

# **Interoperability Specification for ICCs and Personal Computer Systems**

## *Part 10 IFDs with Secure PIN Entry Capabilities*

*Apple Computer, Inc.*

*Axalto*

*Gemplus SA*

*Infineon Technologies AG*

*Ingenico SA*

*KOBIL*

*Microsoft Corporation*

*OMNIKEY*

*Philips Semiconductors*

*SCM Microsystems*

*Toshiba Corporation*

**Revision 2.02.05**

**December 2008**

**Copyright © 1996–2008 Apple, Axalto, Gemplus, Hewlett-Packard, Kobil, IBM, Infineon, Ingenico, Microsoft, Omnikey, Philips, SCM Microsystems, Siemens, Sun Microsystems, Toshiba and VeriFone.  
All rights reserved.**

**INTELLECTUAL PROPERTY DISCLAIMER**

**THIS SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED OR INTENDED HEREBY. AXALTO, BULL CP8, GEMPLUS, HEWLETT-PACKARD, IBM, MICROSOFT, SIEMENS NIXDORF, SUN MICROSYSTEMS, TOSHIBA, AND VERIFONE DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF PROPRIETARY RIGHTS, RELATING TO IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. AXALTO, BULL CP8, GEMPLUS, HEWLETT-PACKARD, IBM, MICROSOFT, SIEMENS NIXDORF, SUN MICROSYSTEMS, TOSHIBA, AND VERIFONE, DO NOT WARRANT OR REPRESENT THAT SUCH IMPLEMENTATION(S) WILL NOT INFRINGE SUCH RIGHTS.**

Windows and Windows NT are trademarks and Microsoft and Win32 are registered trademarks of Microsoft Corporation.

PS/2 is a registered trademark of IBM Corp. JAVA is a registered trademark of Sun Microsystems, Inc. All other product names are trademarks, registered trademarks, or servicemarks of their respective owners.

## Revision History

Revision	Issue Date	Comments
2.00.00	March 1st, 2005	Spec 2.00 First Draft
2.01.00	March 8, 2005	Spec 2.00 Reviewed Draft
2.01.01	March 24, 2005	Spec 2.00 Reviewed Draft – minor revisions
2.01.02	April 19, 2005	Minor edits
2.01.03	June 24, 2004	Spec 2.01 Final Release
2.01.04	September 13, 2005	Corrected data types in PIN structure
2.01.05	September 29, 2005	Changed Schlumberger to Axalto
2.01.06	December 2, 2005	Changes for GET_KEY_PRESSED
2.02.00	May 17, 2006	New feature : applications can provide messages for secure pin entry Cosmetic changes
2.02.01	November 7, 2006	IOCTL for writing to the display and retrieving key information added
2.02.02	March 20, 2007	Microsoft: Added comments to address follow-ups from January 2007 Teleconference meeting.
2.02.03	October 2007	Microsoft: updated based on comments from May 2007 meeting.
2.02.04	November 2008	Gemalto : added warning regarding use of GEY_KEY_PRESSED and WRITE_DISPLAY. Added error code return when SET_SPE_MESSAGE exceeds storage capability. Typo : changed all SPE_SET_MESSAGE to SET_SPE_MESSAGE for consistency Added possible connection to the reader in 2.3
2.02.05	December 2008	Gemalto : changed 'C' structures to tables based of byte offsets; structure packing notes removed. UTF-8 is now the only character set used. "#define" feature removed from 2.3 Some typo and precision changes. Structure and feature chapters created. Rows and Columns (for WRITE_DISPLAY and GEY_KEY) start at index 0.

---

## Contents

---

<b>1</b>	<b>SYSTEM ARCHITECTURE</b>	<b>1</b>
<b>2</b>	<b>DEFINITION OF FEATURES AND CONTROL CODES</b>	<b>3</b>
2.1	General Description	3
2.2	GET_FEATURE_REQUEST	3
2.3	Definition of Features	4
2.4	Function Call	5
2.5	Structure list	5
2.5.1	Type definitions	5
2.5.2	PIN_VERIFY	7
2.5.3	PIN_MODIFY	9
2.5.4	MCT_UNIVERSAL	11
2.5.5	PIN_PROPERTIES	12
2.5.6	DISPLAY_PROPERTIES	12
2.5.7	SET_SPE_MESSAGE	13
2.5.8	PIN_VERIFY_APP_ID	14
2.5.9	PIN_MODIFY_APP_ID	16
2.5.10	WRITE_DISPLAY	18
2.5.11	GET_KEY	18
2.6	Feature list	20
2.6.1	FEATURE_VERIFY_PIN_START / FEATURE_MODIFY_PIN_START	20
2.6.2	FEATURE_GET_KEY_PRESSED	21
2.6.3	FEATURE_VERIFY_PIN_FINISH / FEATURE_MODIFY_PIN_FINISH	22
2.6.4	FEATURE_VERIFY_PIN_DIRECT / FEATURE_MODIFY_PIN_DIRECT	23
2.6.5	FEATURE_ABORT	24
2.6.6	FEATURE_MCT_READERDIRECT	25
2.6.7	FEATURE_MCT_UNIVERSAL	25
2.6.8	FEATURE_IFD_PIN_PROPERTIES	26
2.6.9	FEATURE_IFD_DISPLAY_PROPERTIES	26
2.6.10	FEATURE_SET_SPE_MESSAGE	28
2.6.11	FEATURE_VERIFY_PIN_DIRECT_APP_ID / FEATURE_MODIFY_PIN_DIRECT_APP_ID	29
2.6.12	FEATURE_WRITE_DISPLAY	31
2.6.13	FEATURE_GET_KEY	32
<b>3</b>	<b>ABBREVIATIONS</b>	<b>33</b>
<b>4</b>	<b>REFERENCES</b>	<b>34</b>

## 1 System Architecture

This document deals with secure PIN entry for class 2/3 readers and their integration into the PC/SC architecture.

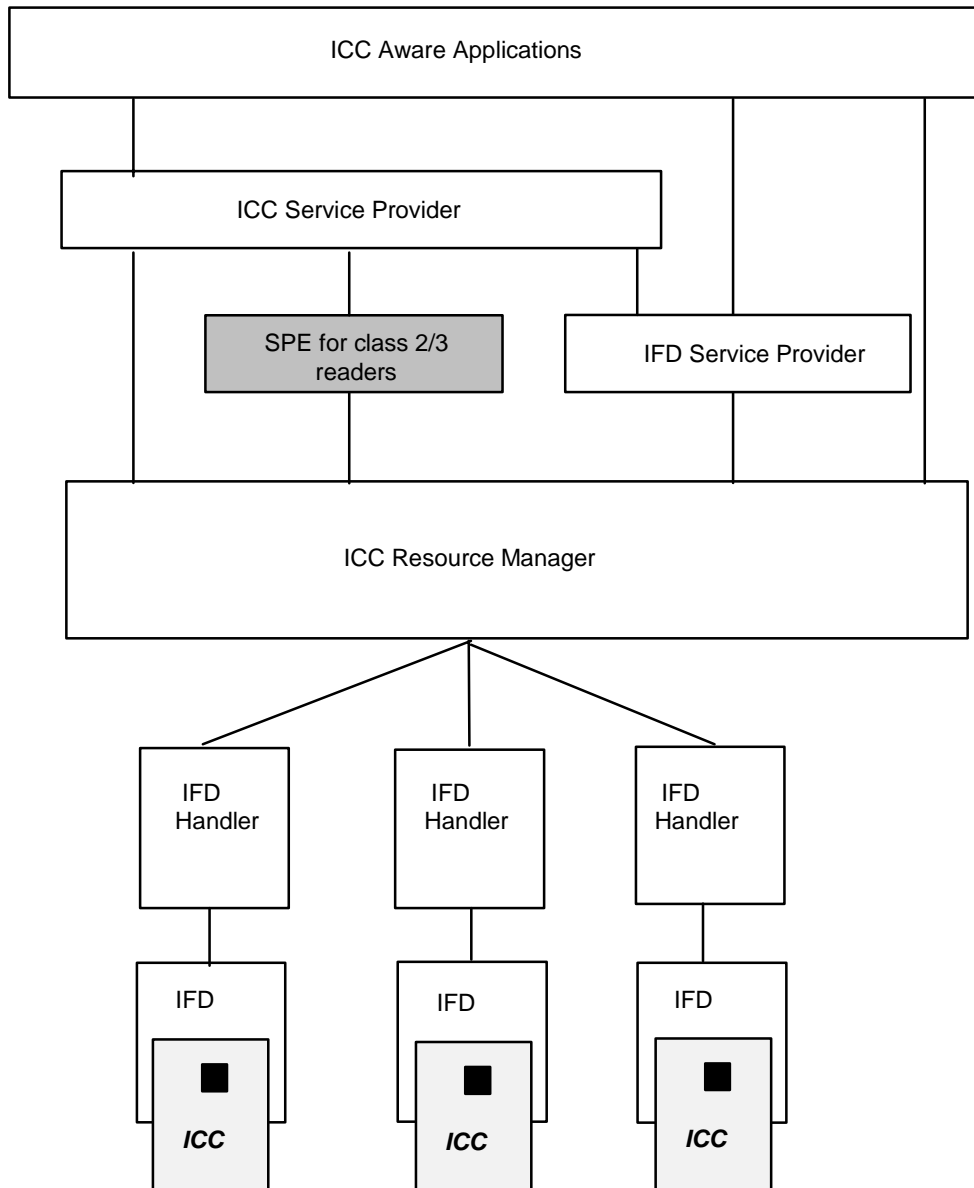


Figure 1- General Architecture

## **2 Definition of Features and Control Codes**

### **2.1 General Description**

An application queries the IFD handler via a special control code, whose features are supported by the IFD.

In the response, the application receives the control codes for all supported features. This mechanism enables different vendors/manufacturers of IFD handlers to define their own control codes.

The control codes, the features and the corresponding parameters are defined as follows.

### **2.2 GET\_FEATURE\_REQUEST**

The corresponding control code is 3400 decimal.

It is mandatory for class 2 drivers to support this control code.

#### **Out:**

The driver shall return SCARD\_SUCCESS and a TLV oriented structure as follows:

**Tag, Length (always 4 bytes), control code value for supported feature**

The control code is returned in big endian format and must be used for the Control function of the Resource Manager in unmodified form.

Implementation Note for Windows:

Under Windows the control code must be defined as follows:

```
#define CM_IOCTL_GET_FEATURE_REQUEST          SCARD_CTL_CODE(3400)
```

## 2.3 Definition of Features

The following features are currently defined:

FEATURE_VERIFY_PIN_START	0x01
FEATURE_VERIFY_PIN_FINISH	0x02
FEATURE_MODIFY_PIN_START	0x03
FEATURE_MODIFY_PIN_FINISH	0x04
FEATURE_GET_KEY_PRESSED	0x05
FEATURE_VERIFY_PIN_DIRECT	0x06
FEATURE_MODIFY_PIN_DIRECT	0x07
FEATURE_MCT_READER_DIRECT	0x08
FEATURE_MCT_UNIVERSAL	0x09
FEATURE_IFD_PIN_PROPERTIES	0x0A
FEATURE_ABORT	0x0B
FEATURE_SET_SPE_MESSAGE	0x0C
FEATURE_VERIFY_PIN_DIRECT_APP_ID	0x0D
FEATURE_MODIFY_PIN_DIRECT_APP_ID	0x0E
FEATURE_WRITE_DISPLAY	0x0F
FEATURE_GET_KEY	0x10
FEATURE_IFD_DISPLAY_PROPERTIES	0x11

e.g. a IFD handler which supports all features will return:

```
01 04 XX XX XX XX 02 04 XX XX XX XX 03 04 XX XX XX XX 04 04 XX XX XX XX 05 04  
XX XX XX XX 06 04 XX XX XX XX 07 04 XX XX XX XX 08 04 XX XX XX XX 09 04 XX  
XX XX XX 0A 04 XX XX XX XX 0B 04 XX XX XX XX 0C 04 XX XX XX XX 0D 04 XX XX  
XX XX 0E 04 XX XX XX XX 0F 04 XX XX XX XX 10 04 XX XX XX XX 11 04 XX XX XX  
XX
```

All control codes are sent to the IFD handler by the Control function of the Resource Manager. Before any control code can be used, a connection to the smart card or the reader is required. This can be achieved by the Connect function.

## 2.4 Function Call

```
RESPONSECODE Control (  
    IN          DWORD          ControlCode,  
    IN          BYTE[]        InBuffer,  
    IN OUT     BYTE[]        Out Buffer,  
    OUT        DWORD          OutBufferLength  
)
```

Implementation Note for Windows:  
Under Windows following function must be used:

```
LONG ScardControl (  
    SCARDHANDLE hCard,  
    DWORD       dwControlCode,  
    LPCVOID     lpInBuffer,  
    DWORD       nInBufferSize,  
    LPVOID     lpOutBuffer,  
    DWORD       nOutBufferSize,  
    LPDWORD    lpBytesReturned  
)
```

The total length of the TLV structure can be retrieved from the lpBytesReturned parameter of the ScardControl function.

## 2.5 Structure list

This chapter defines the structures exchanged to access the features listed in the next chapter.

### 2.5.1 Type definitions

In this chapter, the data types are defined bellow:

Type	Size in bytes
BYTE	1
USHORT	2
ULONG	4

Byte ordering is decided by machine architecture.

Use of "[n]" after a type indicates an array of n elements of the given type.

If n is not specified, this means that the number of elements is specified by another field in the structure.

## 2.5.2 PIN\_VERIFY

Byte offset	Field	Type	Description
0	bTimeOut	BYTE	timeout in seconds (00 means use default timeout)
1	bTimeOut2	BYTE	timeout in seconds after first key stroke
2	bmFormatString	BYTE	formatting options USB_CCID_PIN_FORMAT_xxx
3	bmPINBlockString	BYTE	bits 7-4 bit size of PIN length in APDU bits 3-0 PIN block size in bytes after justification and formatting
4	bmPINLengthFormat	BYTE	bits 7-5 RFU, bit 4 set if system units are bytes clear if system units are bits, bits 3-0 PIN length position in system units
5	wPINMaxExtraDigit	USHORT	XXYY, where XX is minimum PIN size in digits, YY is maximum
7	bEntryValidationCondition	BYTE	Conditions under which PIN entry should be considered complete
8	bNumberMessage	BYTE	Number of messages to display for PIN verification
9	wLangId	USHORT	Language for messages
11	bMsgIndex	BYTE	Message index (should be 00)
12	bTeoPrologue	BYTE[3]	T=1 I-block prologue field to use (fill with 00)
15	ulDataLength	ULONG	length of Data to be sent to the ICC
19	abData	BYTE[]	Data to send to the ICC

Detailed information about each structure element can be found in [4].

### 2.5.3 PIN\_MODIFY

Byte offset	Field	Type	Description
0	bTimeOut	BYTE	timeout in seconds (00 means use default timeout)
1	bTimeOut2	BYTE	timeout in seconds after first key stroke
2	bmFormatString	BYTE	formatting options USB_CCID_PIN_FORMAT_XXX
3	bmPINBlockString	BYTE	bits 7-4 bit size of PIN length in APDU bits 3-0 PIN block size in bytes after justification and formatting
4	bmPINLengthFormat	BYTE	bits 7-5 RFU, bit 4 set if system units are bytes clear if system units are bits, bits 3-0 PIN length position in system units
5	bInsertionOffsetOld	BYTE	Insertion position offset in bytes for the current PIN
6	bInsertionOffsetNew	BYTE	Insertion position offset in bytes for the new PIN
7	wPINMaxExtraDigit	USHORT	XXYY, where XX is minimum PIN size in digits, YY is maximum
9	bConfirmPIN	BYTE	Flags governing need for confirmation of new PIN
10	bEntryValidationCondition	BYTE	Conditions under which PIN entry should be considered complete
11	bNumberMessage	BYTE	Number of messages to display for PIN verification
12	wLangId	USHORT	Language for messages
14	bMsgIndex1	BYTE	Index of 1st prompting message
15	bMsgIndex2	BYTE	Index of 2 <sup>nd</sup> prompting message
16	bMsgIndex3	BYTE	Index of 3 <sup>rd</sup> prompting message
17	bTeoPrologue	BYTE 3]	T=1 I-block prologue field to use (fill with 00)
20	ulDataLength	ULONG	length of Data to be sent to the ICC
24	abData	BYTE[]	Data to send to the ICC

Detailed information about each structure element can be found in [4].



## 2.5.4 MCT\_UNIVERSAL

Byte offset	Field	Type	Description
0	SAD	BYTE	Source Address, see [2]
1	DAD	BYTE	Destination Address, see [2]
2	BufferLength	USHORT	Size in bytes of the following buffer
4	Buffer	BYTE[]	Buffer to send to the device

## 2.5.5 PIN\_PROPERTIES

Byte offset	Field	Type	Description
0	wLcdLayout	USHORT	display characteristics as defined in [4]
2	wLcdMaxCharacters	USHORT	Maximum number of characters on a single line
4	wLcdMaxLines	USHORT	Maximum number of lines that can be used
6	bEntryValidationCondition	BYTE	bitmap as defined in [4]
7	bTimeOut2	BYTE	0 = IFD does not distinguish bTimeOut from bTimeOut2 1 = IFD distinguishes bTimeOut from bTimeOut2

## 2.5.6 DISPLAY\_PROPERTIES

Byte offset	Field	Type	Description
0	wLcdMaxCharacters	USHORT	Maximum number of characters on a single line
2	wLcdMaxLines	USHORT	Maximum number of lines that can be used

## 2.5.7 SET\_SPE\_MESSAGE

Byte offset	Field	Type	Description
0	bApplicationId	BYTE[32]	Unique application ID
32	bMessageIndex	BYTE	Index of the message which should be set
33	wLangId	USHORT	Language ID for message
35	bMessageLength	BYTE	Length of message, in bytes
36	bMessage	BYTE[]	Message string in UTF-8

### 2.5.8 PIN\_VERIFY\_APP\_ID

Byte offset	Field	Type	Description
0	bApplicationId	BYTE [32]	unique application ID
32	bTimeOut	BYTE	timeout in seconds (00 means use default timeout)
33	bTimeOut2	BYTE	timeout in seconds after first key stroke
34	bmFormatString	BYTE	formatting options USB_CCID_PIN_FORMAT_XXX
35	bmPINBlockString	BYTE	bits 7-4 bit size of PIN length in APDU bits 3-0 PIN block size in bytes after justification and formatting
36	bmPINLengthFormat	BYTE	bits 7-5 RFU, bit 4 set if system units are bytes clear if system units are bits, bits 3-0 PIN length position in system units
37	wPINMaxExtraDigit	USHORT	XXYY, where XX is minimum PIN size in digits, YY is maximum
39	bEntryValidationCondition	BYTE	Conditions under which PIN entry should be considered complete
40	bNumberMessage	BYTE	Number of messages to display for PIN verification
41	wLangId	USHORT	Language for messages
43	bMsgIndex	BYTE	Message index (should be 00)
44	bTeoPrologue	BYTE[3]	T=1 I-block prologue field to use (fill with 00)
47	ulDataLength	ULONG	length of Data to be sent to the ICC
51	abData	BYTE[]	Data to send to the ICC

Detailed information about each structure element (except of bApplicationId and bimeout2) can be found in [4].

## 2.5.9 PIN\_MODIFY\_APP\_ID

Byte offset	Field	Type	Description
0	bApplicationId	BYTE[32]	unique application ID
32	bTimeOut	BYTE	timeout in seconds (00 means use default timeout)
33	bTimeOut2	BYTE	timeout in seconds after first key stroke
34	bmFormatString	BYTE	formatting options USB_CCID_PIN_FORMAT_XXX
35	bmPINBlockString	BYTE	bits 7-4 bit size of PIN length in APDU bits 3-0 PIN block size in bytes after justification and formatting
36	bmPINLengthFormat	BYTE	bits 7-5 RFU, bit 4 set if system units are bytes clear if system units are bits, bits 3-0 PIN length position in system units
37	bInsertionOffsetOld	BYTE	Insertion position offset in bytes for the current PIN
38	bInsertionOffsetNew	BYTE	Insertion position offset in bytes for the new PIN
39	wPINMaxExtraDigit	USHORT	XXYY, where XX is minimum PIN size in digits, YY is maximum
41	bConfirmPIN	BYTE	Flags governing need for confirmation of new PIN
42	bEntryValidationCondition	BYTE	Conditions under which PIN entry should be considered complete
43	bNumberMessage	BYTE	Number of messages to display for PIN verification
44	wLangId	USHORT	Language for messages
46	bMsgIndex1	BYTE	Index of 1st prompting message
47	bMsgIndex2	BYTE	Index of 2 <sup>nd</sup> prompting message
48	bMsgIndex3	BYTE	Index of 3 <sup>rd</sup> prompting message
49	bTeoPrologue	BYTE[3]	T=1 I-block prologue field to use (fill with 00)
52	ulDataLength	ULONG	length of Data to be sent to the ICC
56	abData	BYTE[]	Data to send to the ICC

Detailed information about each structure element (except of bApplicationId and bimeout2) can be found in [4].

### 2.5.10 WRITE\_DISPLAY

Byte offset	Field	Type	Description
0	wDisplayTime	USHORT	Display time in ms
2	bPosX	BYTE	Column (starting at 0)
3	bPosY	BYTE	Row (starting at 0)
4	wLangId	USHORT	Language ID of the message
6	bStringLength	BYTE	Length of message, in bytes
7	bString	BYTE[]	message string in UTF-8

bPosX and bPosY must within the boundaries specified by FEATURE\_IFD\_DISPLAY\_PROPERTIES.

### 2.5.11 GET\_KEY

Byte offset	Field	Type	Description
0	wWaitTime	USHORT	Time in seconds to wait for a key to be hit
2	bMode	BYTE	Display mode of key
3	bPosX	BYTE	Column (starting at 0)
4	bPosY	BYTE	Row (starting at 0)

Following values are possible for bMode :

0	Character of key is displayed
1	Asterisk (*) is displayed for each hit key
2	Nothing is displayed
All others	RFU

## **2.6 Feature list**

### **2.6.1 FEATURE\_VERIFY\_PIN\_START / FEATURE\_MODIFY\_PIN\_START**

FEATURE\_VERIFY\_PIN\_START uses the PIN\_VERIFY for the input buffer.  
FEATURE\_MODIFY\_PIN\_START uses the PIN\_MODIFY for the input buffer.

The IFD handler will return no data back to Control function.

Implementation Note for Windows:

In case of a parameter error of the passed structure, the ScardControl may return ERROR\_INVALID\_PARAMETER (0x57).

## 2.6.2 FEATURE\_GET\_KEY\_PRESSED

After the control code for FEATURE\_VERIFY\_PIN\_START or FEATURE\_MODIFY\_PIN\_START has been sent to the IFD handler, the control code for FEATURE\_GET\_KEY\_PRESSED can be used to determine if a key has been pressed. The IFD handler will send one byte back to the Control function according following table.

'0'– '9' button (valid keys)	0x2B
Cancel button	0x1B
Correction/Backspace button	0x08
Enter/Ok button	0x0d
Timeout finished SPE operation	0x0e
No key since last call	0x00
PIN_Operation_Aborted	0x40 (used e.g. for timeout indication, parameter error)
all keys cleared (by backspace)	0x0a (This value can be returned optionally)

Applications can use the returned information to display feedback to the user (e.g. pseudo display). This control code must be used in a polling mode.

Implementation notes for

1) Validation condition is equal to 'Timeout occurred':

The IFD handler may send 0x0d or optionally 0x0e . 0x0e can be used to give the calling application the possibility to distinguish which user operation finished the SPE operation if the validation condition is a combination of 'Timeout occurred' + 'Validation key pressed'.

2) Validation condition is equal to ' Max size reached'

The IFD handler must send for each valid key a 0x2B back to the calling control function. The Ok/Enter button must be ignored for the whole SPE operation. If the max PIN size is reached, the IFD handler must not send a 0x0d to indicate that the SPE operation has finished. The application itself knows when the maximum PIN size has been reached and therefore needs to indication.

### 2.6.3 FEATURE\_VERIFY\_PIN\_FINISH / FEATURE\_MODIFY\_PIN\_FINISH

The control code for the FEATURE\_VERIFY\_PIN\_FINISH or FEATURE\_MODIFY\_PIN\_FINISH must be used to retrieve the final result from the secure pin entry operation.

The IFD handler will return 2 bytes according to following table :

64 00	SPE operation timed out
64 01	SPE operation was cancelled by the 'Cancel' button
64 02 <sup>(1)</sup>	Modify PIN operation failed because two "new PIN" entries do not match
64 03	User entered too short or too long PIN regarding MIN/MAX PIN Length Note: as this error code is not known by CT-API implementations, it should be mapped to 64 01 on CT-API level.
6b 80	invalid parameter in passed structure
SW1 SW2	result from the card

<sup>1)</sup> 64 02 occurs if the user enters two different "new PIN's" during the Modify PIN operation. In that case, no Change PIN command has been sent to the smart card

## **2.6.4 FEATURE\_VERIFY\_PIN\_DIRECT / FEATURE\_MODIFY\_PIN\_DIRECT**

The control codes for these features use the same PIN structures which are already described in sections 2.5.2 and 2.5.3.

The main purposes for these control codes are:

- 1) class 2 readers without display for which no user feedback to the host is either required or supported.
- 2) class 2 readers with display which is used for SPE messages and user feedback.

The IFD handler will return the same responses as described for the FEATURE\_VERIFY\_PIN\_FINISH/ FEATURE\_MODIFY\_PIN\_FINISH.

## 2.6.5 FEATURE\_ABORT

This control code for FEATURE\_ABORT can be used to cancel any of the actions initiated for the following control codes:

- FEATURE\_VERIFY\_PIN\_START
- FEATURE\_MODIFY\_PIN\_START

The control codes for FEATURE\_VERIFY\_PIN\_DIRECT and FEATURE\_MODIFY\_PIN\_DIRECT will NOT require this abort.

This control code may be required for devices, which do not have a display, as end users may use a CANCEL button at the host application to cancel the SPE.

This control code will always succeed. Also, after this ABORT control code, there is no need for applications to use the control code for FEATURE\_VERIFY\_PIN\_FINISH and FEATURE\_MODIFY\_PIN\_FINISH, respectively, as the same 2 bytes as returned by the finish operation can be returned by this ABORT control code.

The IFD handler will return 2 bytes according to following table:

64 80	SPE operation was aborted by the a 'Cancel' operation at the host system
SW1 SW2	result from the card

## **2.6.6 FEATURE\_MCT\_READERDIRECT**

The control code for FEATURE\_MCT\_READERDIRECT can be used to transmit a command to the IFD.

The IFD handler will return a buffer containing data (data size can be null) and two status bytes SW1 SW2 according to table 12 of [1] and chapter 4.2 of [3].

## **2.6.7 FEATURE\_MCT\_UNIVERSAL**

This control code for FEATURE\_MCT\_UNIVERSAL can be used to transmit a command to the IFD or to any of the ICCs of the IFD.

Therefore a MCT\_UNIVERSAL structure is passed to the IFD handler. The SAD and DAD fields have to be filled with values according to table 6 of document [2]. The Buffer field contains the command APDU.

The IFD handler will return a MCT\_UNIVERSAL structure with SAD and DAD fields containing values concerning to table 7 of [2].

The buffer field will contain response data and two status bytes SW1 SW2 according to table 12 of [1] and chapter 4.2 of [3].

## 2.6.8 FEATURE\_IFD\_PIN\_PROPERTIES

This feature can be used – if supported by the IFD handler – to retrieve the properties of the IFD regarding PIN handling :

- 1) The IFD handler returns the size of a possible display as described in [4]
- 2) The IFD handler returns which entry validation conditions are supported as described in [4]
- 3) The IFD handler returns if the reader distinguishes between bTimeOut from bTimeOut2

The input parameter for this feature is a NULL pointer.

The output parameter is a pointer to a PIN\_PROPERTIES structure.

Note : wLcdLayout is also used to indicate if a display is present. If wLcdLayout = 0x0000, the IFD has no display.

If an IFD handler does not support this feature, an application must assume following default values:

wLcdLayout = 0x0000	no display present
bEntryValidationCondition = 0x07	timeout reached, max PIN size reached, validation key pressed
bTimeOut2 = 0x00	IFD does not distinguish bTimeOut from TimeOut2

## 2.6.9 FEATURE\_IFD\_DISPLAY\_PROPERTIES

The IFD handler returns the number of characters that can be displayed on a single line. If this is greater than the physical number of characters, the IFD is capable of scrolling messages that exceed the physical characteristics. The scrolling behavior is specific to the IFD.

The IFD handler returns the number of lines that can be used to display custom message strings. This number may be less than the physical number of lines (ex. if the IFD wants to reserve some space for hard coded messages – say, “Enter PIN”).

The input parameter for this feature is a NULL pointer.

The output parameter is a pointer to a DISPLAY\_PROPERTIES structure.

If an IFD handler does not support this feature, an application must assume following default values:

wLcdMaxCharacters = 0x0000 and wLcdMaxLines = 0x0000 : no display is present

## 2.6.10 FEATURE\_SET\_SPE\_MESSAGE

This feature can be used to define a message which should be displayed during any SPE operation by an IFD with display. After storing the SPE message in the IFD, the application just needs to properly set either bMsgIndex (FEATURE\_VERIFY\_PIN\_DIRECT) or bMsgIndex1, bMsgIndex2 and bMsgIndex3 (FEATURE\_MODIFY\_PIN\_DIRECT) to get the message displayed during any SPE operation.

The message is stored in the IFD until the next power on reset (e.g. reboot, plug off-plug in). Optionally IFDs may also store application dependent SPE messages permanently.

The application must provide following information in the SET\_SPE\_MESSAGE structure :

- bApplicationId :

For each application different SPE strings may be used. Otherwise applications will get conflicts if the same message index is used but the SPE message has a different meaning. bApplicationId should be unique and can completely be defined by any PC/SC application itself.

- wLanguageId:

This must correspond to an ANSI code page that should be used for any conversion. If this is set to zero, then the device may choose any code page which may result in loss of data.

- bMessageIndex:

Index of the SPE message

- bMessageLength

length of the SPE message, in bytes

- bMessage

buffer which contains SPE message in UTF-8

The IFD must provide for each application ID and for each language ID a storage for up to 254 messages. In case the IFD cannot store a message, the IFD handler must return response code Out\_Of\_Memory.

If the message index is not within the range of 0x00 – 0xFE, the IFD handler must return the response code Device\_Wrong\_Parameter. The value 0xFF is reserved for further purposes.

Applications can use FEATURE\_IFD\_PIN\_PROPERTIES to retrieve the display capabilities of the IFD.

If an application sets a message which is longer than `wLcdMaxCharacters`, following 2 options are allowed :

- a. The IFD returns `Device_Wrong_Parameter`
- b. The IFD displays as many characters as possible and cuts off the message string. In this case `Device_Success` must be returned.

An application can use the carriage return character (0x0d) to control which part of the message should be displayed in the first and in the second line.

It is not possible to set more than one message by a single call to the IFD. Each new message requires a separate call.

The IFD MAY convert `bMessage` into a native format using the `wLanguageId` before storing.

The default SPE messages, which do not depend on any application and which are IFD vendor specific, cannot be overwritten. Whenever an application uses 2.6.4 , these message are displayed.

### **2.6.11 FEATURE\_VERIFY\_PIN\_DIRECT\_APP\_ID / FEATURE\_MODIFY\_PIN\_DIRECT\_APP\_ID**

These features can be used as the features described in 2.6.4, but additionally an unique application ID will be passed to the IFD which is necessary to display the application specific SPE messages which have been set by 2.6.10.

## 2.6.12 FEATURE\_WRITE\_DISPLAY

This feature can be used to write any UTF-8 based message to the display if SPE is not active.

The input buffer is a pointer to a WRITE\_DISPLAY structure.

The output buffer is a NULL pointer.

The parameter display time has the following meaning :

wDisplayTime = 0	Message is displayed forever or until a new message is written again
wDisplayTime > 0	Message is displayed as long as specified. After the time has elapsed, a terminal specific message is displayed again (idle message)

### Warning:

There is a potential security hazard when supporting this feature; any string may be easily displayed by any PC/SC application, including a malware.

Combination of 2.6.12 and 2.6.13 features may lead to a malware requesting a false PIN request and retrieving the PIN code.

### 2.6.13 FEATURE\_GET\_KEY

This feature can be used to retrieve the value of a pressed key if SPE is not active. The GET\_KEY structure is used for the input buffer, the output buffer holds just one byte for the pressed key.

The application get following values for a pressed keys :

Key	Returned Value
0	0x30
1	0x31
2	0x32
3	0x33
4	0x34
5	0x35
6	0x36
7	0x37
8	0x38
9	0x39
*	0x2A
.	0x2E
Cancel	0x1b
Backspace	0x08
Menu	0x4D ('M')
OK	0x0d

Warning:

There is a potential security hazard when supporting this feature; the keys entered by using this GET\_KEY feature at the Secure PIN IFD may be easily grabbed by any PC/SC application, including a malware. Combination of 2.6.12 and 2.6.13 features may lead to a malware requesting a false PIN request and retrieving the PIN code.

### **3 Abbreviations**

IFD	Interface Device
MCT	Multifunctional Card Terminal
PIN	Personal Identification Number
SPE	Secure PIN Entry
TLV	Tag Length Value

## **4 References**

- [1] International technology – Identification cards – Integrated circuit(s) cards with contacts – Part 4: Interindustry commands for interchange; International Standard ISO/IEC 7816-4:1995(E)
- [2] Multifunctional Card Terminals, Part 3 - Application Independent Card Terminal Application Programming Interface for ICC Applications (CT-API 1.1); Deutsche Telekom AG / PZ Telesec, SIT Fraunhofer Institut für Sichere Telekooperation, TÜV Informationstechnik GmbH, TeleTrust Deutschland e.V.; 2002
- [3] Multifunctional Card Terminals, Part 4 - CT-BCS - Application Independent Card Terminal Basic Command Set; Deutsche Telekom AG / PZ Telesec, SIT Fraunhofer Institut für Sichere Telekooperation, TÜV Informationstechnik GmbH, TeleTrust Deutschland e.V.; 2002
- [4] USB Serial Bus Device Class Spec of USB Chip/Smart Card Interface Devices, Revision 1.00