** The **Internal Authenticate** only supports ECDSA.

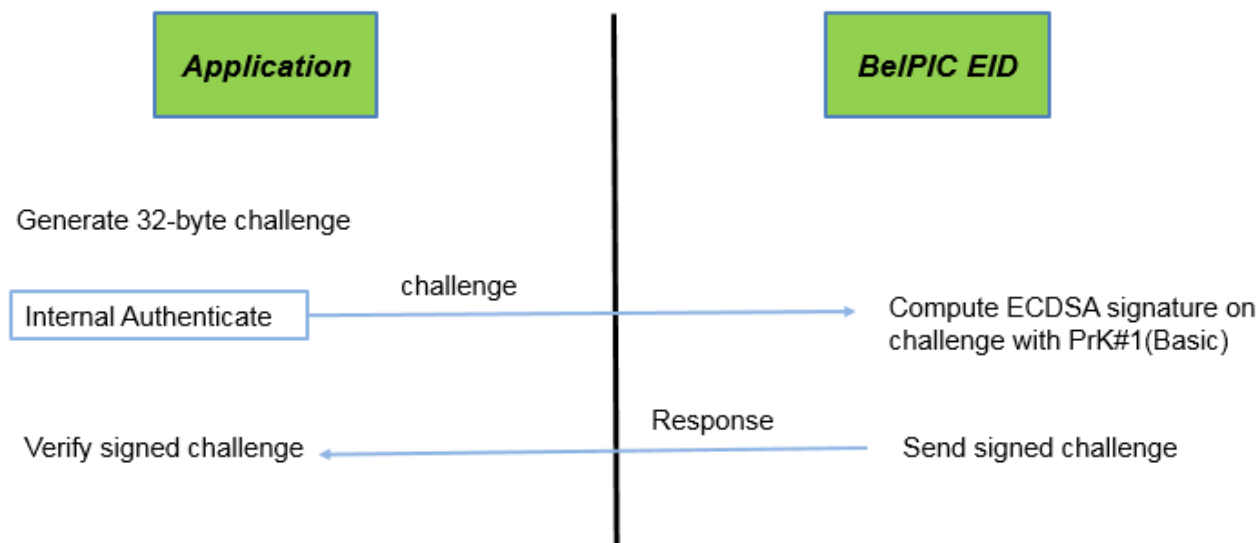


Refer to INTERNAL AUTHENTICATE Command (ISO 7816-4) for algorithms, key type and key length supported.



For security reasons, the **Internal Authenticate** command is limited to 5000 calls. After that, **the Internal Authenticate** command is blocked until a successful **Verify Pin** has been presented with **PIN<sub>cardholder</sub>** or until after a successful **External Authenticate** with the **role<sub>Activate</sub>** to activate the EID card.

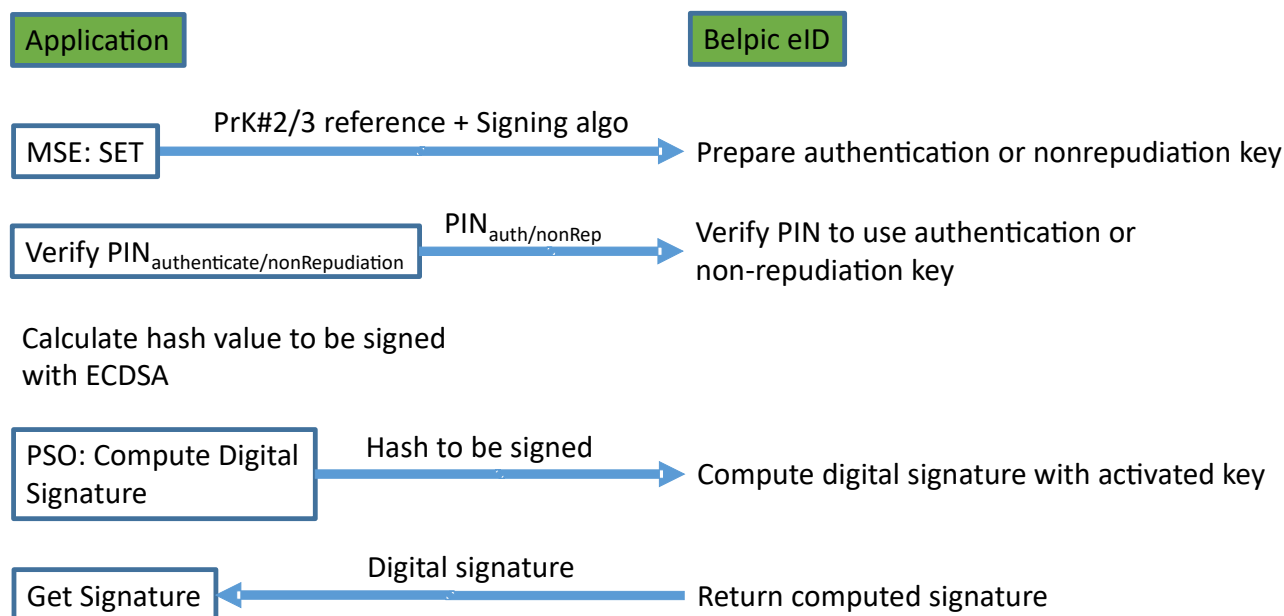
The internal authentication process uses the **Internal Authenticate (ISO 7816-4)** APDU command:



**Figure 5 - Internal Authentication process using Internal Authenticate command with PrK#1(Basic) and ECDSA using SHA-2-256 algorithm**

### 5.3 User Authentication

The user authentication is the process whereby the external application authenticates the cardholder. The algorithm must use the **ECDSA** format (see [ER10]) to disable any attack based on a malformed message.



**Figure 6 – Producing an authentication or non-repudiation signature**



## 6 Command interface

During the operational phase, the EID card shall offer 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
READ RECORD(s)	'80'	'B2'	First Data Ref.	Last Data Ref.	-	-	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 <sub>cardholder</sub>    New PIN <sub>cardholder</sub>	-	-
MVP: RESET RETRY COUNTER	'00'	'2C'	'00'	Data Ref.	-	-	-	-
INTERNAL AUTHENTICATE	'00'	'88'	Algo Ref	Data Ref.	Length	Challenge to sign	Length	Signature
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 Protocol for T=0 (ISO 7816-3)

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**.

### 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 APDU

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 APDU

### Status bytes

Value	Meaning
‘61 xx’	<b>xx</b> remaining bytes to retrieve from the card
‘6C xx’ <sup>1</sup>	<b>Le</b> too long, only <b>xx</b> bytes available (hexadecimal value)
‘6D 00’	Command not available within the current life cycle
‘6E 00’	CLA not supported
‘69 85’	There is no data to retrieve
‘90 00’	Normal ending of the command

Table 10 – GET RESPONSE Status bytes

<sup>1</sup> After a Status Word ‘6C XX’, if Get Response Le is different from ‘XX’, the Get Response chaining mechanism is not aborted

## 6.2 Logical Channels (ISO 7816-4)

The introduction of logical channels does not change the application behavior and is transparent for the end user.

This paragraph is informational since Belpic application is only selectable on logical channel 0.

### 6.2.1 Introduction

The Gemalto smart cards are compliant with Java Card 2.2.1 standard and therefore support the logical channel mechanism.

- **The Belpic application is only selectable on logical channel 0** because:
  - Belpic is the default selectable application, always selected on logical channel 0 after a card reset.
  - Belpic is not a multi-selectable application.
  - Belpic application rejects its selection on logical channel different from 0.
- **Trying to select Belpic application on a logical channel different from 0 is forbidden.**
- **Trying to open a new logical channel with Manage Channel command is forbidden.**

### 6.2.2 Description

The logical channels mechanism is described in ISO7816-4 [ER2] and in Java Card 2.2.1 Runtime Environment specification.

The logical channel mechanism allows the card to maintain up to 4 applets active in the same time. Only one applet is selected at a time and the selected applet is the one, which processes the current APDU. However, depending on the origin channel information of the CLA byte, an APDU may be forwarded to one of the 4 active applets.

Channel 0 is also called 'basic channel'. It is automatically open by the system and cannot be closed.



***Belpic commands are only available on basic logical channel 0.***



#### **Compatibility note**

Applet V1.7: A wrong Class byte returns '6E 00' or '68 81'

### 6.2.3 Chaining Output Data

When the length of the data returned by the card is too long for a single response data field (i.e. 256 bytes), the output command chaining has to be used.

#### Case of command case 4 (input and output data) returning more than 256 bytes:

- The data retrieval is performed by using the command GET RESPONSE. The IFD shall recover the data that way:

Incoming data	Outgoing data	Status Word returned
Incoming command CLA INS P1 P2 Lc Data (case 4)	N/A	SW = '61 00' '00' means 256 bytes are at least available
Incoming command GET RESPONSE with Le='00'	Return 256 bytes	If there are still remaining data SW = '61 Lxx' 'Lxx ' indicates the length of data to retrieve 'Lxx ' = '00' means 256 bytes are available If all the data were retrieved SW = '9000'
Incoming command GET RESPONSE with Le='xx'	Return xx bytes	Idem as above
Carry on the sequence of GET RESPONSE command until the SW '9000' is returned		

Table 11 - Output command case 4 chaining

#### Case of command case 2 (output data) returning more than 256 bytes:

- The data retrieval is performed by using the command GET RESPONSE. The IFD shall recover the data that way:

Incoming data	Outgoing data	Status Word returned
Incoming command CLA INS P1 P2 Le (case 2) with Le='00'	Return 256 bytes	SW = '61 Lxx' 'Lxx ' indicates the length of data to retrieve 'Lxx ' = '00' means 256 bytes are available
Incoming command GET RESPONSE Lxx	Return Lxx bytes	If there are still remaining data SW = '61 Lxx' 'Lxx ' indicates the length of data to retrieve 'Lxx ' = '00' means 256 bytes are available If all the data were retrieved SW = '9000'
Carry on the sequence of GET RESPONSE command until the SW '9000' is returned		

Table 12 - Output command case 2 chaining

## 6.2.4 List of commands supporting chaining

The following tab summarizes the list of commands supporting chaining in input or response:

Commands	Input Data Chaining	Output Data Chaining
Internal Authenticate	✗	256 bytes max
Get key	NA	✓
Read Record	NA	✓

Table 13 – Commands supporting chaining

**Caption:**

- ✓ chaining supported
- ✗ chaining not supported
- NA chaining not applicable, i.e. not needed for the command

## 6.3 Coding of algorithm

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

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Meaning
0	0	0	0	0	0	0	1	ECDSA SHA-2-256 (no predefined padding)
0	0	0	0	0	0	1	0	ECDSA SHA-2-384 (no predefined padding)
0	0	0	0	0	1	0	0	ECDSA SHA-2-512 (no predefined padding)
0	0	0	0	1	0	0	0	ECDSA SHA-3-256 (no predefined padding)
0	0	0	1	0	0	0	0	ECDSA SHA-3-384 (no predefined padding)
0	0	1	0	0	0	0	0	ECDSA SHA-3-512 (no predefined padding)
0	1	0	0	0	0	0	0	ECDSA with 'hash not specified' <sup>1</sup>
X	0	0	0	0	0	0	0	00 (others are RFU)

Table 14 – ECDSA Algorithm references

### Compatibility note

This has changed with respect to previous applets: this applet supports only ECDSA.

## 6.4 Coding of data object references

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

b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>	Meaning
0	0	0	0	0	0	0	0	No information is given
0	-	-	-	-	-	-	-	Global reference data
1	-	-	-	-	-	-	-	Specific reference data
-	X	X	-	-	-	-	-	'00' others are RFU
-	-	-	X	X	X	X	X	Data object number

Table 15 - Data object references (ISO 7816)

<sup>1</sup> ECDSA using hash with OID-specified externally.

## 6.5 Coding of the data object number

The data objects are divided into two categories

- Key objects
- PIN objects

Number	Meaning
'81'	Reference Prk#1 (Private Basic key)
'82'	Reference Prk#2 (Private authentication key)
'83'	Reference Prk#3 (Private non-repudiation key)
'87'	Reference Puk#7 (Public role key)

Table 16 - Key object numbers

Number	Meaning
'01'	Reference the PIN <sub>cardholder</sub>

Table 17 - PIN object numbers

## 6.6 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)



**Notes:** Select Belpic applet by AID on a logical channel different from the basic one is forbidden.

### 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) 2. '08' (the data field shall contain an absolute path) 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) 2. Absolute path without the identifier of the MF 3. Full AID (between 5 and 16 bytes) <sup>1</sup>
Le	Empty

**Table 18 – SELECT FILE Command APDU**

### **Response APDU**

Field	Value
Data	Empty
SW1-SW2	Status Bytes

**Table 19 – SELECT FILE Response APDU**

### **Status bytes**

Value	Meaning
'64 00'	No precise diagnostic
'65 81'	EEPROM corrupted (followed by a temporary mute state)
'69 85'	Condition of use not satisfied (File not activated)
'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 20 – SELECT FILE Status bytes**

<sup>1</sup> A Select with an empty AID is an implicit selection of the Card Manager. The Belpic applet can be re-selected explicitly (Select by instance AID) or implicitly (card reset).



## 6.7 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 current 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 21 – 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 22 – 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	<b>Le</b> incorrect, <b>XX</b> 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

Value	Meaning
'90 00'	Normal ending of the command

**Table 23 – READ BINARY Status bytes**

## 6.8 READ RECORD Command

### Description

This command shall be used to read the content of certain fields of a structured EF, e.g., EF(ID#RRN) and EF(Address#RRN).



The structured EF shall be previously updated by UPDATE BINARY commands with a consistent TLV-records structure.



The structured EF can also be read by the READ BINARY command.



The structured EF records are encoded in a simplified TLV format:

- A one byte tag field identifying the data
- A one byte length<sup>1</sup> field encoding the data length (from 0 to 255 bytes)
- A value field encoding the data

The TLV tags are always in ascending order ('00', '01', '02',...).

### Conditions of Use

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

The security conditions to fulfil before performing this command is the same as for the READ BINARY command.

The structured EF to be read shall be previously updated with a consistent TLV-records structure with several UPDATE BINARY commands.

### Command structure

Instruction	CLA	INS	P1	P2	Lc	Le	Case
READ RECORD	'80'	'B2'	First Data Ref.	Last Data Ref.	-	Length	2

### Command APDU

Field	Value
CLA	'80'
INS	'B2'
P1	First data Reference: field number of the first field to be read (first TLV Tag)
P2	Last data reference: field number of the last field to be read (last TLV Tag)
Lc	Empty
Data	Empty
Le	Length of the data to read

Table 24 – READ RECORD Command APDU

<sup>1</sup> Nor the tag, nor the length are counted in this length.



**Note:** if  $L_e$  is equal to 0, then it is interpreted as 256 bytes



**Note:** If  $L_e$  is equal to 0, then the command should read completely either the single requested record, or the requested sequence of records, depending on P1 and P2 and with response chaining mechanism if response length is greater than 256 bytes (see §6.2.3).

## Response APDU

Field	Value
Data	Read data. If first field != last field: Tag of first field    length    value of first field    ...    Tag of last field    length    value of last field.  If first field equals the last field: Tag of single field    length    value of single field.
SW1-SW2	Status Bytes

**Table 25 – READ RECORD Response APDU**

## Response APDU details

Case a — Partial read of one record (the  $L_e$  field is not set to '00' and  $L_e < \text{record size}$ )

$T_n$ (one byte)	$L_n$ (one byte)	First bytes of $V_n$
------------------	------------------	----------------------

Case b — Complete read of one record (the  $L_e$  field is set to '00')

$T_n$ (one byte)	$L_n$ (one byte)	All the bytes of $V_n$
------------------	------------------	------------------------

Case c — Partial read of a sequence of records (the  $L_e$  field is not set to '00')

$T_n - L_n - V_n$	...	$T_{n+m} - L_{n+m} - V_{n+m}$ (First bytes of the record)
-------------------	-----	---

Case d — Read several records up to the file end (the  $L_e$  field is set to '00')

$T_n - L_n - V_n$	...	$T_{n+m} - L_{n+m} - V_{n+m}$
-------------------	-----	-------------------------------

## Status bytes

Value	Meaning
'64 00'	No precise diagnostic
'65 81'	EEPROM corrupted (followed by a temporary mute state)
'69 81'	Command incompatible with the file structure (read record a non-structured EF)
'69 82'	Security status not satisfied
'69 85'	Condition of use not satisfied (File not activated)
'69 86'	Command not allowed (no current EF)
'6A 83'	Record not found (Tag not found, $P2 < P1$ )
'6C' XX	$L_e$ incorrect ( $L_e > \text{record size}$ ), <b>XX</b> indicates the expected length (hexadecimal value)
'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 solicits a GET RESPONSE command with $L_e = 'xy'$
'6E 00' / '68 81'	Wrong coding of class byte

Value	Meaning
'90 00'	Normal ending of the command

**Table 26 – READ RECORD Status bytes**

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

### Description

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

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

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

If the verification is not successful and if the PIN retry counter is decreased to 0, the “PIN verification” procedure is blocked for the PIN.

### 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.

### 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 <sub>cardholder</sub> )
Lc	Length of verification data
Data	Verification data
Le	Empty

Table 27 – MVP: VERIFY Command APDU

## Response APDU

Field	Value
Data	Empty
SW1-SW2	Status Bytes

Table 28 – MVP: VERIFY 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)
'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 29 – MVP: VERIFY Status bytes

## 6.10 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.

This command shall be used if the user changes his PIN value.

The previous referenced PIN access condition shall be discarded; the current PIN shall be presented and compared with the one stored in the EID card.

If the comparison fails, the PIN retry counter shall be decreased and the PIN value not changed.

Otherwise if the verification is successful, the PIN value shall be modified with the new PIN value and the associated access right granted.

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

The PIN update is performed atomically.

### 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 <sub>cardholder</sub> )
Lc	Length of subsequent data field
Data	Existing PIN    New PIN
Le	Empty

Table 30 – MVP: CHANGE REFERENCE DATA Command APDU



## Response APDU

Field	Value
Data	Empty
SW1-SW2	Status Bytes

**Table 31 – MVP: CHANGE REFERENCE DATA Response APDU**

## Status bytes

Value	Meaning
'63 CX'	Verification failed, ' <b>X</b> ' 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 blocked)
'6A 80'	Incorrect parameter in data field (e.g. wrong PIN format)
'6A 86'	Wrong parameter P1-P2
'6A 88'	Referenced data not found (PIN reference 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 32 – MVP: CHANGE REFERENCE DATA Status bytes**

## 6.11 INTERNAL AUTHENTICATE Command (ISO 7816-4)

### Description

This command is used by the external application to authenticate the EID card (refer to 5.2).

The command will be used in two authentication processes:

1. Internal Authentication (refer to 5.2). In that case the data field contains the challenge from application (CHL), which is an off-card hash to be signed by the applet. The (CHL) length shall be consistent with hash algorithm used. The private key used by the command must be the **basic key** (P2='81'). **New feature V1.8:** The **Internal Authenticate** supports only ECDSA signatures with at least 256-bit modulus. **Key length supported:**

Only the following key length are supported for **Internal Authenticate** according to **EID personalization**:

- ✚ ECDSA key 256 bits
- ✚ ECDSA key 384 bits
- ✚ ECDSA key 512 bits
- ✚ ECDSA key 521 bits

### Algorithms used:

The EID card applet uses the ECDSA algorithm to sign the challenge receives from the internal authenticate command.

Only the following algorithms are available for **Internal Authenticate** according to ECDSA key size personalized:

- ✚ ECDSA using SHA-2-256 algorithm
- ✚ ECDSA using SHA-2-384 algorithm
- ✚ ECDSA using SHA-2-512 algorithm

### Signature schemes:

The following ECDSA signature schemes are supported for **Internal Authenticate**:

ECDSA Schemes	ECDSA Signature Schemes			Table 33 – Signature
	Scheme	ECDSA key length (bits)	Hash algorithm	
	ECDSA	256	SHA-2-256	
		384	SHA-2-384	
		512	SHA-2-512	
		521	SHA-2-512	




### Compatibility note

Applet V1.7h only supports RSA-based internal authenticate commands

### Command chaining:

- The command input chaining is not supported.
- The command response chaining is mandatory for ECDSA keys 512 and 521 bits.

## Conditions of Use

- no security conditions are required.
-  No Security Environment is needed however this command, in normal ending case, clears the current Security Environment. Command structure

Instruction	CLA	INS	P1	P2	Lc	Le	Case
Internal Authenticate	'00'	'88'	Algorithm	private key reference	input data length	output data length	4

## Command APDU

Field	Value
CLA	'00' or '10' (chaining)
INS	'88'
P1	Algorithm reference (refer to Table 14)
P2	Private key reference (refer to Table 16) <ul style="list-style-type: none"> <li>'81' (basic key)</li> </ul>
Lc	Length of the subsequent data field
Data	<u>Internal Authentication Mode</u> : Tag for challenge = '94'    Length <sup>(1)</sup>    CHL
Le	Length of the response

Table 34 – INTERNAL AUTHENTICATE Command APDU



### Notes:

- If **Le** is equal to '00', then it is interpreted as 256 bytes.
- <sup>(1)</sup> See Data field description paragraph. The TLV lengths shall be ASN.1 encoded (see Table 35 - ASN.1 Length Encoding Rules). The Elliptic Curve Diffie-Hellman public keys  $K_{ICC}$  (EID public key) and  $K_{IFD}$  (Terminal public key) are expressed as an uncompressed Elliptic Curve Point, as defined in [\[TR-03111\]](#), composed of the following concatenated fields: '04' ||  $Y_x$  ||  $Y_y$ . The x- and y-coordinates of the public point are octet strings (hexadecimal bytes) with length equivalent to the domain parameter  $n$  (order of base point).



### ASN.1 Length Encoding Rules:

Range	Number of bytes	1 <sup>st</sup> byte	2 <sup>nd</sup> byte	3 <sup>rd</sup> byte
0 to 127	1	binary value	<i>absent</i>	<i>absent</i>
0 to 255	2	'81'	binary value	<i>absent</i>
0 to 65535	3	'82'	binary value MSB byte	LSB byte

Table 35 - ASN.1 Length Encoding Rules

## Data field description:

The data field is coded in TLV format.

### 1. Internal Authentication Mode

The TLV value is CHL, which is an off-card hash used for the card signature.

The data field length is coded as follow according to selected hash algorithm:

- a) **Internal Authenticate** with ECDSA 256 bits basic key and ECDSA using SHA-2-256 signature algorithm:  
Data = Tag for challenge = '94' || Length = '20' || CHL of 32 bytes
- b) **Internal Authenticate** with ECDSA 384 bits basic key and ECDSA using SHA-2-384 signature algorithm:  
Data = Tag for challenge = '94' || Length = '30' || CHL of 48 bytes
- c) **Internal Authenticate** with ECDSA 512 bits basic key and ECDSA using SHA-2-512 signature algorithm:  
Data = Tag for challenge = '94' || Length = '40' || CHL of 64 bytes
- d) **Internal Authenticate** with ECDSA 521 bits basic key and ECDSA using SHA-2-512 signature algorithm:  
Data = Tag for challenge = '94' || Length = '40' || CHL of 64 bytes

## Response APDU

Field	Value
Data	Response (RES <sub>card</sub> ) In the internal authenticate mode: ECDSA Digital Signature in <b>plain format r    s</b>
SW1-SW2	Status Bytes

**Table 36 – INTERNAL AUTHENTICATE Response APDU**

(1) The response is encapsulated into a BER-TLV object.



ECDSA signatures in plain format are encoded as a direct concatenation of two byte strings **r || s** (as specified in [\[TR-03111\]](#)).

## Status bytes

Value	Meaning
'64 00'	No precise diagnostic
'65 81'	EEPROM corrupted (followed by a temporary mute state)
'67 00'	Wrong length
'69 82'	Security Status not satisfied
'69 85'	Condition of use not satisfied (algorithm locked, key not initialized, command blocked)
'6A 80'	Wrong Data (wrong challenge tag or length)
'6A 88'	Referenced key not found (Basic key not found)
'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 37 – INTERNAL AUTHENTICATE Status bytes**

## 6.12 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 14 – ECDSA Algorithm references)
- Selecting the Private Key (by using a key reference inside the **Digital Signature Template**) that shall be used in the digital signature creation process (see Table 16).

Only the private “Signature keys” can be selected for signature (see §4.1):

- Authentication key
- Non-Repudiation key



**New feature V1.8:** Only ECDSA signature algorithm is supported.



See the available signature schemes in Table 41 – ECDSA 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'    Tag for Algorithm reference = '80'    <b>Algorithm reference</b> (refer to Table 14)    Tag for private key reference = '84'    <b>Private key reference</b> (refer to Table 16) = '82', '83'
Le	Empty

Table 38 – MSE: SET command APDU

### Response ADPU

Field	Value
Data	Empty
SW1-SW2	Status Bytes

**Table 39 – 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 40 – MSE: SET Status bytes**

## 6.13 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.

Only "Signature keys" apply to signature algorithms (see §4.1):

- Authentication key
- Non-Repudiation key



**New feature V1.8:** Only ECDSA signature algorithm is supported.

### Key length supported:

The command supports only ECDSA signatures with at least 256-bit modulus according to **EID personalization**:

- ✚ ECDSA key 256 bits is supported.
- ✚ ECDSA key 384 bits is supported.
- ✚ ECDSA key 512 bits is supported.
- ✚ ECDSA key 521 bits is supported.

### Algorithms used:

Only ECDSA signature algorithm is supported:

- ✚ ECDSA with SHA-2/3-256 algorithm is supported.
- ✚ ECDSA with SHA-2/3-384 algorithm is supported.
- ✚ ECDSA with SHA-2/3-512 algorithm is supported.
- ✚ ECDSA with 'hash not specified'<sup>1</sup> is supported.

### Signature schemes:

The following ECDSA signature schemes are supported:

ECDSA Signature Schemes		
Scheme	ECDSA key length (bits)	Hash algorithm
ECDSA	256	SHA-2/3-256 SHA-2/3-384 SHA-2/3-512 Hash not specified
	384	SHA-2/3-256 SHA-2/3-384 SHA-2/3-512 Hash not specified

<sup>1</sup> For the signature scheme ECDSA with 'hash not specified', it is possible to leave the hash algorithm unspecified to support any hash algorithm. In this case the hash lengths (i.e. DTBS) are limited to 160, 224, 256, 384 and 512 bits.

	512	SHA-2/3-256 SHA-2/3-384 SHA-2/3-512 Hash not specified
	521	SHA-2/3-256 SHA-2/3-384 SHA-2/3-512 Hash not specified

Table 41 – ECDSA Signature Schemes



### Compatibility note

Applet V1.7 only supported RSA signatures.  
Applet V1.8 supports only ECDSA signatures.



**Command chaining:** *PSO: CDS* does not support command chaining:

- It is not necessary to support GET RESPONSE since a single 256-byte response APDU can hold a complete raw signature result.

## 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<sub>authentication</sub>** access right must have been granted at any time before using the signature authentication key (refer to §5.1.1).
- Each time the signature non-repudiation key is used, a successful **MVP: Verify (PIN<sub>nonRepudiation</sub>)** APDU command has to be executed either just before or at most one command before using the non-repudiation signing key (refer to §5.1.2).

## 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: <ul style="list-style-type: none"> <li>- 32 bytes for ECDSA with SHA-2/3-256 algorithm</li> <li>- 48 bytes for ECDSA with SHA-2/3-384 algorithm</li> <li>- 64 bytes for ECDSA with SHA-2/3-512 algorithm</li> <li>- 'xx' bytes for ECDSA with 'hash not specified' algorithm with 'xx' = 20, 28, 32, 48 or 64 bytes</li> </ul>
Data	Data to be signed (DTBS) i.e. off-card hash value (not padded):
Le	Length of the signature : <ul style="list-style-type: none"> <li>- '00' (means 256 bytes)</li> </ul>

**Table 42 – PSO: CDS command APDU**

### Response ADPU

Field	Value
Data	Signature: ECDSA Digital Signature in <b>plain format r    s</b>
SW1-SW2	Status Bytes

**Table 43 – PSO: CDS Response APDU**



ECDSA signatures in plain format are encoded as a direct concatenation of two byte strings **r || s** (as specified in [\[TR-03111\]](#)), each one on the length of the elliptic curve base point order n.

### 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)
'6A 80'	Incorrect parameter in data field
'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 solicits a GET RESPONSE command with Le='xy'
'6E 00' / '68 81'	Wrong coding of class byte
'90 00'	Normal ending of the command

**Table 44 – PSO: CDS Status bytes**

## 6.14 GET CARD DATA Command

### Description

This command is used to retrieve some useful information about the card, including the value of the different PIN counters.

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

### Compatibility note

Applet V1.7h only supports backward compatible mode (P2='00')

Applet V1.8 supports legacy (P2='00') and new Pin counter mode (P2='01')

### 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' or '01'	-	Length	2

### Command ADPU

Field	Value
CLA	'80'
INS	'E4'
P1	'00'
P2	'00': backwards compatible output of the GET CARD DATA command '01': same data as with '00', but appended with 3 bytes: <ul style="list-style-type: none"> <li>byte 1: the number of remaining retries for PIN<sub>cardholder</sub></li> <li>byte 2: reserved for future use</li> <li>byte 3: reserved for future use</li> </ul>
Lc	Empty
Data	Empty
Le	Length of the response = '1C' (P2='00') Length of the response = '1F' (P2='01')

**Table 45 - GET CARD DATA Command ADPU**

## Response APDU

Field	Value
Data	<p>If P2 equals '00': Serial Number (16 bytes)     Component code (1 byte)     OS number (1 byte)     OS version (1 byte)     Softmask number (1 byte)     Softmask version (1 byte)     Applet version (1 bytes)     Global OS Version (2 bytes)     Applet interface version (1 bytes)     PKCS#1 support (1 byte)     Key Exchange version (1 byte)     Applet Life cycle (1 byte)</p> <p>If P2 equals '01', the same information as above, but appended with:  # attempts for PIN<sub>authenticate</sub> (1 byte)     # future use(1 byte)     # future use(1 byte)</p>
SW1-SW2	Status Bytes

**Table 46 – GET CARD DATA Response APDU**



If P2 equals '01' and PIN is not found, the # attempts returned for this PIN is set to 'FF'.

## Response APDU description

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 inside all chips from this manufacturer.
Component code	1	Chip dependent
OS number	1	To update according to selected platform
OS version	1	To update according to selected platform
Softmask number	1	'xx' (platform specific)
Softmask version	1	'xx' (platform specific)
Applet version	1	'18' = Applet version 1.8
Global OS Version	2	'0001' = Belpic V1.8
Applet interface version	1	'00'
PKCS#1 support	1	'00' = no PKCS#1 version 2.x support
Key Exchange version	1	'02'
Applet Life cycle	1	'07' = SELECTABLE state '0F' = PERSONALIZED state
# attempts for PIN <sub>authenticate</sub>	1	# attempts remaining for this PIN
future use	1	'FF'
future use	1	'FF'

**Table 47 – 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	<b>Le</b> incorrect, <b>XX</b> 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 48 – GET CARD DATA Status bytes**

## 6.15 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 49 - LOG OFF Command APDU

### Response APDU

Field	Value
Data	Empty
SW1-SW2	Status Bytes

Table 50 – 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 51 – LOG OFF Status bytes

**- END OF DOCUMENT -**