ZETES

EID - Readers technical compatibility

As the EID project is still in a pilot phase, the current document will continue to evolve.

However, no fundamental changes should be introduced in order to minimise the impact on parties using the document as a reference.

This document describes the technical requirements for a smart card reader to be compatible with the Belgian Electronic Identity Card.

This document describes the minimum technical requirements to have a device that ***works*** with the Electronic Identity Card. Note that some more requirements – technical or functional – may be added by some institutions in order to meet their criteria for a compliance with their own policy, or by some official approving instances. Also, all standards required by the European Commission or the Belgian regulation still apply (like electric and electro-magnetic norms, etc.).

ZETES

Change History

Date	Version	Modifications
11-12-2002	1.3	First version for distribution
30-12-2002	1.7	Middleware-level functions extended
07-01-2003	1.8	Added PUK format
07-01-2003	1.9	Replaced ***SCR_USAGE_ENCR*** by ***SCR_USAGE_DECR***
		Replaced ***SCR_USAGE_MAINTENANCE*** by ***SCR_USAGE_ADMIN***
		Clarifications and examples
30-01-2003	2.0	Modification in ***PIN reset*** function
		Corrected ***Lc*** parameter values for commands ***Verify*** and ***Change Reference Data***
		Added values for '***supported***' parameter in '***SCR_Init***' function
05-02-2003	2.1	Corrected PUK lengths in reader commands
		Explanations and examples
07-02-2003	2.2	Only T=0 is supported
17-02-2003	2.4	Corrected parameter ***P2*** in ***Card activate***
		Added support for incorrect PIN entry handling
		Specified time-out
		Added examples of translated messages
		Added two return codes for the DLL functions
24-02-2003	2.5	Generalised Global Platform format for all PUK (cf. readers commands)
25-02-2003	2.6	Correction: interface version = **1** in '***SCR_Init***'
12-03-2003	2.7	Corrected PIN reference in ***Card Activate*** example
19-06-2003	2.7.1	Typos
24-07-2003	2.7.2	Added the new proposed card ATR
12-08-2003	2.7.3	Typos
16-09-2003	2.7.4	Made more clear that PUK lengths may be variable
30-10-2003	2.7.5	Made PIN reference explicit in ***RESET RETRY COUNTER*** and ***PIN Reset***

ZETES

Table of contents

ZETES

1. Hardware compatibility with the chip

The reader must be compliant with the following norms, standards or recommendations:

- ***ISO/IEC 7816-1:1998*** *Integrated circuit(s) cards with contacts – Part 1: Physical characteristics*
- ***ISO/IEC 7816-2:1999*** *Integrated circuit(s) cards with contacts – Part 2: Dimensions and location of the contacts*
- ***ISO/IEC 7816-3:1997*** *Integrated circuit(s) cards with contacts – Part 3: Electronic signals and transmission protocols*
  - ***Amd 1:2002*** *Electrical characteristics and class indication for integrated circuit(s) cards operating at 5 V, 3 V and 1,8 V*
- ***ISO/IEC 7816-10:1999*** *Integrated circuit(s) cards with contacts – Part 10: Electronic signals and answer to reset for synchronous cards*
- ID1 card format
- Protocol T=0 (T=1 is optional)
- Electrical tension VCC between 1.62 V and 5 V

ZETES

2. Software compatibility with the chip

2.1. All readers

All readers must comply with the following standards for the exchange commands format:

- ***ISO/IEC 7816-4:1995*** *Integrated circuit(s) cards with contacts – Part 4: Inter-industry commands for interchange*
- ***ISO/IEC 7816-8:1999*** *circuit(s) cards with contacts – Part 8: Security related inter-industry commands*

Card ATR before 1-1-2004:	3B	98	94	40	FF	A5	03	01	01	01	AD	13	10

Card ATR from 1-1-2004:	3B	98	94	40	0A	FF	A5	03	01	01	01	AD	13	10

2.2. Readers with PIN-pad

Readers with a PIN-pad must implement all commands (APDU) needing a PIN input without transmitting the PIN outside the reader.

2.2.1. Time-out

If no PIN is entered after a certain amount of time, the command must return with a time-out status. This time-out can occur either if no key is pressed, or if the time is too long between two keystrokes.

The time-out must be between 15 seconds and 40 seconds (best 30 seconds).

2.2.2. PIN / PUK input

As the PIN (and possibly the PUK) may have a variable length, the citizen must end the PIN / PUK entry by hitting the OK button.

2.2.3. Incorrect PIN entry handling

When the card returns an status word ***'63 Cx'*** to a ***VERIFY*** command depending on a user input, meaning that the PIN / PUK is wrong and that '**x**' retries remain before blocking that PIN / PUK, the reader must notify the citizen; it must:

- ❒ display an error message containing the number of remaining retries
- ❒ wait for the citizen to hit the OK button
- ❒ if the citizen didn't hit the OK button before the time-out, stop waiting
- ❒ return the status word ***'63 Cx'*** to the application

The error message should look like

```
PIN erroné – essais restants: 2
Appuyez sur 'OK'
```

This functionality can be implemented in the DLL described in section 3.2, or directly in the firmware.

2.2.4. Display clearing

After an interaction on the reader, the display must always be cleared – either after the citizen's input, or after the time-out. A vendor-dependant message may optionally be displayed.

This functionality can be implemented in the DLL described in section 3.2, or directly in the firmware.

2.2.5. Secure PIN entry notification

When asking for a PIN / PUK that will be sent directly to the card, the reader should display on the screen a symbol (preferably a key symbol) to notify the citizen that the reader is in 'secure PIN entry mode' and not in a mode where the PC reads the keys hit on the pinpad.

Only the firmware must be able to display this symbol.

ZETES

2.2.6. Card commands (APDU) to implement

- ***VERIFY (PIN Verify) – ISO 7816-4 § 6.12***

Field	Value
CLA	'00'
INS	'20'
P1	'00'
P2	PIN reference
Lc	Length of verification data – always '08'
Data	Verification data
Le	Empty

- ***CHANGE REFERENCE DATA (PIN Change) – ISO 7816-8 § 12.2***

Field	Value
CLA	'00'
INS	'24'
P1	'00' (User) '01' is intended for PIN reset, thus not covered here
P2	PIN reference
Lc	Length of subsequent data field – always '10'
Data	Existing PIN **||** New PIN (concatenation)
Le	Empty

ZETES

2.2.7. Reader commands (APDU) to implement

This section describes the APDU command the reader must implement in order to interact with the PIN–related commands from the card.

To be accessed through the standard EID middleware, the manufacturer must supply the DLL implementing the right interface, as described in section 3.2. In this case, a proprietary firmware interface may be used and this section does not apply.

When however the reader is accessed through some other means (e.g. custom applications) then the proposed commands are sufficient for application builders. Although this solution will technically work, it is quite limited and could be refused by some institutions and some official approving instances that may require the compatibility with section 3.2.

Both commands described in section 2.2.6 must be automatically called with a PIN (or 2 PIN) asked on the PIN-pad when the parameter ***Lc*** from the above commands is equal to ***0***. In this case, the reader's firmware must first read the PIN from the PIN-pad, then substitute the parameters ***Lc*** and ***Data*** with the right information constructed from the entered PIN.

The return code (status byte) should be the one from the card command, one defined in the ISO 7816-4 , and 7816-8 standards, or one of the following:

Status byte	Meaning
'ECD2'	PIN/PUK entry time-out
'ECD6'	Cancelled by the user
'ECB6'	Unknown error

- ***PIN Verify on Reader***

Field	Value
CLA	'00'
INS	'20'
P1	'00'
P2	PIN reference
Lc	**'00'**
Data	**Empty**
Le	Empty

Lc will become **'08'** when the reader sends the APDU to the card with the PIN data

- ***PIN Change on Reader***

Field	Value
CLA	'00'
INS	'24'
P1	'00' (User)
P2	PIN reference
Lc	**'00'**
Data	**Empty**
Le	Empty

For the PIN Change, the new PIN must be entered twice (with comparison) to avoid mistakes.

Consequently, the length **Lc** will become **'10'**, since both the old value and the new value will be sent to the card.

2.2.8. PIN format

The PIN are 8-bytes strings with the following format (by nibble) as defined in the ***Global PIN*** section from the document "***Global Platform - Card Specification – version 2.0.1 – April 7, 2000***" and in the ***F2PB*** section from "***ISO 9564-1 – Banking – Personal Identification Number (PIN) management and security – 2002***":

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	PIN digits (minimum 4)
P/'F'	The rest of the PIN digits depending on the length, 'F' else
'F'	Padding contains always 'F'

Example: The PIN '1234' must be coded as

2	4	1	2	3	4	F	F	F	F	F	F	F	F	F	F

ZETES

2.3. Special readers for the Municipality

Readers used to activate and reset the cards in the Municipalities have to be compliant with point 2.2 (Readers with PIN-pad). Additionally these readers also have to implement some special functions.

2.3.1. Card commands (APDU) to implement

- ***CHANGE REFERENCE DATA (PIN Change) – ISO 7816-8 § 12.2***

Field	Value
CLA	'00'
INS	'24'
P1	'00' (User) **'01' (Administrator)**
P2	PIN reference
Lc	Length of subsequent data field – **'08' or** '10'
Data	P1 = '00' - Existing PIN || New PIN (concatenation) **P1 = '01' - New PIN**
Le	Empty

- ***RESET RETRY COUNTER – ISO 7816-8 §12.5***

Field	Value
CLA	'00'
INS	'2C'
P1	'00'
P2	PIN reference
Lc	'08'
Data	Verification data (PIN_{reset})
Le	Empty

- ***GET CHALLENGE – ISO 7816-4 §6.15***

Field	Value
CLA	'00'
INS	'84'
P1	'00'
P2	'00'
Lc	Empty
Data	Empty
Le	Length of the required random value (Le<>0)

ZETES

2.3.2. Reader commands (APDU) to implement

The return code (status byte) should be the one from the card command, one defined in the ISO 7816-4 , and 7816-8 standards, or one of the following:

Status byte	Meaning
'ECD2'	PUK entry time-out
'ECD6'	Cancelled by the user
'ECB6'	Unknown error

- ***Internal function: Merge(algorithm, PIN)***

Algorithm:	***PIN:***	***Output:***
1. activate 2. unblock 3. reset	PIN to merge with PIN read on the PIN-pad.	resulting PIN

These PIN can never leave the reader.

The merge algorithms are very basic to implement. The exact algorithms are not public, and will be given to the reader producers that will implement them.

ZETES

❒ ***Card activate***

Field	Value
CLA	'00'
INS	'F5'
P1	'01'
P2	'00'
Lc	Data length '08'
Data	PUK_2 (to be merged with PUK_1 read on the PIN-pad)
Le	Empty

1. Data displayed on the reader's: `ID-M: PUK1 ? ******`
2. The reader calls $\boldsymbol{PIN_{Activate} = Merge(\ activate, PUK_2\ )}$
3. The reader calls the ***PIN Verify*** command with
   P1='00', P2='84', Lc='08', Data=$PIN_{Activate}$

Example: PUK_2 given in the function is: 'ABCDEF', which is coded in the APDU as:

2	6	A	B	C	D	E	F	F	F	F	F	F	F	F	F

PUK_1 typed in by the user is: '123456' (6 digits)

The resulting $PIN_{Activate}$ is:

2	C	x	x	x	x	x	x	x	x	x	x	x	x	F	F

Where **xx xx xx xx xx xx** is the result of the PIN merge algorithm

The APDU in step 3 is: **00 20 00 84 08 2C xx xx xx xx xx xx FF**

ZETES

❒ ***PIN Unblock***

Field	Value
CLA	'00'
INS	'F5'
P1	'02'
P2	PIN reference
Lc	Data length '08'
Data	PUK_2 (to be merged with PUK_1 read on the PIN-pad)
Le	Empty

1. Data displayed on the reader's: `ID-M: PUK1 ? ******`
2. The reader calls $\boldsymbol{PIN_{Unblock} = Merge(\ unblock,\ PUK_2\ )}$
3. The reader calls the ***Reset Retry Counter*** command with ***P1='00', P2='01', Lc='08', Data=$PIN_{Unblock}$***

Example: $\boldsymbol{PUK_2}$ given in the function is: 'ABCDEF', which is coded in the APDU as:

2	6	A	B	C	D	E	F	F	F	F	F	F	F	F	F

$\boldsymbol{PUK_1}$ typed in by the user is: '123456' (6 digits)

The resulting $\boldsymbol{PIN_{Activate}}$ is:

2	C	x	x	x	x	x	x	x	x	x	x	x	x	F	F

Where **xx xx xx xx xx xx** is the result of the PIN merge algorithm

The APDU in step 3 is: **00 2C 00 01 08 2C xx xx xx xx xx xx FF**

❒ ***PIN Reset***

Field	Value
CLA	'00'
INS	'F5'
P1	'03'
P2	PIN reference
Lc	Data length '08'
Data	PUK_3 (to be merged with PUK_1 read on the PIN-pad)
Le	Empty

1. The reader asks a random to the card (***Get Challenge***)

2. PIN_{New} = 4 nibbles (4 is an arbitrary choice). Each nibble must be constructed out of the bits from this random, with '0' as most significant bit.
   Example: random = 0100 1001 1111 0010 …
   PIN_{New} = 0100 0001 0111 0010

3. Data displayed on the reader's: `ID-M: PUK1 ? ******`

4. The reader calls PIN_{Reset} = ***Merge(reset, PUK_3)***

5. The reader calls the ***PIN Verify*** command with
   P1='00', P2='02', Lc='08', Data=PIN_{Reset}

6. The reader calls the ***Change Reference Data*** command with
   P1='01', P2='01', Lc='08', Data=PIN_{New}

7. Data displayed on the reader's: `ID-M: PIN = 1234`

Example: PUK_3 given in the function is: 'ABCDEF', which is coded in the APDU as:

2	6	A	B	C	D	E	F	F	F	F	F	F	F	F	F

PUK_1 typed in by the user is: '123456' (6 digits)

The resulting PIN_{Reset} is:

2	C	x	x	x	x	x	x	x	x	x	x	x	x	F	F

Where **xx xx xx xx xx xx** is the result of the PIN merge algorithm

The APDU in step 5 will be: **00 20 00 02 08 2C xx xx xx xx xx xx FF**

If the random number received by the ***Get Challenge*** is: '49 F2 A4 ...',
then PIN_{New} will be: '4172'.

The data PIN_{New} sent to the card in the "Change Reference Data" command will be coded as:

2	4	4	1	7	2	F	F	F	F	F	F	F	F	F	F

The APDU in step 6 will be: **00 24 01 01 08 24 41 72 FF FF FF FF FF**

ZETES

2.3.3. PUK format

The PUK are 8-bytes strings with the following format (by nibble) as defined in the ***Global PIN*** section from the document "***Global Platform - Card Specification – version 2.0.1' – April 7, 2000***" and in the ***F2PB*** section from "***ISO 9564-1 – Banking – Personal Identification Number (PIN) management and security – 2002***":

C	L	P	P	P	P	P	P	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 PUK (in nibbles) – currently '6'
P	PUK digits
P/'F'	The rest of the PUK digits depending on the length, 'F' else
'F'	Padding contains always 'F'

Example: The PUK '123456' must be coded as

2	6	1	2	3	4	5	6	F	F	F	F	F	F	F	F

ZETES

3. Software compatibility with the middleware

This section is intended for readers that connect to a computer that will use the card through either the ***Microsoft CryptoAPI***, or through the ***PKCS#11*** interface.

3.1. PC/SC drivers

All readers must provide a driver compliant with ***PC/SC*** version 1.1 for the target Operating System (see www.pcscworkgroup.com).

3.2. Integration with the middleware

In order to integrate with the middleware, the reader provider must develop, for each target Operating System, a ***Dynamic Linked Library*** (or equivalent) implementing the following functions:

- ***SCR_Init()***
- ***SCR_VerifyPIN()***
- ***SCR_ChangePIN()***

A ***C*** header file is provided at the end of the document, with all the function prototypes and the constants for the return codes and parameters.

Currently, only the Belgian Identity Application resides on the card; to anticipate possible other applications (implemented as separate *Data Files* or *Directories* in the card) in the future, some functions receive a parameter called ***Application identifier*** (***AID***) that contains a string identifying the application that want to interact with one of its PIN. The goal is that the reader displays (see 3.3) the application that asks for a PIN.

3.2.1. Parameters and return code

- All functions must return a standard ***PC/SC*** return code (LONG).
  Additional return codes:
  - SCARD_E_CANCELLED: the user pressed Cancel
  - SCR_E_UNINITIALIZED: SCR_Initialised( ) was not called
  - SCR_I_PIN_CHECK_FAILED: New PIN mismatch in SCR_ChangePIN( )
- The parameter ***language*** is a 2-characters string as defined in the ISO 639 standard.
  The DLL must, at least, support the following languages:
  - ***fr*** French
  - ***nl*** Dutch
  - ***de*** German
  - ***en*** English
- The functions use the following structures:

SCR_Bytes		
data	BYTE *	The data itself
length	DWORD	The data length

SCR_Card		
hCard	SCARDHANDLE	PC/SC handle to the card
language	char *	ISO 639 2-characters code
id	SCR_bytes	Card type unique identifier. Usually the PKCS#15 ***OID***.
pinFormat	void *	Not used yet; should be ***NULL***

SCR_Application		
id	SCR_bytes	Application unique identifier (***AID***).
shortString	char *	3-characters (max.) string application identifier in the user's language
longString	char *	Long string application identifier in the user's language

SCR_PinUsage			
code	DWORD	SCR_USAGE_AUTH SCR_USAGE_SIGN SCR_USAGE_DECR SCR_USAGE_PREF_MODIF SCR_USAGE_ADMIN	Authentication / logon Non-repudiation Decryption Preference file modification Administrative maintenance
shortString	char *	3-characters (max.) string application identifier in the user's language	
longString	char *	Long string application identifier in the user's language	

3.2.2. SCR_Init( )

This function will be called immediately after loading the DLL. It is used to initialise the DLL and check if it supports the connected reader.

Parameters	***in/out***	***Type***	***Description***
szReader	in	LPCTSTR	Reader name
version	in	DWORD	Calling program interface version: **1**
supported	out	DWORD *	- SCR_SUPPORT_OK - SCR_SUPPORT_INCOMPATIBLE_CALLING_VERSION - SCR_SUPPORT_INCOMPATIBLE_FIRMWARE - SCR_SUPPORT_INCOMPATIBLE_FIRMWARE_VERSION *This allows to dynamically try all registered DLL to find the one corresponding to the reader –cf. "plug and play" mechanism.*

3.2.3. SCR_VerifyPIN( )

This function verifies a user's PIN entered on the reader's PIN-pad.

Parameters	***in/out***	***Type***	***Description***
card	in	const SCR_Card *	Handle to the card
pinID	in	BYTE	PIN identifier in the card – provided by the middleware. To specify as parameter P2 in function ***Reader PIN Verify***
usage	in	const SCR_PinUsage *	Reason to enter the PIN
application	in	const SCR_Application *	Application to display on the reader's display.
cardStatus	out	BYTE *	Card status (2 bytes). Fill with 'O' if no specific answer from the card

ZETES

3.2.4. SCR_ChangePIN( )

This function changes a user's PIN entered on the reader's PIN-pad by a new one, also entered on the reader's PIN-pad.

Parameters	*in/out*	*Type*	*Description*
card	in	const SCR_Card *	Handle to the card
PinID	in	BYTE	PIN identifier in the card. To specify as parameter P2 in function ***Reader PIN Change***
application	in	const SCR_Application *	Application to display on the reader's display.
cardStatus	out	BYTE *	Card status (2 bytes). Fill with 'O' if no specific answer from the card

3.3. Display

The parameters ***usage*** and ***application*** are important information for the citizen.

usage shows why the citizen is asked to enter his PIN (to authenticate himself, to sign a document, etc.).

The DLL has to present, on the reader's display, a string corresponding to this usage; the DLL may either display the provided ***usage.longString***, or the provided ***usage.shortString*** if the display is very small,or make use of the provided ***usage.code*** to optimise it for its display.

application identifies the application. Currently, only the Belgian Identity application (***ID***) exists on the card, but some other ones could come later (ex: Doctors: ***Doc***, Lawyers: ***LAW***, etc.). Therefore this information must be given also. The same rule applies for ***application.longString***, ***application.shortString***, and ***application.code***.

Here is an example of the information the reader must provide on its display when calling the above functions:

3.3.1. SCR_VerifyPIN( )

If the display has limited space, the minimum information to display is :

ApplicationID-Usage: PIN ? ****

ex: ID-S: PIN ? ****.

If the display has space enough, the information should be translated in the language defined in the ***SCR_Init*** function, like

Application:	*Identité*
Accès:	*Signature (non-répudiation)*
Entrez votre PIN:	**** .

3.3.2. SCR_ChangePIN( )

If the display has limited space, the minimum information to display is :

ApplicationID-PIN: Old PIN ? ****	*ApplicationID*-PIN: New PIN ? ****

ex:

Doc-PIN: Oude PIN ? ****	Doc-PIN: Nieuwe PIN ? ****

If the display has space enough, the information should be translated in the language defined in the ***SCR_Init*** function, like

```
                PIN Verandering
Applicatie:          Advocaten
Oude    PIN ?        ****
Nieuwe PIN ?         ****
Nieuwe PIN ?         ****    (Controle)
```

ZETES

3.4. C Header file “scr.h”

```
#include <winscard.h>

#ifndef IN
# define IN
#endif

#ifndef OUT
# define OUT
#endif

// ‘usage’ parameter
#define SCR_USAGE_AUTH          1L
#define SCR_USAGE_SIGN          2L
#define SCR_USAGE_DECR          4L
#define SCR_USAGE_PREF_MODIF    8L
#define SCR_USAGE_ADMIN        16L

// ‘supported’ parameter of ‘SCR_Init’ function
#define SCR_SUPPORT_OK                                  0L
#define SCR_SUPPORT_INCOMPATIBLE_CALLING_VERSION        1L
#define SCR_SUPPORT_INCOMPATIBLE_FIRMWARE               2L
#define SCR_SUPPORT_INCOMPATIBLE_FIRMWARE_VERSION       3L

// Errors
#define SCR_E_UNINITIALIZED    0xE0100E01L // SCR_Initialised not called
#define SCR_I_PIN_CHECK_FAILED 0x60100E02L // PIN mismatch in SCR_ChangePIN


typedef struct _SCR_Bytes {
  BYTE *data;
  DWORD length;
} SCR_Bytes;
```

```
typedef struct _SCR_Card {
  SCARDHANDLE hCard;
  char *language;
  SCR_Bytes id;
  void *pinFormat;  // reserved for future use
} SCR_Card;

typedef struct _SCR_Application {
  SCR_Bytes id;
  char *shortString;
  char *longString;
} SCR_Application;

typedef struct _SCR_PinUsage {
  DWORD code;
  char *shortString;
  char *longString;
} SCR_PinUsage;
```

```
/* All return codes comply with PC/SC return codes */
/* If the user presses CANCEL, SCARD_E_CANCELLED must be returned */

LONG SCR_Init(
  IN LPCTSTR szReader,
  IN DWORD version,
  OUT DWORD *supported
);

LONG SCR_VerifyPIN(
  IN const SCR_Card *card,
  IN BYTE pinID,
  IN const SCR_PinUsage *usage,
  IN const SCR_Application *application,
  OUT BYTE *cardStatus
);

LONG SCR_ChangePIN(
  IN const SCR_Card *card,
  IN BYTE pinID,
  IN const SCR_Application *application,
  OUT BYTE *cardStatus
);
```

ZETES

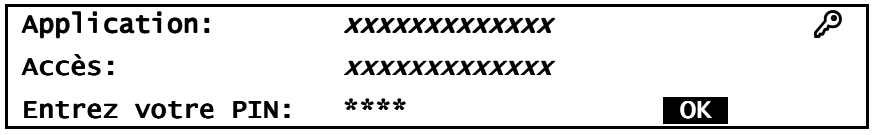
4. Annex 1 – Example of messages

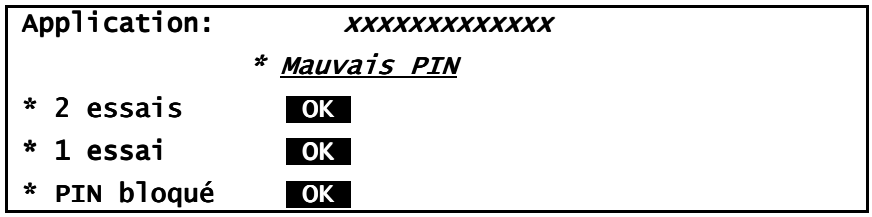
Here is an example of the information to display, in 4 languages.

This has to be adapted to fit the target display.

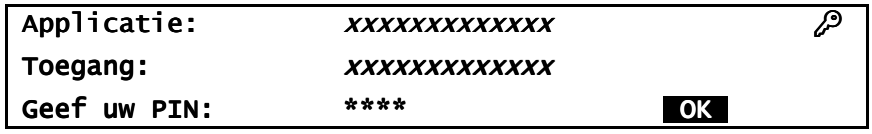
4.1. PIN Verify

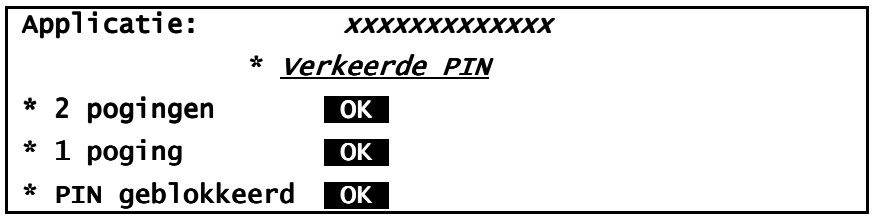
4.1.1. French





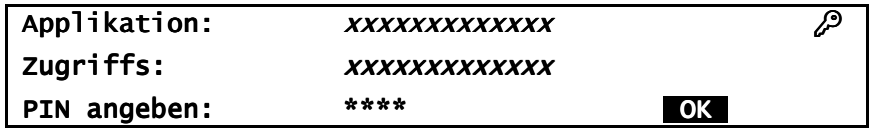
4.1.2. Dutch

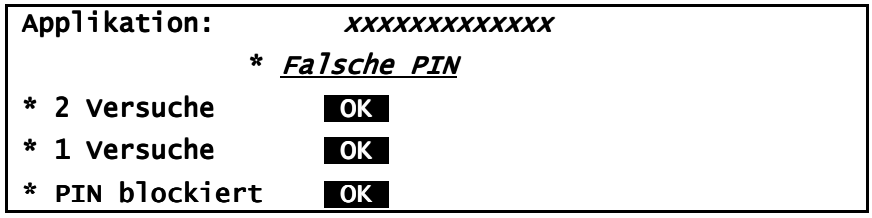




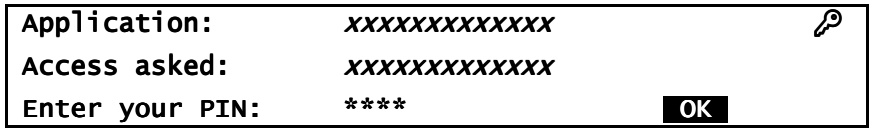
ZETES

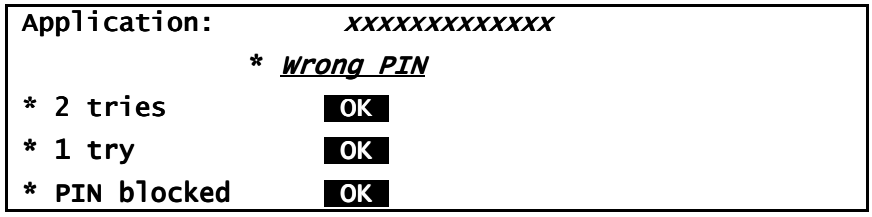
4.1.3. German





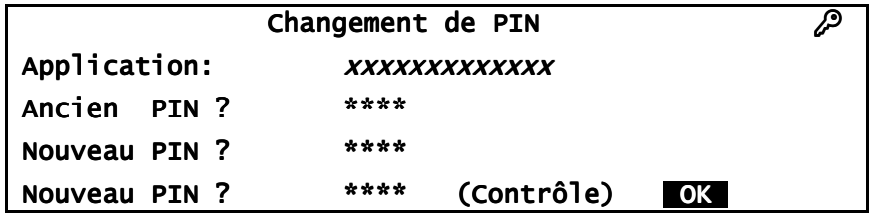
4.1.4. English

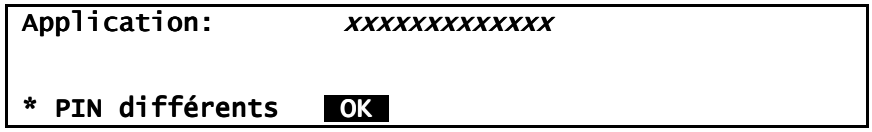




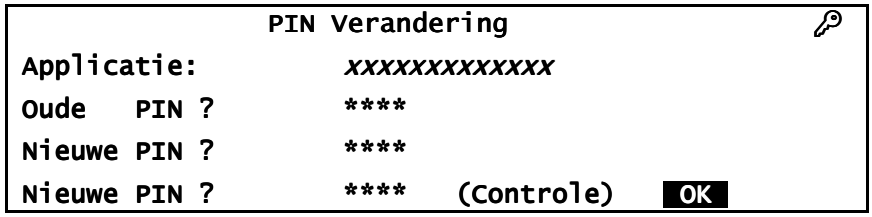
4.2. PIN Change

4.2.1. French

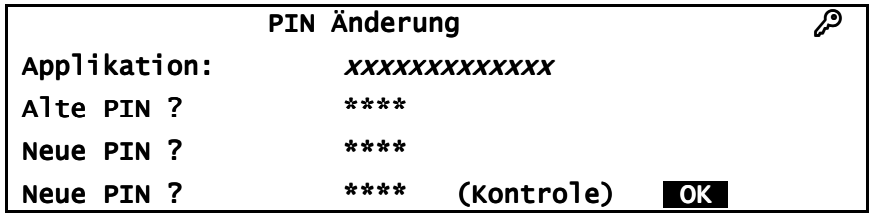


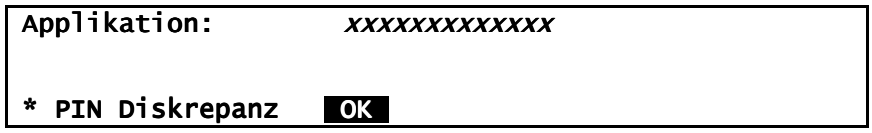


4.2.2. Dutch

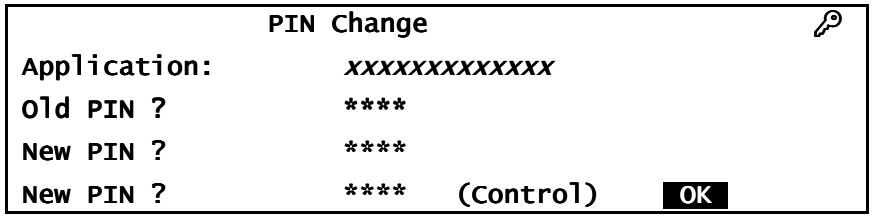


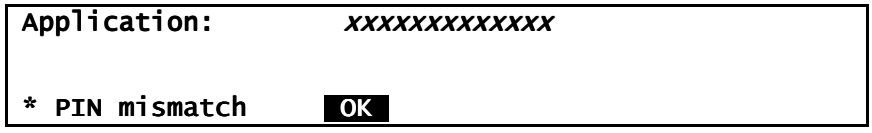
4.2.3. German





4.2.4. English





ZETES

5. Annex 2 – Compatibility sheet

This page is intended to be filled by readers producers to document the compatibility of their product with these specifications.

Mark:..... Model: .....

Supported Operating Systems	PC/SC drivers	DLL (see section 3.2)

Please tick the supported features compliant with the specifications

❒ PIN entry: max. .....digits

- ❒ Display short messages
- ❒ Display long messages
- ❒ Incorrect PIN entry message
- ❒ Display clearing after input
- ❒ Secure PIN entry notification

❒ Time-out: .....seconds Max. communication speed:.....bauds

❒ PIN change

❒ Municipality commands