EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

WORKSHOP AGREEMENT

CWA 14050-25

November 2000

ICS 35.200; 35.240.15

Extensions for Financial Services (XFS) interface specification - Release 3.0 - Part 25: Identification Card Device Class Interface - PC/SC Integration Guidelines

Ref. No CWA 14050-25:2000 E

Rue de Stassart, 36 • B-1050 Bruxelles
Tel : +32 2 550 08 11 • Fax : +32 2 550 08 19

Table of Contents

Foreword

This CWA is revision 3.0 of the XFS interface specification.

The move from an XFS 2.0 specification (CWA 13449) to a 3.0 specification has been prompted by a series of factors.

Initially, there has been a technical imperative to extend the scope of the existing specification of the XFS Manager to include new devices, such as the Card Embossing Unit.

Similarly, there has also been pressure, through implementation experience and the advance of the Microsoft technology, to extend the functionality and capabilities of the existing devices covered by the specification.

Finally, it is also clear that our customers and the market are asking for an update to a specification, which is now over 2 years old. Increasing market acceptance and the need to meet this demand is driving the Workshop towards this release.

The clear direction of the CEN/ISSS XFS Workshop, therefore, is the delivery of a new Release 3.0 specification based on a C API. It will be delivered with the promise of the protection of technical investment for existing applications and the design to safeguard future developments.

The CEN/ISSS XFS Workshop gathers suppliers as well as banks and other financial service companies. A list of companies participating in this Workshop and in support of this CWA is available from the CEN/ISSS Secretariat.

This CWA was formally approved by the XFS Workshop meeting on 2000-10-18. The specification is continuously reviewed and commented in the CEN/ISSS Workshop on XFS. It is therefore expected that an update of the specification will be published in due time as a CWA, superseding this revision 3.0.

The CWA is published as a multi-part document, consisting of:

Part 1: Application Programming Interface (API) - Service Provider Interface (SPI); Programmer's Reference

Part 2: Service Classes Definition; Programmer's Reference

Part 3: Printer Device Class Interface - Programmer's Reference

Part 4: Identification Card Device Class Interface - Programmer's Reference

Part 5: Cash Dispenser Device Class Interface - Programmer's Reference

Part 6: PIN Keypad Device Class Interface - Programmer's Reference

Part 7: Check Reader/Scanner Device Class Interface - Programmer's Reference

Part 8: Depository Device Class Interface - Programmer's Reference

Part 9: Text Terminal Unit Device Class Interface - Programmer's Reference

Part 10: Sensors and Indicators Unit Device Class Interface - Programmer's Reference

Part 11: Vendor Dependent Mode Device Class Interface - Programmer's Reference

Part 12: Camera Device Class Interface - Programmer's Reference

Part 13: Alarm Device Class Interface - Programmer's Reference

Part 14: Card Embossing Unit Class Interface - Programmer's Reference

Part 15: Cash In Module Device Class Interface- Programmer's Reference

Part 16: Application Programming Interface (API) - Service Provider Interface (SPI) - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 17: Printer Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 18: Identification Card Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 19: Cash Dispenser Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 20: PIN Keypad Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 21: Depository Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 22: Text Terminal Unit Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 23: Sensors and Indicators Unit Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 24: Camera Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 25: Identification Card Device Class Interface - PC/SC Integration Guidelines

In addition to these Programmer's Reference specifications, the reader of this CWA is also referred to a complementary document, called Release Notes. The Release Notes contain clarifications and explanations on the CWA specifications, which are not requiring functional changes. The current version of the Release Notes is available online from http://www.cenorm.be/isss/Workshop/XFS.

Revision History:

3.00	October 18, 2000	First edition

1. Introduction

1.1 Background to Release 3.0

The CEN XFS Workshop is a continuation of the Banking Solution Vendors Council workshop and maintains a technical commitment to the Win 32 API. However, the XFS Workshop has extended the franchise of multi vendor software by encouraging the participation of both banks and vendors to take part in the deliberations of the creation of an industry standard. This move towards opening the participation beyond the BSVC's original membership has been very succesful with a current membership level of more than 20 companies.

The fundamental aims of the XFS Workshop are to promote a clear and unambiguous specification for both service providers and application developers. This has been achieved to date by sub groups working electronically and quarterly meetings.

The move from an XFS 2.0 specification to a 3.0 specification has been prompted by a series of factors. Initially, there has been a technical imperative to extend the scope of the existing specification of the XFS Manager to include new devices, such as the Card Embossing Unit.

Similarly, there has also been pressure, through implementation experience and the advance of the Microsoft technology, to extend the functionality and capabilities of the existing devices covered by the specification.

Finally, it is also clear that our customers and the market are asking for an update to a specification, which is now over 2 years old. Increasing market acceptance and the need to meet this demand is driving the Workshop towards this release.

The clear direction of the XFS Workshop, therefore, is the delivery of a new Release 3.0 specification based on a C API. It will be delivered with the promise of the protection of technical investment for existing applications and the design to safeguard future developments.

1.2 PC/SC Background

As part of the XFS specification, the IDC (Identification Card) Device Class includes the following support for ICC (Integrated Circuit Card, also referred as chip) management, in respect to ISO 7816 (contacted) and ISO 10536 (contactless) international standards:

- retrieve ATR (Answer To Reset), when contacting the chip through a WFS_CMD_IDC_READ_RAW_DATA command;
- communicate with the chip, through WFS_CMD_IDC_CHIP_IO commands;
- handle power management, through the WFS_CMD_IDC_CHIP_POWER command.

Although these mechanisms provide independence to the channel used to exchange data with the chip, through the card reader, it does not provide independence to a given subset of capabilities provided by software embedded on the chip, which may offer the same level of functionality, but may be implemented in different ways for chips manufactured by different vendors.

The smart card subsystem provided in the Win32® environment follows the specifications published by the PC/SC Workgroup, including the following components of interest:

- a resource manager providing access to readers and smart cards,
- several COM-based smart card service providers, each providing access to a subset of capabilities for a chip.

The PC/SC (Personal Computer / Smart Card) Workgroup was formed in May 1996 in partnership with major PC and smart card companies. The main focus of the workgroup has been to develop specifications that ensure interoperability among smart cards, smart card readers, and computers made by different manufacturers. Version 1.0 of these specifications were released in December 1997.

This document aims to describe the relation between XFS and PC/SC and to provide guidelines for integration of PC/SC compliant readers in the XFS subsystem, ensuring compatibility with future extensions to the IDC Device Class.

1.3 References

This document is intended to cover in detail the relation between XFS and PC/SC. However, many details about PC/SC and smart cards are intentionally omitted for the interest of brevity, and this document often assumes background knowledge on these topics. The following documents may be of interest:

1.3.1 XFS specifications

A.1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer’s Reference Revision 3.0, October 18, 2000
A.2. XFS Identification Card Unit Device Class Interface (IDC), Programmer’s Reference Revision 3.00, October 18, 2000

1.3.2 PC/SC Workgroup specifications

Interoperability Specification for ICCs and Personal Computer Systems, version 1.0, December 1997
These specifications are available on the Web at: http://www.pcscworkgroup.com

B.1. Part 1. Introduction and Architecture Overview
B.2. Part 2. Interface Requirements for Compatible IC Cards and Interface Devices
B.3. Part 3. Requirements for PC-Connected Interface Devices
B.4. Part 4. IFD Design Considerations and Reference Design Information
B.5. Part 5. ICC Resource Manager Definition
B.6. Part 6. ICC Service Provider Interface Definition
B.7. Part 7. Application Domain/Developer Design Considerations
B.8. Part 8. Recommendation for Implementation of Security and Privacy ICC Devices

1.3.3 International standards

C.1. ISO/IEC 7816-4 Identification Cards - Integrated Circuit(s) cards with contacts - Part 4: Interindustry commands for interchange
C.2. ISO/IEC 7816-5 Identification Cards - Integrated Circuit(s) cards with contacts - Part 5: Numbering system and registration procedure for application identifiers

1.3.4 Microsoft documentation

D.1. MSDN Library, Platform SDK, Base Services, Smart Card
D.2. MSDN Library, DDK Documentation, Smart Card DDK
D.3. Smart Cards for Windows, on the Web at: http://www.microsoft.com/windowsce/smartcard

1.4 Glossary

APDU	Application Protocol Data Unit.
ATR	Answer To Reset. The transmission sent by an ICC to the reader in response to a reset condition.
CLSID	Class Identifier. A globally unique identifier (GUID) associated with a COM class object. If a class object will be used to create more than one instance of an object, the associated server application should register its CLSID in the system registry so that clients can locate and load the executable code associated with the object(s).
CHV	Card Holder Verification.
COM	Component Object Model. The OLE object-oriented programming model that defines how objects interact within a single process or between processes. In COM, clients have access to an object through interfaces implemented on the object. *See also* Interface.
COM Object	An object that conforms to the OLE Component Object Model (COM). A COM object is an instance of an object definition, which specifies the object's data and one or more implementations of interfaces on the object. Clients interact with a COM object only through its interfaces. *See also* Component Object Model and Interface.
Cold Reset	The reset of an ICC that occurs when the supply voltage and other signals to the ICC are raised from the inactive state and the reset signal is applied.
EMV	Europay, MasterCard, and Visa.
GUID	Globally Unique Identifier. 128-bit integer value, used to assign world-wide unique identifiers for COM interfaces and CoClasses.
ICC	Integrated Circuit Card. Plastic card containing an integrated circuit (chip), which is compatible with ISO 7816.
IDC	Identification Card.
IFD	Interface Device. A terminal, communication device, or machine to which the integrated circuit card is electrically connected during operation. As used in PC/SC specification, refers to a PC peripheral device that supports bi-directional I/O to an ISO 7816 standard ICC.
Interface	A group of semantically related functions that provide access to a COM object. Each interface defines a contract that allows objects to interact according to the Component Object Model (COM). *See also* Component Object Model and COM object.
Mute	State in which chip is unresponsive.
PC/SC	Personal Computer / Smart Card.
Warm Reset	The reset of an ICC that occurs when the supply voltage and the clock lines are maintained in their active state and the reset signal is applied.

2. PC/SC Overview

This chapter gives a high-level overview of PC/SC architecture:

- general concepts,
- how card readers and smart card are exposed to the application,
- components of interest.

The description given here is not a replacement for existing PC/SC specifications. The reader may refer to the following documentation for details :

- [B.1] and next parts for a full description,
- [D.1] for implementation in the smart card subsystem.

2.1 Architecture

The software and hardware components of PC/SC are linked as follows:

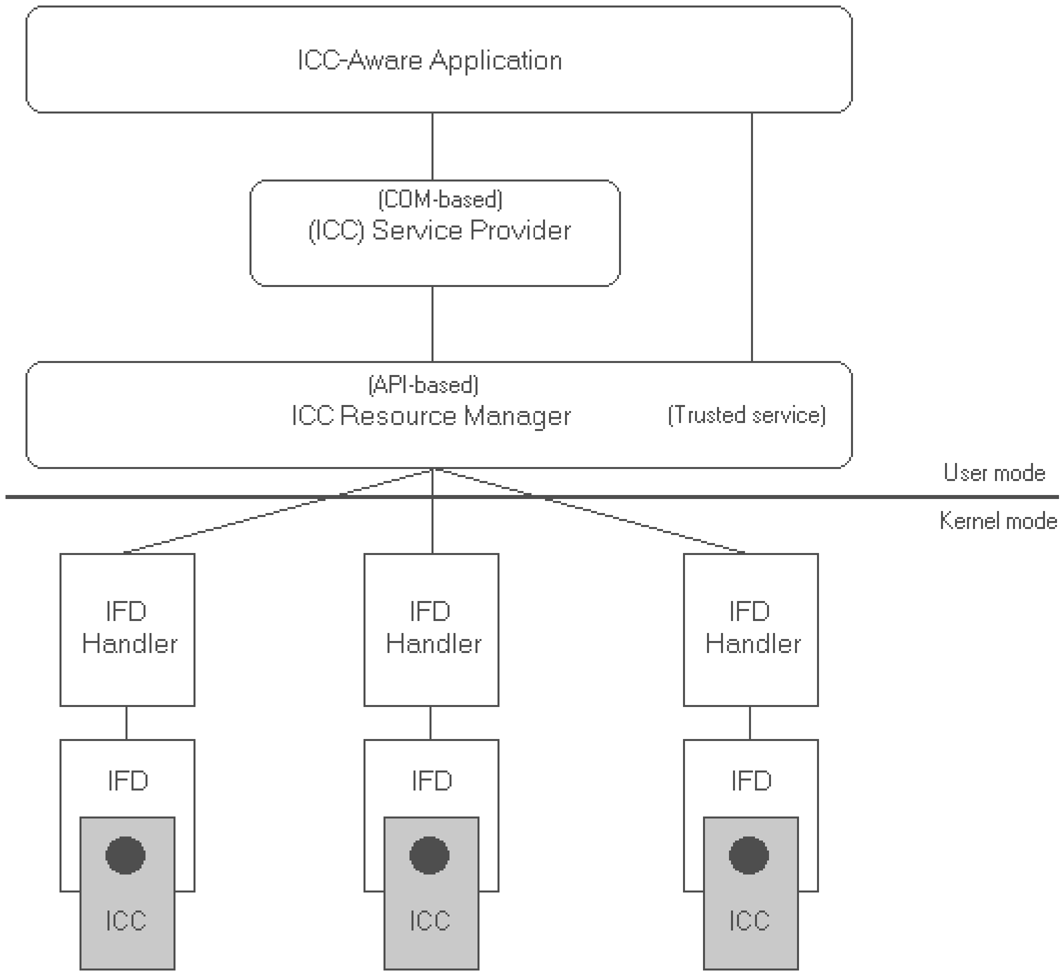

Figure 2.1 — PC/SC Architecture

The elements that comprised the architecture are defined in the next sections.

2.1.1 Integrated Circuit Card (ICC)

The ICC (commonly called « smart card ») is a credit-card sized plastic case with an embedded microprocessor chip. This specification specifically deals with contact-type ICCs as defined by ISO/IEC 7816 standard.

2.1.2 Interface Device (IFD)

The IFD (commonly called « smart card reader ») is the physical interface device through which an ICC communicates with a PC. The IFD establishes a set of electrical connections with the embedded microprocessor of an ICC through the electrical contacts on the surface of the ICC.

A compliant IFD will conform to the ISO/IEC 7816-1, 7816-2 and 7816-3 standards.

2.1.3 Interface Device Handler

The IFD handler encompasses the PC software necessary to map native capabilities of the IFD to the IFD handler interface defined in [B.3]. In the smart card subsystem, it runs in kernel mode (see [D.2] for details).

2.1.4 ICC Resource Manager

The ICC Resource Manager (referred as *Smart Card Resource Manager* in [D.1]) is a key component of the PC/SC Workgroup's architecture. It is responsible for managing the other ICC-relevant resources within the system and for supporting controlled access IFDs and, through them, individual ICCs. The ICC Resource Manager is assumed to be a system-level component of the architecture: in the smart card subsystem, it runs as a trusted service in user mode.

To manage ICCs and IFDs, the Resource Manager performs the following tasks:
- identification and tracking of resources (a database of known cards and readers is maintained)
- allocation of readers and resources across multiple applications,
- support of transaction primitives for accessing services available on a given ICC.

The ICC Resource Manager can be accessed directly through the Resource Manager API, or indirectly through any ICC Service Provider.

2.1.5 Service Provider

The Service Provider is responsible for encapsulating functionality exposed by a specific ICC and making it accessible through high-level programming interfaces. Access is provided through COM-based interfaces within the smart card subsystem.

The PC/SC specification differentiates between ICC Service Providers and Cryptographic Service Providers, to deal with international import/export laws for cryptographic devices. [D.1, Smart Card Concepts, Relation to Other Services] shows relation between both.

2.1.6 ICC Service Provider

The ICC Service Provider (ICCSP) encapsulates functionality exposed by a specific ICC and make them available through high level programming interfaces. Access is provided through COM-based interfaces within the smart card subsystem.

[B.5, §3.2], [B.6], [B.7, §2.1] and [D.1, Smart Card Service Providers] define the following types of ICC Service Providers:

1) Base ICCSPs, providing basic smart cards capabilities, such as:

INTERFACE	DESCRIPTION
ISCard	open and manage connection to a smart card, implement transactions
ISCardCmd	maintain a command APDU and reply APDU
ISCardISO7816	build an ISO7816-4 APDU
ISCardDatabase	query the smart card database (not a true ICCSP, provide access to the database of readers and smart cards, through Resource Manager)

The Base ICCSPs are provided by the smart card subsystem.

2) Vendor ICCSPs:
These ICCSPs expose services specific to a given ICC.

3) Layered ICCSPs:
These ICCSPs are developed to fit ICC-aware application needs, and may be build upon others ICCSPs. Domain specific interfaces may be added to these ICCSPs.

Optional interfaces are defined for commonly implemented ICC capabilities, as defined in [B.6, §3.4] and [D.1, Smart Card Service Providers, Vendor Wrapper Service Provider]. If an ICC support these capabilities, Vendor ICCSPs and Layered ICCSPs should implement them.

These interfaces are typically implemented by Vendor ICCSPs (with the same GUID as defined in [B.6, Appendix A]), and aggregated in Layered ICCSPs built upon them:

INTERFACE	DESCRIPTION
ISCardAuth	Authentication services (application, smart card, user authentication)
ISCardFileAccess	Card-based file system services
ISCardVerify	Verification services (set CHV code and verify the user)

2.2 XFS IDC Capabilities not directly exposed by Smart Card subsystem

Since PC/SC architecture was designed to manage smart cards and smart cards readers connected to a PC, there are subsets of capabilities, which may be supported by IDC readers (IFDs) and exposed through XFS IDC Device Class, that are not directly exposed by the smart card subsystem.

As defined in [B.3, §3.2.4] and [D.1, Smart Card Resource Manager API], these capabilities may be managed directly using the following ICC Resource Manager services:

FUNCTION	DESCRIPTION
SCardControl ( )	Provide direct control of the reader
SCardGetAttrib ( )	Get reader attribute (doesn’t affect the state of IFD or ICC)
SCardSetAttrib ( )	Set reader attribute (doesn’t affect the state of IFD or ICC)

Such capabilities are described in the next sections.

2.2.1 Motorized Card Readers

- Status: (see [A.2, §3.1] fwMedia)
  No status information is reported if the card is jammed into the reader.

- Capabilities: (see [A.2, §3.2])
  The following capabilities cannot be retrieved from the device:

CAPABILITY	DESCRIPTION
fwPowerOnOption	specifies the power-on capabilities of the device hardware
fwPowerOffOption	specifies the power-off capabilities of the device hardware
bFluxSensorProgrammable	specifies whether the Flux Sensor on the card unit is programmable
bReadWriteAccessFollowingEject	specifies whether a card may be read or written after having been pushed to the exit slot with an eject command

- Retain Bin management:
  Although the PC/SC specification defines a command to retain the card (CONFISCATE), it is not supported within the smart card subsystem available on Win32 environment.

2.2.2 Magnetic tracks

- Capabilities: (see [A.2, §3.2])
  The following capabilities cannot be retrieved from the device:

CAPABILITY	**DESCRIPTION**
fwReadTracks	specifies the tracks that can be read by the card unit
fwWriteTracks	specifies the tracks that can be written by the card unit
fwWriteMode	specifies the write capabilities (loco / hico magnetic stripes)

IFD Control commands should be available to read and write a given ISO track.

2.2.3 Associated Security Module

- Status: (see [A.2, §3.1] fwSecurity)
  The state of the security unit cannot be retrieved.

- Commands: (see [A.2, §4.6], WFS_CMD_IDC_SETKEY)
  The DES key that is necessary for operating a CIM86 module may be set by a new Control command, although this is not always true (depending on underlying architecture of the hardware device).

3. Architectural Overview

This chapter gives a comprehensive overview of the proposed model allowing integration of PC/SC in XFS:

- key components,
- interfaces which need to be specified,
- how this model may be extended when integrating new ICC capabilities in XFS,
- how existing PC/SC compliant smart cards readers and PC/SC compliant smart cards may be integrated in this architecture.

3.1 Generic Model

The generic model for integration of PC/SC in XFS is given below:

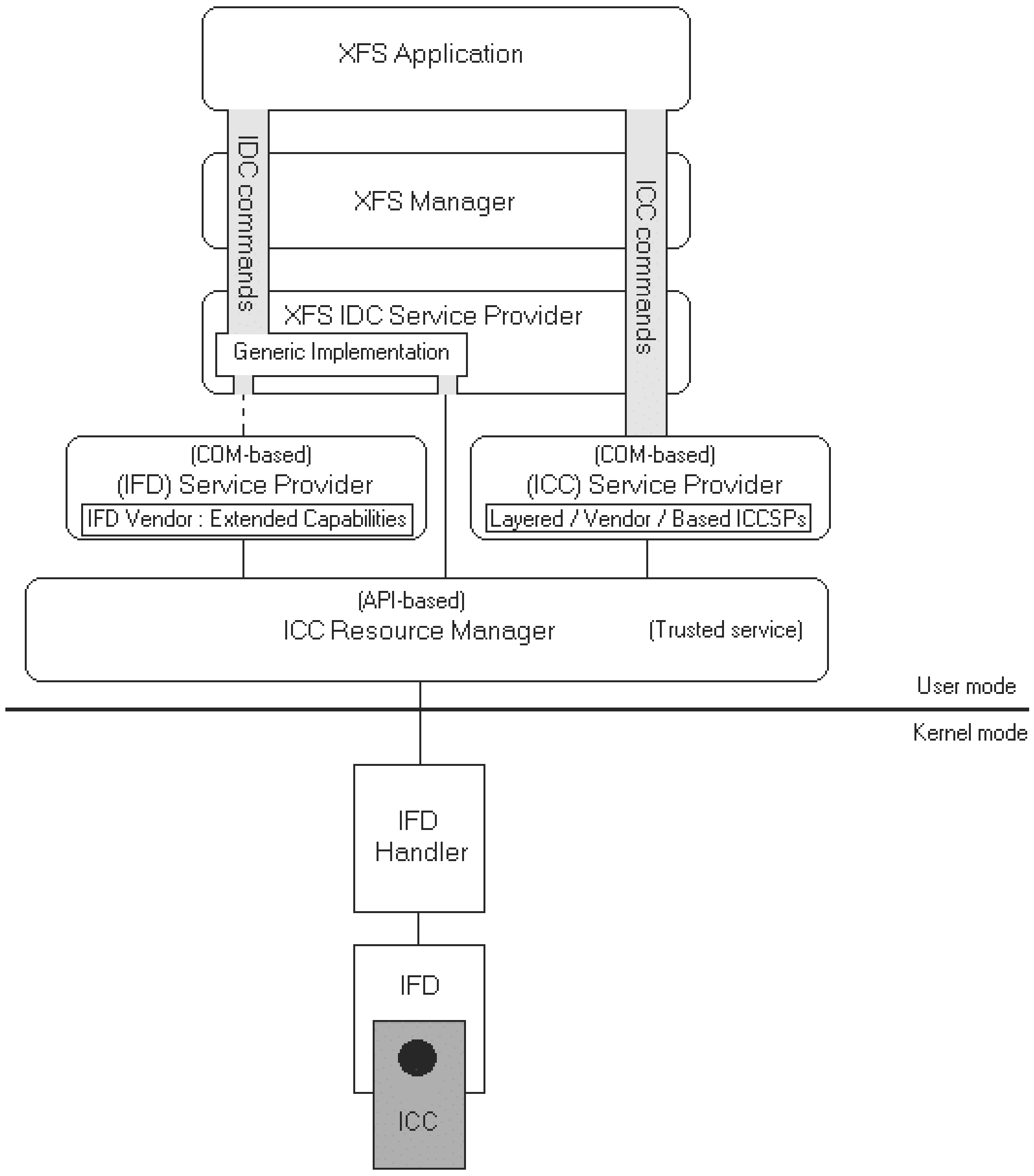

Figure 3.1 — Generic model: Integration of PC/SC in XFS

The key components of this architecture are described in the next sections.

3.1.1 IFD Service Provider

In the following, we will refer the capabilities defined in §2.2 as « extended capabilities ».

If an IFD provides a subset (or all) of these extended capabilities, and the IFD handler supports new Control commands and attributes to handle them, an IFD Service Provider must be provided, to allow the XFS IDC Service Provider to manage these extended capabilities.

If the IFD doesn't support any extended capability, there is no need to provide this IFD Service Provider.

The IFD Service Provider is implemented as a COM-based object whose CLSID is known by the IDC Service Provider, through a vendor-dependent configuration parameter (usually in NT registry database).

The IFD Service Provider implements at least one COM-based interface, IXFSIFDManage, which allows to give the card reader handle of the IFD handler to the object.

It may implements one or more of the following interfaces, depending on the supported extended capabilities:

INTERFACE	**DESCRIPTION**
IXFSIFDMotorized	Motorized card readers capabilities, as defined in §2.2.1
IXFSIFDMagneticTracks	Magnetic tracks capabilities, as defined in §2.2.2
IXFSIFDSecurityModule	Associated Security Module capabilities, as defined in §2.2.3

3.1.2 ICC Service Provider

The ICC Service Provider implements the COM-based interfaces that provide the ICC services exposed in IDC SPI commands.

When a new set of ICC commands is defined as part of the IDC Device Class, a corresponding COM-based interface is defined, with its associated GUID. This interface has one method for each new command, with the same input/output parameters. This allow the IDC Service Provider to act as a wrapper between the new IDC SPI commands and the COM-based interface that implements them.

These interfaces may be provided by :
- Base ICCSPs (e.g. ISCardISO7816),
- Vendor ICCSPs, implementing defined interfaces for common ICC services (e.g. ISCardFileAccess),
- Layered ICCSPs, developed specifically for a given ICC service (e.g. electronic purse), and aggregating Vendor ICCSPs to manage the target ICC.

3.1.3 XFS IDC Service Provider generic implementation

The services provided by the smart card subsystem and the optional IFD Service Provider allow to define a generic implementation for IDC SPI (queries and commands), regardless of the target IFD.

3.2 New Interfaces

As part of this generic model, the following interfaces need to be specified :

- IFD Service Provider COM-based interfaces, as defined in §3.1.1.

These interfaces will be specified in § 4.

3.3 Integration of card readers

The proposed architecture allow to add support for a card reader by designing its IFD handler rather than its XFS IDC Service Provider.

Existing IFD handlers, with no extended capabilities, are immediately supported (PC/SC compliant card readers).

If extended capabilities are provided by the IFD, the IFD handler should be designed with new Control commands and attributes (whose values are vendor-dependent), the associated IFD Service Provider implementing required interfaces to manage these extensions to the IFD handler. This IFD Service Provider provides interfaces which are not vendor-dependent.

If the IFD handler is provided, all smart cards with an ICCSP implementing the required interfaces are supported.

3.4 Integration of smart cards

If a subset of new ICC services provided as part of XFS IDC Device Class is based on base interfaces for commonly used ICC services (e.g. ISCardFileAccess), all smart cards with an ICCSP implementing this interface are supported.

If this subset of new ICC services is associated with a newly defined COM-based interface (e.g. electronic purse), and the capabilities of the target smart card match the required interface, a Layered ICCSP has to be developed to implement this interface, but it’s not an issue for the XFS vendor. This may be done by the smart card manufacturer, the application developer, or third parties, using the Base ICCSPs or an existing Vendor ICCSP.

4. IFD Service Provider COM-based interfaces

PC/SC compliant smart cards are accessed through PC/SC compliant smart card readers (IFDs), as part of the smart card subsystem.

Although implementation of IDC Service Provider is vendor-dependent, this section aims to propose guidelines for integration of PC/SC compliant smart card readers in the XFS subsystem, with new interfaces to allow integration of readers having extended capabilities available in XFS, which are not supported by PC/SC.

4.1 IFD Service Provider

As exposed in § 3.1.1, if an IFD provides a subset (or all) of extended capabilities, and the IFD handler supports new *Control* commands to handle them, an IFD Service Provider must be provided, to allow the XFS IDC Service Provider to manage these extended capabilities.

If the IFD doesn’t support any extended capability, there is no need to provide this IFD Service Provider.

Note: This IFD Service Provider is not a service provider in XFS terminology (like IDC). It may be viewed as a set of services provided by the IFD handler to allow the XFS IDC Service Provider to manage the full capabilities of the reader. It’s better to think of it as a PC/SC service provider, since it communicates with the IFD handler through the ICC Resource Manager, as part of the smart card subsystem.

The IFD Service Provider is implemented as a COM-based object whose CLSID is known by the IDC Service Provider, through a vendor-dependent configuration parameter (usually in NT registry database).

The IFD Service Provider implements one or more of the following interfaces, depending on the supported extended capabilities:

INTERFACE	**DESCRIPTION**
IXFSIFDManage	Required, to give the card reader handle of the IFD handler to the object.
IXFSIFDMotorized	Motorized card readers capabilities, as defined in §2.2.1.
IXFSIFDMagneticTracks	Magnetic tracks capabilities, as defined in §2.2.2.
IXFSIFDSecurityModule	Associated Security Module capabilities, as defined in §2.2.3.

The next sections define these interfaces.

Defining an IFD Service Provider by means of these interfaces allows the XFS IDC Service Provider to manage a PC/SC compliant reader in a generic manner, the vendor-dependent operations being part of the implementation of these interfaces in the IFD Service Provider.

4.2 IXFSIFDManage

This interface must be implemented by any IFD Service Provider. It contains one single method, called by the IDC Service Provider to provide the handle of the target card reader.

```
GUID: {2F92D5F6-69D3-11D3-930B-0090272C227C}
```

4.2.1 Properties

None.

4.2.2 Methods

```
HRESULT      AttachByHandle (
       IN          HSCARD         hCard               | Handle to the reader,
                                                      | Used to communicate with the
                                                      | IFD handler.
             );
```

4.3 IXFSIFDMotorized

This interface is provided to manage motorized card readers capabilities, as defined in § 2.2.1.

```
GUID: {2F92D5F7-69D3-11D3-930B-0090272C227C}
```

4.3.1 Properties

NAME	TYPE	ACCESS	DESCRIPTION
MediaStatus	Enum	RO	Extended status, to support Media Jammed. Values: see WFSIDCSTATUS.fwMedia
PowerOnOption	Enum	RO	Power-on capabilities. Values: see WFSIDCCAPS.fwPowerOnOption
PowerOffOption	Enum	RO	Power-off capabilities. Values: see WFSIDCCAPS.fwPowerOffOption
FluxSensorProgrammable	Boolean	RO	Specifies whether the Flux Sensor on the card unit is programmable.
FluxSensorActive	Boolean	RW	Specifies whether the Flux Sensor on the card unit is active.
ReadWriteAccessFollowingEject	Boolean	RO	Specifies whether a card may be read or written after having been pushed to the exit slot with an eject command.
RetainBinCapacity	WORD	RO	Maximum number of cards the retained bin can hold.
RetainBinCount	WORD	RW	Number of cards retained.

4.3.2 Methods

```
HRESULT      GetLastPowerAction (
       OUT        Enum *          pwAction,            Last action performed by
                                                       either the automatic power
                                                       on/off action of the device.
                                                       Values: see
                                                       WFSIDCCARDACT.wAction

       OUT        Enum *          pwPosition           Position of card before
                                                       retained or ejected.
                                                       Values: see
                                                       WFSIDCCARDACT.wPosition
             );

HRESULT RetainCard (
       OUT        Enum *          pwStatus             Status of operation.
                                                       Values: see error codes of
                                                       command
                                                       WFS_CMD_IDC_RETAIN_CARD
             );
```

4.4 IXFSIFDMagneticTracks

This interface is provided to manage magnetic tracks capabilities, as defined in § 2.2.2.

GUID: {2F92D5F8-69D3-11D3-930B-0090272C227C}

4.4.1 Properties

NAME	TYPE	ACCESS	DESCRIPTION
ReadTracks	Enum	RO	Specifies the tracks that can be read by the card unit. Values: see WFSIDCCAPS.fwReadTracks
WriteTracks	Enum	RO	Specifies the tracks that can be written by the card unit. Values: see WFSIDCCAPS.fwWriteTracks
WriteMode	Enum	RO	Specifies the write capabilities (loco / hico magnetic stripes). Values: see WFSIDCCAPS.fwWriteMode

4.4.2 Methods

```
HRESULT      ReadTrack (
       IN         Enum            wTrack,              Track to retrieve.
                                                       Values: see property
                                                       ReadTracks.

       OUT        Enum *          pwStatus,            Status of operation.
                                                       Values: see
                                                       WFSIDCCARDDATA.wStatus

       OUT        ULONG *         pulDataLength,       Length of returned data.

       OUT        LPBYTE *        ppbData              Data read on track.
             );
```

```
HRESULT       WriteTrack (
       IN          Enum            wTrack,             Track to write.
                                                       Values: see property
                                                       WriteTracks.

       IN          Enum            wWriteMethod,       Write method (loco / hico).
                                                       Values: see
                                                       WFSIDCCARDDATA.fwWriteMethod

       IN          ULONG           ulDataLength,       Length of data to be written.

       IN          LPBYTE          pbData,             Data to be written on track.

       OUT         Enum *          pwStatus            Status of operation.
                                                       Values: see
                                                       WFSIDCCARDDATA.wStatus
             );
```

4.5 IXFSIFDSecurityModule

This interface is provided to manage associated security module, as defined in § 2.2.3.

GUID: {2F92D5F9-69D3-11D3-930B-0090272C227C}

4.5.1 Properties

NAME	TYPE	ACCESS	DESCRIPTION
Type	Enum	RO	Specifies the type of security module used. Values: see WFSIDCCAPS.fwSecType
Status	Enum	RO	Specifies the state of the security module. Values: see WFSIDCSTATUS.fwSecurity
Enabled	Boolean	RW	Specifies if the security check will be performed when accessing tracks.

4.5.2 Methods

```
HRESULT       SetKey (
       IN          WORD            wKeyLength,         Length of key value

       IN          LPBYTE          pbKeyValue          Key value

             );
```