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 2.01.05

September 2005

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

Contents

1 SYSTEM ARCHITECTURE 1

2 DEFINITION OF FEATURES AND CONTROL CODES 2

2.1 General Description 2

2.2 GET_FEATURE_REQUEST 2
2.2.1 Implementation Note for Windows 2

2.3 Definition of Features 3

2.4 Function Call 4
2.4.1 Implementation Note for Windows 4

2.5 Structure for PIN Verify 5
2.5.1 Implementation Note for Windows 5

2.6 Structure for PIN Modify 6
2.6.1 Implementation Note for Windows 6

2.7 MCT-Structures 7
2.7.1 Implementation Note for Windows 7

2.8 Structure for PIN Properties 8
2.8.1 Implementation Note for Windows 8

2.9 FEATURE_VERIFY_PIN_START / FEATURE_MODIFY_PIN_START 9
2.9.1 Implementation Note for Windows 9

2.10 FEATURE_GET_KEY_PRESSED 10

2.11 FEATURE_VERIFY_PIN_FINISH / FEATURE_MODIFY_PIN_FINISH 11

2.12 FEATURE_VERIFY_PIN_DIRECT / FEATURE_MODIFY_PIN_DIRECT 12

2.13 FEATURE_ABORT 13

2.14 FEATURE_MCT_READERDIRECT 14

2.15 FEATURE_MCT_UNIVERSAL 14

2.16 FEATURE_IFD_PIN_PROPERTIES 15

3 ABBREVIATIONS 16

4 REFERENCES 17

1 System Architecture

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

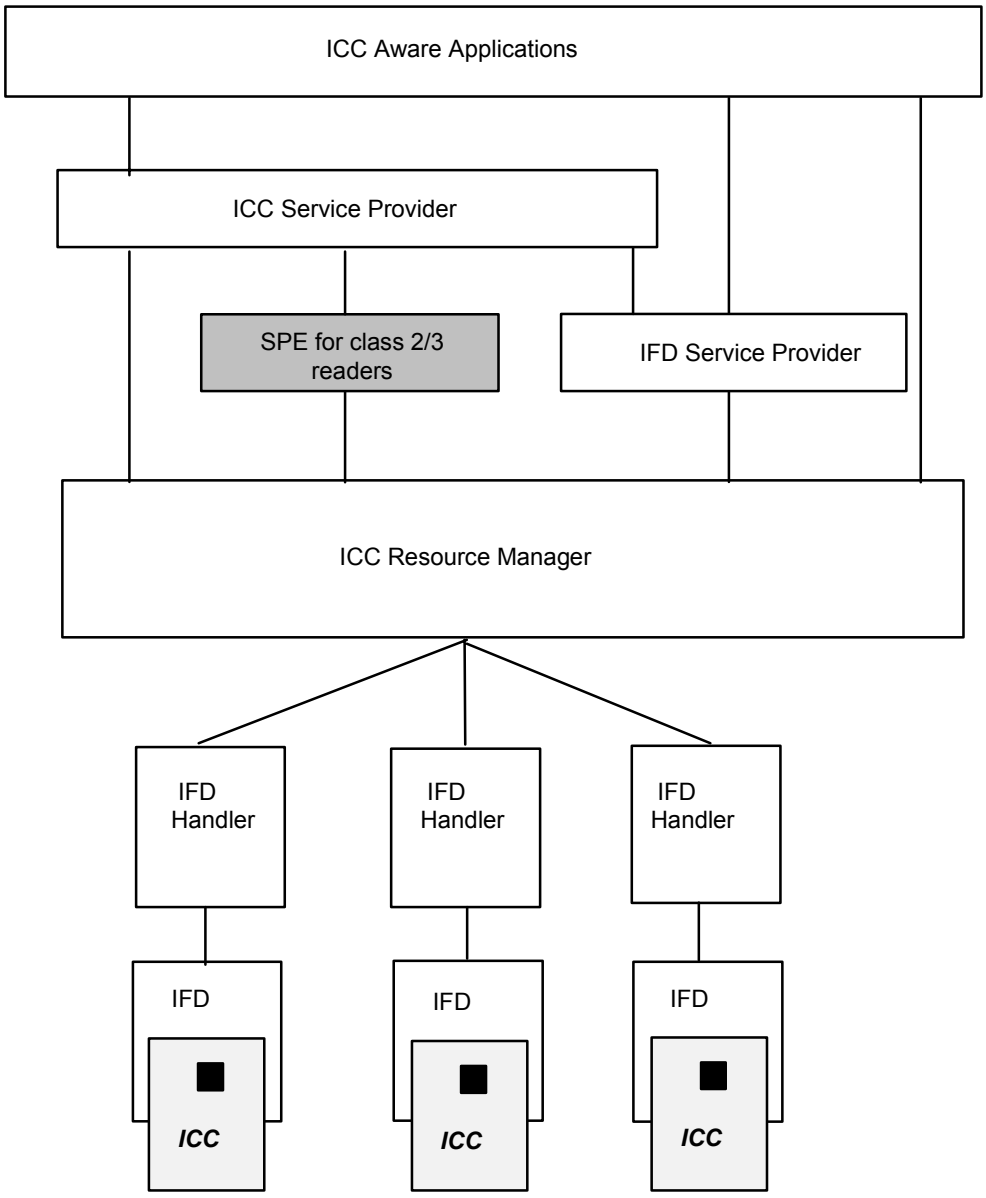

Figure 1- General Architecture

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.

2.2.1 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

```
#define FEATURE_VERIFY_PIN_START          0x01
#define FEATURE_VERIFY_PIN_FINISH         0x02
#define FEATURE_MODIFY_PIN_START          0x03
#define FEATURE_MODIFY_PIN_FINISH         0x04
#define FEATURE_GET_KEY_PRESSED           0x05
#define FEATURE_VERIFY_PIN_DIRECT         0x06
#define FEATURE_MODIFY_PIN_DIRECT         0x07
#define FEATURE_MCT_READERDIRECT          0x08
#define FEATURE_MCT_UNIVERSAL             0x09
#define FEATURE_IFD_PIN_PROP              0x0A
#define FEATURE_ABORT                     0x0B
```

The following tags 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_PROP	0x0A
FEATURE_ABORT	0x0B

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

01 04 XX XX XX XX 02 04 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

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 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
        )
```

2.4.1 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 for PIN Verify

```
typedef struct _PIN_VERIFY_STRUCTURE
{

BYTE        bTimeOut;                   //  timeout in seconds (00 means use default timeout)
BYTE        bTimeOut2                   // timeout in seconds after first key stroke
BYTE        bmFormatString;             //  formatting options
BYTE        bmPINBlockString;           //  bits 7-4 bit size of PIN length in APDU, bits 3-0 PIN
                                        // block  size  in bytes after justification and formatting
BYTE        bmPINLengthFormat;          //  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
USHORT      wPINMaxExtraDigit;          //  XXYY, where XX is minimum PIN size in digits,
                                        // YY is maximum
BYTE        bEntryValidationCondition;  //  Conditions under which PIN entry should be
                                        // considered complete
BYTE        bNumberMessage;             //  Number of messages to display for PIN verification
USHORT      wLangId;                    //  Language for messages
BYTE        bMsgIndex;                  //  Message index (should be 00)
BYTE        bTeoPrologue[3];            //  T=1 I-block prologue field to use (fill with 00)
ULONG       ulDataLength                // length of Data to be sent to the ICC
BYTE        abData[1];                  //  Data to send to the ICC

} PIN_VERIFY_STRUCTURE, *PPIN_VERIFY_STRUCTURE;
```

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

2.5.1 Implementation Note for Windows

Under Windows the definition of this structure must be enclosed between a #pragma pack(1) and #pragma pack() precompiler instruction to ensure that the application and the IFD handler use the same structure packing.

2.6 Structure for PIN Modify

```
typedef struct _PIN_MODIFY_STRUCTURE
 {
BYTE        bTimeOut;                   //  timeout in seconds (00 means use default timeout)
BYTE        bTimeOut2                   // timeout in seconds after first key stroke
BYTE        bmFormatString;             //  formatting options USB_CCID_PIN_FORMAT_xxx)
BYTE        bmPINBlockString;           //  bits 7-4 bit size of PIN length in APDU, bits 3-0 PIN
                                        // block size in bytes after justification and formatting
BYTE        bmPINLengthFormat;          //  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

                                        // bits, bits 3-0 PIN length position in system units
BYTE        bInsertionOffsetOld;        //  Insertion position offset in bytes for the current PIN
BYTE        bInsertionOffsetNew;        //  Insertion position offset in bytes for the new PIN
USHORT      wPINMaxExtraDigit;          //  XXYY, where XX is minimum PIN size in digits,
                                        // YY is maximum
BYTE        bConfirmPIN;                //  Flags governing need for confirmation of new PIN
BYTE        bEntryValidationCondition;  //  Conditions under which PIN entry should be
                                        // considered complete
BYTE        bNumberMessage;             //  Number of messages to display for PIN verification
USHORT      wLangId;                    //  Language for messages
BYTE        bMsgIndex1;                 //  Index of 1st prompting message
BYTE        bMsgIndex2;                 //  Index of 2d prompting message
BYTE        bMsgIndex3;                 //  Index of 3d prompting message
BYTE        bTeoPrologue[3];            // T=1 I-block prologue field to use (fill with 00)
ULONG       ulDataLength                // length of Data to be sent to the ICC
BYTE        abData[1];                  // Data to send to the ICC
} PIN_MODIFY_STRUCTURE , *PPIN_MODIFY_STRUCTURE ;
```

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

2.6.1 Implementation Note for Windows

Under Windows the definition of this structure must be enclosed between a #pragma pack(1) and #pragma pack() precompiler instruction to be sure that the application and the IFD handler use the same structure packing.

2.7 MCT-Structures

```
typdef struct _MCTUniversal
{
        BYTE SAD;
        BYTE DAD;
        USHORT BufferLength;
        BYTE buffer[1];
      } MCTUniversal_t, *PMCTUniversal_t;
```

2.7.1 Implementation Note for Windows

Under Windows the definition of this structure must be enclosed between a #pragma pack(1) and #pragma pack() precompiler instruction to be sure that the application and the IFD handler use the same structure packing.

2.8 Structure for PIN Properties

```
typedef struct _PIN_PROPERTIES_STRUCTURE
{
  USHORT        wLcdLayout;                 // display characteristics as defined in [4]
  BYTE          bEntryValidationCondition;  // bitmap as defined in [4]
  BYTE          bTimeOut2;                  // 0 = IFD does not distinguish bTimeOut from
bTimeOut2,
                                            // 1 =  IFD distinguishes bTimeOut from bTimeOut2,
} PIN_PROPERTIES_STRUCTURE, *PPIN_PROPERTIES_STRUCTURE;
```

2.8.1 Implementation Note for Windows

Under Windows the definition of this structure must be enclosed between a #pragma pack(1) and #pragma pack() precompiler instruction to be sure that the application and the IFD handler use the same structure packing.

2.9 FEATURE_VERIFY_PIN_START / FEATURE_MODIFY_PIN_START

FEATURE_VERIFY_PIN_START uses the PIN_VERIFY_STRUCTURE for the input buffer.
FEATURE_MODIFY_PIN_START uses the PIN_MODIFY_STRUCTURE for the input buffer.

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

2.9.1 Implementation Note for Windows

In case of a parameter error of the passed structure, the ScardControl may return ERROR_INVALID_PARAMETER (0x57).

2.10 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
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.

2.11 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 (1)	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

1) 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

2.12 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 and 2.6.

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.

2.13 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

2.14 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].

2.15 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 MCTUniversal struct 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 MCTUniversal struct 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].

2.16 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

3 Abbreviations

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

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

