Interoperability Specification for ICCs and Personal Computer Systems

Part 3. Requirements for PC-Connected Interface Devices

Apple Computer, Inc.

Axalto

Gemplus SA

Infineon Technologies AG

Ingenico SA

Microsoft Corporation

Omnikey

Philips Semiconductors

Toshiba Corporation

Revision 2.01.03

April 2005

Revision History

Revision	Issue Date	Comments
2.00.10	May 10, 2004	Spec 2.0 Final Draft
2.00.20	Feb 28, 2005	Added identification of contactless cards
2.01.00	March 8, 2005	Spec 2.01 Reviewed Draft
2.01.01	March 23, 2005	Spec 2.01 Reviewed Draft - minor edits
2.01.02	April 4, 2005	Spec 2.01 Incorporated minor edits
2.01.03	April 19, 2005	Spec 2.02 Incorporated minor edits

Contents

1 SCOPE 1

2 IFD SUBSYSTEM COMPONENTS 3

2.1 Interface Device 3

2.2 I/O Channel 3

2.3 IFD Handler 4

3 FUNCTIONALITY REQUIREMENTS 5

3.1 Mandatory Functionality 5
3.1.1 Common Functionality 5
3.1.1.1 General Operational characteristics 5
3.1.1.2 Enumeration of Device Capabilities 5
3.1.1.3 ICC Events 10
3.1.1.3.1 ICC Events 10
3.1.1.3.2 Behavior under System Power Management events 10
3.1.1.3.2.1 System Power down events 10
3.1.1.3.2.2 System Power up events 11
3.1.1.4 ICC Interface Management 11
3.1.1.5 Protocol Support 13
3.1.1.6 Channel mechanism support 13
3.1.2 Contact Environment Specifics 13
3.1.2.1 ISO/IEC 7816 Protocol Support 13
3.1.2.1.1 Protocol Negotiation 14
3.1.2.1.2 T=0 Protocol Support 15
3.1.2.1.3 T=1 Protocol Support 16
3.1.2.2 ICC Power Management 17
3.1.3 Contactless Environment Specifics 17
3.1.3.1 General Operational Characteristics 17
3.1.3.2 Contactless Protocol Support 18
3.1.3.2.1 Protocol Negotiation 18
3.1.3.2.2 Error Recovery 20
3.1.3.2.3 ATR 21
3.1.3.2.3.1 Contactless Smart Cards 21
3.1.3.2.3.2 Contactless Storage Cards 21
3.1.3.3 ICC Events 22

3.2 Optional Functionality 23
3.2.1 Common Functionality 23
3.2.1.1 Vendor-Specific Features 23
3.2.2 Contact/Contactless Environment Specifics 24
3.2.2.1 Storage Card Functionality Support 24
3.2.2.1.1 General Aspects 24

3.2.2.1.2 Common Error Code Definition 25
3.2.2.1.3 Get UID Command 26
3.2.2.1.4 Load Keys Command 27
3.2.2.1.5 Authenticate Command 28
3.2.2.1.6 Verify Command 29
3.2.2.1.7 Read Binary Command 30
3.2.2.1.8 Update Binary Command 31
3.2.2.2 Mechanical Characteristics 31
3.2.2.3 Security Assurance Devices 31

4 LOGICAL DEVICE INTERFACE REFERENCE 32

4.1 Syntax 32

4.2 Common Methods 32

4.3 Slot Logical Device Methods 33
4.3.1 Administrative Services 34
4.3.1.1 Retrieving an IFD Capability 34
4.3.1.2 Setting an IFD Capability 34
4.3.1.3 Protocol Information and Negotiation 35
4.3.1.4 ICC Power Management 36
4.3.1.5 Mechanical Characteristics 38
4.3.2 Communication Services 39
4.3.2.1 Data Exchange with the ICC 39
4.3.2.2 ICC Insertion and Removal 42
4.3.3 IFDSP Related Methods 42

1 Scope

This document discusses requirements for Interface Devices compliant with the *Interoperability Specification for ICCs and Personal Computer Systems*. For the purposes of this discussion, Interface Devices are PC-connected peripherals designed to interface to contact cards and/or to contactless cards. Such devices are also commonly called ICC smart card "Readers" or "Terminals". Cards encompassed by this specification must comply with the ISO/IEC 7816 for contact cards and ISO/IEC 14443 / 15693 for contactless cards. Other specifications may describe IFD or ICC Terminal functionality that differs in scope from the Interface Devices requirements herein.

In this document, the term "IFD Subsystem" shall be used to denote a PC peripheral subsystem consisting of:

- An Interface Device (IFD), which provides principally the interface for ICCs.
- An I/O channel managed by an I/O device driver on the PC side.
- PC-hosted IFD handler software, which interfaces with the upper layer of the overall *Interoperability Specification for ICCs and Personal Computer Systems* architecture and the low-level I/O device drivers, as depicted in Figure 1-1.

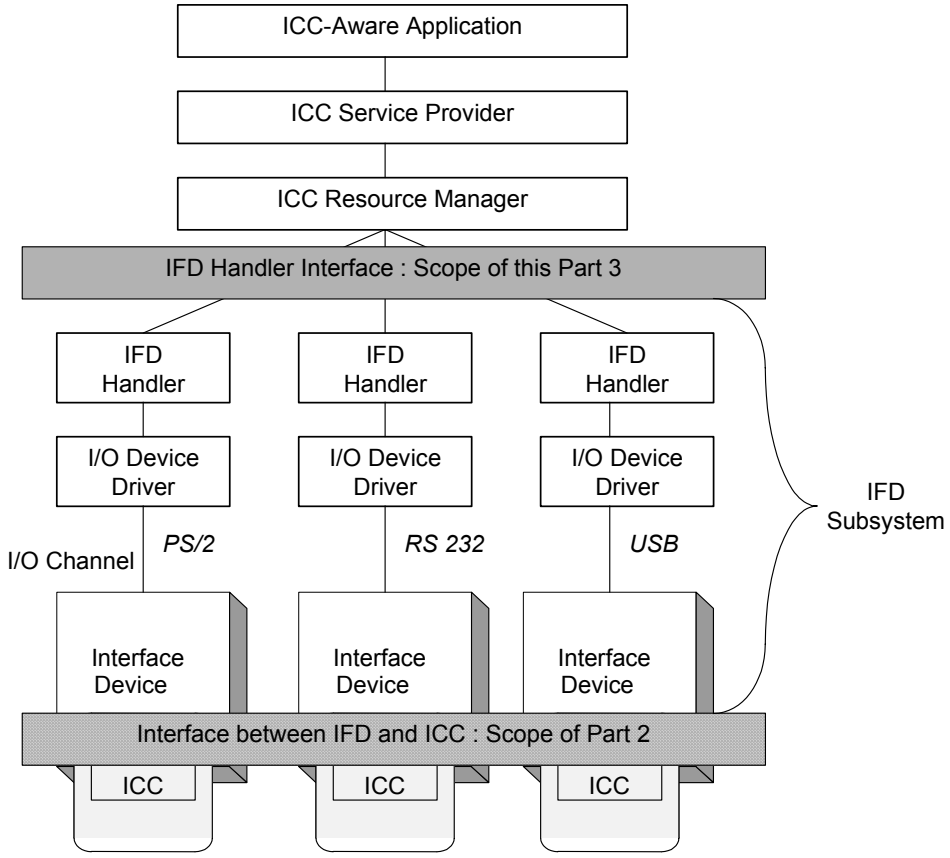


Figure 1.1. IFD Subsytem components and interfaces with other components

A vendor must fully implement the IFD Subsystem. In doing so, they may create the entire subsystem or take advantage of standard device drivers, I/O channels, and/or default IFD Handlers, which may be provided by other vendors. In order to claim compliance however, they must insure that the entire subsystem functions in accordance with this specification. In particular, this means they must support:

- An ICC interface as described in Part 2, “Interface Requirements for Compatible IC Cards and Interface Devices.”
- A PC software interface and associated functionality, as described in this document.

This document identifies specific requirements for compliant Interface Devices. Reference designs for specific types of Interface Devices can be found in Part 4 of these specifications “Interface Device Design Considerations and reference design information”.

2 IFD Subsystem Components

The IFD Subsystem is a PC peripheral subsystem consisting of the components identified in Section 1. It is the intent of this specification to define requirements that are supportable across a wide variety of Interface Device designs and that encourage continued innovation by Interface Device vendors. In order to achieve this, the IFD Subsystem employs a layered architecture, which allows for design variations and supports a variety of data communications interfaces between the Interface Device and PC. Requirements on compliant devices are specified at the interface to the ICC and at the programming interface (IFD Handler Interface) exposed on the PC.

2.1 Interface Device

The Interface Device is a PC peripheral that:

- Interfaces to ICCs as described in Part 2, “Interface Requirements for Compatible IC Cards and Interface Devices.”
- Supports bidirectional data communications between ICCs and PC.
- Incorporates functionality required to support the interface exposed by the IFD Handler.
- Interfaces some additional enhanced capabilities, if any, such as access to a display and a keyboard.

There are limited physical design requirements imposed on compliant Interface Devices, primarily those dictated by compatibility with the ICC physical interface.

In the interest of reliability, it is strongly recommended that these devices for contact ICCs employ:

- A landing card or landing contact design.
- A microswitch for card detection.

For contactless Interface Devices, Part 2 describes design considerations to provide the required card detection functionality.

The Interface Device should employ a tamper-evident design if it incorporates any of the optional security assurance features discussed in section 3.2.2.3. These features should provide reasonable assurance to the end user that attempts to access or modify the Interface Device are detectable. See ISO 13491:1995 for information regarding tamper-evident devices.

2.2 I/O Channel

This specification is compatible with IFD Subsystems using any available PC I/O channel to communicate with the Interface Device. It is the responsibility of the vendor to create an IFD Handler and device drivers, as necessary to insure that the ICC Service Providers can communicate with the ICC through the Interface Device. It is also the responsibility of the Interface Device vendor to provide appropriate management services and error handling related to the PC I/O channel to insure reliable, error free communication.

The USB interface is likely to be the best solution for desktop PCs, due to the industry momentum to make it the preferred I/O channel for PC peripherals.

Alternate designs are not precluded, and may include:

- *PS/2 Keyboard Integrated Interface Device*
- *Serial Port (RS-232C) interface*
- *Interface Devices employing custom interface boards*
- *Parallel Port interfaces*
- *PC Card-based Interface Devices*
- *SCSI interfaces*

2.3 IFD Handler

The IFD Handler is software running on the PC that implements a standard, hardware-independent and I/O channel–independent interface into the IFD subsystem for PC-based software. It is the responsibility of this IFD Handler to interface to the I/O channel used by the Interface Device, either via standard device drivers or by implementing appropriate functionality. It is also required to map Interface Device functionality onto these interfaces.

The IFD handler can expose two types of logical devices: the slot logical device and the functional logical device.

The slot logical device provides a communication channel with an ICC. For contact card readers, there are as many slot logical devices as there are ICC slots in the IFD (SAMs acceptance devices may also be modeled as slots). For contactless readers, the slot logical channel corresponds to a data link layer channel.

A functional logical device, as described more precisely in Part 9, interfaces the extended capabilities of an IFD, such as a display or a keypad.

Depending on the vendor-implemented Interface Device functionality, the complexity of the IFD Handler can vary dramatically. For example, given an ISO/IEC 7816-compatible ICC Interface Device with limited intelligence and support for only the ISO/IEC 7816 physical data link layer, the IFD Handler will need to implement the T=0 and T=1 data link layers, provide associated error handling, and so on. In contrast, a vendor could implement a sophisticated Interface Device that internally supports the T=0 and T=1 data link layers. In this case, the IFD Handler will be significantly simpler in design.

These specifications do not mandate a specific implementation for IFD Handlers. Compatible IFD Handlers may be implemented as statically linked libraries, dynamically linked libraries (sometimes called shared libraries), or device drivers. Recommended implementations for a specific environment should be specified by the Operating System vendor. This allows each OS vendor to optimize the implementation in consideration of the OS services, device driver model, and supported hardware.

3 Functionality Requirements

This Section presents information on functional requirements for the IFD Subsystem visible at the PC-software interface implemented by the IFD Handler. It discusses both mandatory and optional functionality.

3.1 Mandatory Functionality

The following subsections identify mandatory functionality for compliant IFD subsystems. It is presented first as functionality common to both contact and contactless environments. Then specific requirements for contact environments only and contactless environments only are discussed in sections 3.1.2 and 3.1.3, respectively.

3.1.1 Common Functionality

3.1.1.1 General Operational characteristics

The logical devices exposed by IFD handlers shall present a uniform set of services to the Resource Manager. An IFD Handler shall only support a single, active, logical connection between a PC-based application and a logical device at any given time. That is, the IFD Handler is not required to support multiple active connections to PC software programs nor is it required to provide any services related to the management of multiple logical connections and associated command streams.

For the slot logical device, this does not preclude use of the “session” management features defined in ISO/IEC 7816 such as the “channel” mechanisms defined in 7816-4.. However, implementation of these features is the responsibility of the ICC and associated ICC Service Provider(s).

3.1.1.2 Enumeration of Device Capabilities

The slot logical device shall provide an interface that supports enumeration of the functionality, both mandatory and optional, that is implemented by the IFD subsystem. An ICC Service Provider shall be able to query this interface at any time. Information will be returned using a TLV (tag-length-value) structure. At a minimum, the following information shall be retrievable.

Table 3-1. Codes for Enumerating Interface Device Capabilities

Information Class	Data Element	TAG	Max. Length	Data Encoding
Vendor				
	• Vendor Name	0x0100	32 bytes	ASCII string
	• Vendor-specified IFD Type	0x0101	32 bytes	ASCII string
	• Vendor-specified IFD Version number	0x0102	4 bytes	DWORD encoded as 0X*MMmmbbbb* where: • *MM* = major version • *mm* = minor version • *bbbb* = build number
	• IFD Serial Number	0x0103	32 bytes	ASCII string
Communications				
	• Channel ID	0x0110	4 bytes	DWORD encoded as 0x*DDDDCCCC* where: • *DDDD* = data channel type • *CCCC* = channel number The following encodings are defined for *DDDD*: • 0x01 serial I/O; *CCCC* is port number. • 0x02 parallel I/O; *CCCC* is port number. • 0x04 PS/2 keyboard port; *CCCC* is zero. • 0x08 SCSI; *CCCC* is SCSI ID number. • 0x10 IDE; *CCCC* is device number. • 0x20 USB; *CCCC* is device number. • 0xF*y* vendor-defined interface with *y* in the range 0 through 15; *CCCC* is vendor defined.

Information Class	Data Element	TAG	Max. Length	Data Encoding
Protocol (see Part 2 of this specification)				
	• ISO 7816 (asynchronous) / 14443 / 15693 cards.	0x0120	4 bytes	DWORD encoded as 0x*00RRPPPP* where: • *RR* is RFU and should be 0x00. • *PPPP* encodes the supported protocol types. A '1' in a given bit position indicates support for the associated ISO protocol. Example: 0x00000003 indicates support for T=0 and T=1. This is the only *compliant* value that may be returned by contact devices at this time. Contactless devices may return 0x00010000. All other values (T=14, T=15, and so on, are outside of this specification, and must be handled by vendor-supplied drivers).
	• Default CLK	0x0121	4 bytes	Default ICC CLK frequency in KHz encoded as little endian integer value. Example: 3.58 MHz is encoded as the integer value 3580. 13560 is be the value for Contactless.
	• Max CLK	0x0122	4 bytes	Maximum supported ICC CLK frequency in KHz encoded as little endian integer value. 13560 is the value for Contactless.
	• Default Data Rate	0x0123	4 bytes	Default ICC I/O data rate in bps encoded as little endian integer.
	• Max Data Rate	0x0124	4 bytes	Maximum supported ICC I/O data rate in bps.
	• Max IFSD	0x0125	4 bytes	DWORD indicating maximum IFSD supported by IFD. At least 32,254 is recommended.

Information Class	Data Element	TAG	Max. Length	Data Encoding
	• Synchronous Protocol Types supported (for ISO/IEC 7816-compatible ICCs)	0x0126	4 bytes	DWORD encoded as 0x*40RRPPPP* where: • *RR* is RFU and should be 0x00. • *PPPP* encodes the supported protocol types. A '1' in a given bit position indicates support for the associated protocol. 0x0001 indicates support for 2-wire protocol. 0x0002 indicates support for 3-wire protocol. 0x0004 indicates support for I²C-Bus protocol. All other values are outside of this specification, and must be handled by vendor-supplied drivers.
Power Management				
	• Power Mgt Supported	0x0131	4 bytes	If 0, device does not support power down while ICC inserted. If non-zero, device does support it.
Security Assurance features				
	• User to Card Authentication Devices	0x0140	4 bytes	DWORD that is the result of a bitwise OR operation performed on the following values: • 0x00000000 No devices • 0x00000001 RFU • 0x00000002 Numeric (that is, PIN) pad • 0x00000004 Keyboard • 0x00000008 Fingerprint scanner • 0x00000010 Retinal scanner • 0x00000020 Image

Information Class	Data Element	TAG	Max. Length	Data Encoding
				scanner • 0x00000040 Voice print scanner • 0x00000080 Display device • 0x0000*dd*00 *dd* is vendor selected for vendor-defined device
	• User Authentication Input Device	0x0142	4 bytes	DWORD that is the result of a bitwise OR operation performed on the following values: • 0x00000000 No devices • 0x00000001 RFU • 0x00000002 Numeric (that is, PIN) pad • 0x00000004 Keyboard • 0x00000008 Fingerprint scanner • 0x00000010 Retinal scanner • 0x00000020 Image scanner • 0x00000040 Voice print scanner • 0x00000080 Display device • 0x0000*dd*00 *dd* in the range 0x01 to 0x40 is vendor selected for vendor-defined device • 0x00008000 Used to indicate that encrypted input is supported
Mechanical Characteristics				
	• Mechanical characteristics supported	0x0150	4 bytes	DWORD that is the result of a bitwise OR operation performed on the following values: • 0x00000000 No special characteristics • 0x00000001 Card swallowing mechanism • 0x00000002 Card ejection mechanism

Information Class	Data Element	TAG	Max. Length	Data Encoding
				• 0x00000004 Card capture mechanism • 0x00000008 Contactless. • All other values are RFU
Vendor Defined Features		May use values in range 0x0180–0x01F0	--	

Note that the Protocol information identified above should indicate the potential range of ICC protocols and parameters supported. The actual protocols and parameters that may be used at any given time are dictated by the intersection of this set and the ICC supported set. This is discussed more thoroughly in Sections 3.1.2.1.1 for contact cards and 3.1.3.2.1 for contactless cards.

3.1.1.3 ICC Events

3.1.1.3.1 ICC Events

The slot logical device shall support the following ICC-related events:

- Card insertion notification
- Card removal notification

When one of these events occurs, it is the responsibility of the IFD Handler to notify the ICC Resource Manager layer.

3.1.1.3.2 Behavior under System Power Management events

PC operating systems may implement power management functionality that reduces overall system power consumption. This feature, if present, requires the IFD subsystem to power off the IFD and any inserted (or energized, in a contactless environment) smart card. Since cards and readers are currently unable to operate at a low power state, this specification only distinguishes system states of normal operation and stand-by mode.

Normal mode implies that the system is operable and can be used. Stand-by mode implies that the system is inoperable and can be wakened-up by user request.

3.1.1.3.2.1 System Power down events

A power down event is defined as the transition from system normal operation to system stand-by mode.

If such an event request occurs during I/O command processing (pending card insertion/removal events are not considered I/O operations in this case), the IFD subsystem must refuse such a request.

On power down events, the IFD handler must record the card state (card present/absent) and power off any inserted smart card.

3.1.1.3.2.2 System Power up events

A power up event is defined as the transition from system stand-by mode to system normal operation.

The following IFD handler behavior is required on power up, for each slot logical device:

Case 1: No card in logical slot before power-down and no card in logical slot at power-up. The IFD handler must NOT complete any card insertion notification request

Case 2: No card in logical slot before power-down and a card in logical slot at power-up.
The IFD handler must complete any insertion notification request.

Case 3: card in logical slot before power-down and no card in logical slot at power-up.
The IFD handler must complete any card removal notification request

Case 4: card in logical slot before power-down and card in logical slot at power-up.
The IFD handler must complete any card removal notification request. This is necessary due to the fact that the card could have been removed and inserted during stand-by mode.

3.1.1.4 ICC Interface Management

The IFD Subsystem shall be able to manage the electrical interface to ICCs in accordance with Part 2 of this specification. This includes provision of all required signals, within stated limits, at the ICC contacts or in the coupling area (energizing field), and compliance with required activation and deactivation sequences. The activation sequence includes several procedures according to each standard respectively, as follows;

1) in case of ISO/IEC 7816-compatible ICCs : proper activation of the ICC contacts, performing a cold reset of the ICC, and retrieving and parsing the ATR sequence;

2) in case of contactless ICCs:
   - proper initialization and anticollision
   - with ISO 14443-4 Type A PICCs, retrieval of the ATS. With Type B PICCs, retrieval of the ATQB. ISO 14443-3/ISO 15693 VICCs: Recording of UID and card properties.
   - generating an ATR sequence.

As stated in Part 2, the IFD Subsystem should notify to the ICC Resource Manager in the event that a valid ATR is not transmitted by the ICC. It is up to the ICC Resource Manager either to deactivate the card or to proceed with the default parameters. If a

valid ATR is transmitted, and the Interface Device supports programmable protocol parameters (that is, selection of frequency, IFS, and so on) the IFD subsystem shall parse the ATR and determine all supported protocol and associated parameter options. The supported options for a given ICC are determined by the intersection of the options supported by the Interface Device and the ICC (as specified in the ATR). Note that an Interface Device for ISO/IEC 7816-compatible ICCs is required to support T=0 and T=1 only at a default clock frequency.

To ensure that the PC software can determine the state of the ICC at any time, the slot logical device shall expose an interface, which allows the ICC state to be queried. This will include the ability to determine the following information, through the following TLV structure.

Table 3-2. Codes for Enumerating ICC State

Information	TAG	Maximum Length	Responses (return as integer)
ICC presence	0x0300	1 byte	• 0 = not present • 1 = card present but not swallowed (applies only if the IFD supports ICC swallowing) • 2 = card present (and swallowed if the IFD supports ICC swallowing) • 4 = card confiscated
ICC interface status[1]	0x0301	1 byte	BOOLEAN, 0 = contact inactive 1 = contact active
ATR string	0x0303	32 bytes	Contains the ATR string as returned by the IFD subsystem
ICC type	0x0304	1 byte	ISO/IEC card or unknown • 0 = unknown ICC type, • 1 = 7816 Asynchronous • 2 = 7816 Synchronous (unspecified) • 3 = 7816-10 Synchronous (Type 1) • 4 = 7816-10 Synchronous (Type 2) • 5 = 14443 (Type A) • 6 =14443 (Type B) • 7 = ISO 15693 • Other values RFU

In addition, in case of the IFD compliant with ISO/IEC 7816, to enable a PC application to ensure that an ICC is in a known state, the IFD Subsystem will provide a mean to reinitialize an ICC using a warm reset on demand. For ISO/IEC 14443-compatible ICCs,

[1] This is for ISO/IEC 7816-compatible card only.

equivalent functionality can be achieved by placing the ICC in the Halt State and then waking it up.

The IFD Subsystem is responsible for determining when unrecoverable data communications errors have occurred and informing the logically connected ICC Service Provider, if any. At a minimum, the IFD Subsystem shall be able to distinguish between an unresponsive ICC and unrecoverable communications errors, indicated by repeated parity and/or checksum errors, and make this information available to a connected ICC Service Provider. Those error codes can be defined by the operating system vendor either as new error codes or mapped onto standard operating system error codes, provided the previous stated requirements are met.

3.1.1.5 Protocol Support

To free the application from managing all protocol-related functionality with each standard (ISO/IEC 7816/14443/15693 and similar), each IFD subsystem shall implement the physical layers and the proper link layers in accordance with each standard, as discussed in the subsequent sections.

3.1.1.6 Channel mechanism support

Upon support of ISO/IEC 7816, slot logical devices provide implicit support for the logical sessions using the ‘channel’ mechanisms of ISO/IEC 7816-4. It is the responsibility of the ICC Service Provider, and associated ICC, to implement and manage such channels.

3.1.2 Contact Environment Specifics

This section presents mandatory requirements that pertain to contact environments only.

3.1.2.1 ISO/IEC 7816 Protocol Support

In terms of the ISO 7816 layer definition, the IFD Subsystem, as a whole, is responsible for the implementation of the physical and data link layers, as defined in 7816-3 and 7816-10. The flow of information between the ICC Service Provider and the IFD Subsystem is depicted in Figure 3-1 and further explained in the following chapters. In a nutshell, the IFD subsystem hides from the application level all protocol-related details and presents a standard interface based on ISO 7816-4 commands/responses structure

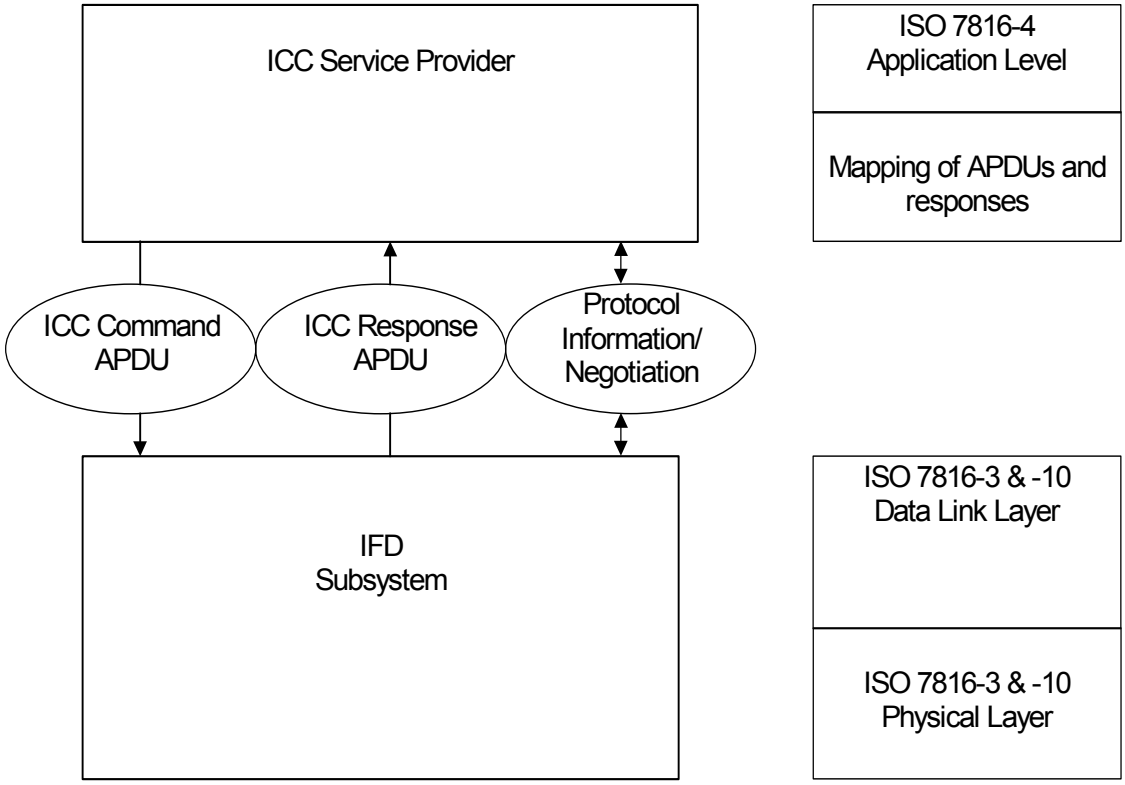


Figure 3-1. Role of the IFD Subsystem in terms of the ISO 7816 layers

3.1.2.1.1 Protocol Negotiation

A compliant Interface Device is required to support both the T=0 and T=1 protocols. It may support optional synchronous protocols or protocols for contact storage cards. It must support the default global parameters and a default CLK frequency in the range of 1 to 5 MHz. It is not, however, required to support flexible protocol and parameter selection. That is, an IFD Subsystem need only parse an ATR to determine the default protocol and parameters.

It is recommended that Interface Devices provide support for protocol selection and setting of associated parameters in order to maximize performance. Interface Devices providing this ability must parse the ATR and determine the intersection of the ICC-supported options with those supported by the Interface Device. In general, an IFD should wait until the first application connects to the device prior to negotiating the protocol settings (using PPS). Connection requests contain hints indicating the protocol desired and an indication of whether timing parameters should be optimized or left at the default value.

The slot logical device will expose an interface that allows a Service Provider to enumerate current protocol setting and parameter options. The following table indicates the available information and which of these may be set explicitly. Explicit setting of the protocol basic timing parameters is not allowed because the IFD will automatically negotiate these settings based on its capabilities and the ICC capabilities. Other parameters may not be settable based on T=0 and T=1 protocol definitions.

Table 3-3. Codes for Enumerating Interface Device Protocol Options

Data Element	TAG	Max. Length	Read Only	Comments
Current Protocol Type	0x0201	4 bytes	X	DWORD encoded in same manner as Available Protocol Types. It is illegal to specify more than a single protocol in this value.
Current CLK	0x0202	4 bytes	X	Current ICC CLK frequency in KHz encoded as little endian integer value. Example: 3.58 MHz is encoded as the integer value 3580.
Current F (clock conversion factor)	0x0203	4 bytes	X	F encoded as a little endian integer. (May be modified through PPS)
Current D (bit rate conversion factor)	0x0204	4 bytes	X	D encoded as a little endian integer. (May be modified through PPS)
Current N (guard time factor)	0x0205	4 bytes	X	N encoded as a little endian integer. (May be modified through PPS)
Current W (work waiting time)	0x0206	4 bytes	X	W encoded as a little endian integer. Only valid if current protocol is to T=0.
Current IFSC (information field size card)	0x0207	4 bytes	X	IFSC encoded as a little endian integer. Only valid if current protocol is T=1.
Current IFSD (information field size Reader)	0x0208	4 bytes		IFSD encoded as a little endian integer. Only valid if current protocol is T=1.
Current BWT (block waiting time)	0x0209	4 bytes	X	BWT encoded as a little endian integer. Only valid if current protocol is T=1.
Current CWT (character waiting time)	0x020A	4 bytes	X	CWT encoded as a little endian integer. Only valid if current protocol is T=1.
Current EBC encoding	0x020B	4 bytes	X	EBC encoded as: • 0 = LRC • 1 = CRC Only valid if current protocol is T=1.

3.1.2.1.2 T=0 Protocol Support

From the ICC Service Provider perspective, an IFD Subsystem shall accept Case 1, 2 or 3 short APDUs, as defined in ISO 7816-4 and asynchronously return the response from the ICC. Direct Case 4 APDU support is not required. The IFD Subsystem shall change this APDU to TPDU for T=0 command. Especially it is mandatory to change APDU case1 format *(CLA INS P1 P2*) to TPDU with *P3*='00' (*CLA INS P1 P2 '00'*).

The IFD Subsystem is responsible for detecting and responding to physical link errors (bit parity errors) as required by ISO/IEC 7816. In terms of processing ICC response bytes, the IFD Subsystem is only required to correctly process an ICC ACK response. It will then either send remaining data associated with the last command header or retrieve

remaining data from the ICC. In this case, the IFD Subsystem does not return the ACK byte to the ICC Service Provider.

It is not required to interpret or respond to other ICC response bytes, as these are generally application specific, and shall return them to the ICC Service Provider.

3.1.2.1.3 T=1 Protocol Support

From the ICC Service Provider perspective, an IFD Subsystem shall accept 7816-4 APDUs, construct the necessary T=1 blocks to convey those APDUs and asynchronously return the response from the ICC. It is implicit that the ICC Service Provider has the right to transmit the first such block. The IFD Subsystem must at all times track whether it, or the ICC, has the right to transmit. If the ICC has the right to transmit, an IFD Subsystem may return an error if an ICC Service Provider attempts to initiate a block transmission request.

The IFD Subsystem is responsible for detecting the following errors:

- Transmission error (incorrect parity or an EDC error) denoted by the least significant nibble of the PCB of the R-Block sent by the ICC
- BWT time-out
- Loss of synchronization (wrong number of characters received)

In the event such an error is detected, the IFD Subsystem shall interpret it to mean that the associated block is invalid.

The IFD Subsystem is generally responsible for automatic retransmission of the last block sent, in the event of a protocol error. A maximum of three such attempts will be made, after which the ICC shall be deactivated and the ICC Service Provider informed that an unrecoverable error has occurred. All other errors are the responsibility of the ICC Service Provider and/or ICC. The following lists the specific procedures to be implemented by the IFD Subsystem:

1. If there is no response from the ICC to a block sent by the IFD within BWT, the IFD Subsystem shall follow the scenario 33 or 35 defined in the ISO 7816-3 Annex A document, in order to try to recover the communication. If it fails, the IFD Subsystem shall deactivate the ICC and inform the ICC Service Provider that an unrecoverable error has occurred.

2. If an invalid block is received in response to an R-block, the sender shall retransmit the R-block.

3. If an S(response) is not received in response to an S(request), the sender shall retransmit the S(request).

4. If an invalid block is received in response to an S(response) block, the IFD Subsystem shall transmit an R-block with bit 5 = sequence number of the next expected I-block.

5. If the IFD Subsystem detects an underrun or overrun condition, it will wait the greater of CWT or BWT before transmitting the last block.

6. When the ICC sends an S(IFS request) and receives an invalid response, it will retransmit the block only one time to elicit an S(IFS response) and then remain in receive mode.

The only time an IFD Subsystem is expected to initiate a block is for the purpose of establishing an IFSD larger than the default size of 32 bytes. The IFD Subsystem should do this only as the first block following negotiation of the T=1 protocol. In other words, the IFD Subsystem may send an S(IFS request) and then process an S(IFS response) prior to transmitting the first ICC Service Provider block.

The ICC Service Provider should assume this will be done if the Interface Device capabilities indicate it supports a maximum IFSD greater than 32 bytes. As such, it should wait until after the first block has been transmitted to determine the effective IFSD setting.

Compliant IFD Subsystems provide implicit support for the chaining function, as described in ISO 7816-3, so as to enable the IFD (the ICC) to transmit information longer than IFSC (IFSD).

Compliant IFD Subsystems may provide support for the logical sessions using the Node Addressing mechanisms associated with T=1. If logical sessions are not supported, DAD and SAD should be set to 0 (zero).

3.1.2.2 ICC Power Management

An Interface Device must support the ability to activate and deactivate an inserted ICC under control of an ICC Service Provider. For ISO/IEC 7816-compatible IFDs, this may be desirable to minimize power consumption in certain environments when the ICC is expected to be inserted for long periods of time, but used infrequently. If implemented, a request to activate an ICC shall result in activation of the ICC contacts, generation of a cold reset, and processing of the ATR sequence as specified above.

3.1.3 Contactless Environment Specifics

This section presents mandatory requirements that pertain to contactless environments only.

3.1.3.1 General Operational Characteristics

The underlying requirement of the contactless support in the PC/SC architecture is to emulate the contact card environment to provide consistent system functionality. When the technology requires the use of specific mechanisms, such as anticollision management for contactless ICCs, the IFD subsystem is responsible for the management of these processes

A contactless environment allows the presence and the use of several ICCs at the same time. The IFD Handler can support multiple logical connection and present each of them as an ICC slot logical device to the Resource Manager and higher components. A logical connection is associated to a card when it enters the energizing field and is recognised. Thus, the calling software can communicate with several cards by connecting to the different slot logical device.

The IFD subsystem must establish and keep an association between an ICC and a slot logical device as long as the card remains logically inserted. The IFD Subsystem is responsible to address the proper ICC when a transmission is required to a slot logical device and to serialize all commands, at the data link layer, that pertain to a single IFD. The IFD subsystem must also ensure that communicating with an ICC does not affect the current state of other logically inserted ICCs present in the field.

3.1.3.2 Contactless Protocol Support

The IFD Subsystem in this specification is responsible for the implementation of the physical and data link layers, as defined in ISO/IEC 14443-2, -3, and –4 or ISO 15693. The flow of information between the ICC Service Provider and the IFD Subsystem is depicted in Figure 3-2 and further explained in the following sections. The IFD Subsystem must hide all protocol-related details from the application level and presents an interface based on ISO/IEC 7816 and/or other proprietary application command /response. An application will communicate with the card using the SCARD_PROTOCOL_T0, SCARD_PROTOCOL_T1, or SCARD_PROTOCOL_RAW protocol identifier, as defined in part 5.

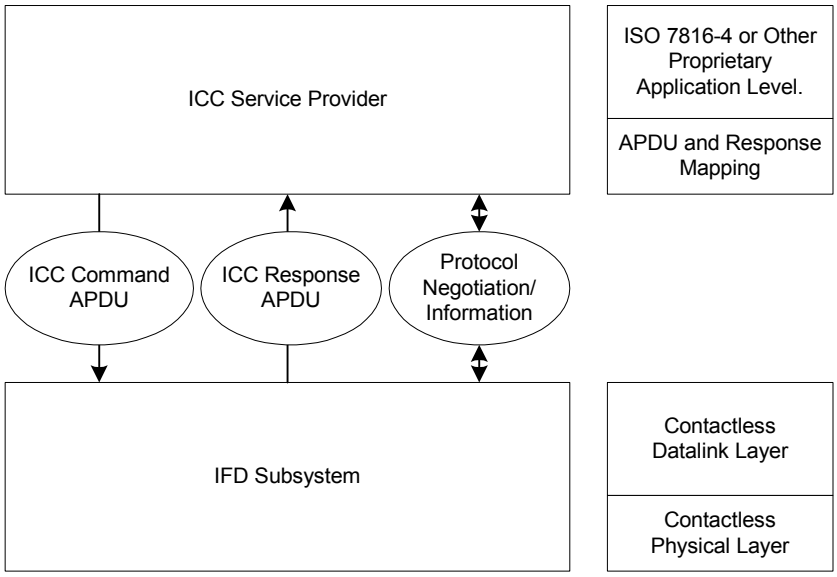


Figure 3-2. Role of the IFD Subsystem in Terms of the ISO 14443 layers

3.1.3.2.1 Protocol Negotiation

After detecting the presence of a card, the IFD Subsystem shall manage the communication with the card to bring it into the Active State and establish the half-duplex block protocol, as defined in ISO/IEC 14443-3, -4 / ISO 15693. The IFD Subsystem must perform the communication settings before the half-duplex transmission begins.

A card insertion notification is sent to the Resource Manager only after the PICC/VICC gets recognized. In that state, the PICC/VICC can be directly addressed and card tracking can be ensured:

- For ISO 14443-4 Type A PICC: the Active State is reached when the activation sequence is completed, i.e. after the ATS is received by the PCD and PPS has been performed.
- ISO 14443-3 or ISO 15693 cards: After recognizing a card (which does not support multi-activation) it is required to record its properties and to halt it immediately.
- For ISO 14443 Type B PICC: Active State is reached after the ATTRIB command is issued.

Then, the IFD handler associates the card with a slot logical device and fills the data structure related to the device capabilities. An ATR is constructed according to the rules specified in section 3.1.3.2.3. Finally, the IFD handler generates a card insertion notification related to the slot logical device assigned to the card.

The IFD Subsystem must manage the communication exchanges according to the half-duplex block protocol described in ISO/IEC 14443 / 15693 and transparently construct block, perform chaining and handle error communication.

When a card that is not compliant with this specification enters the energizing field, the IFD Subsystem must place the card into the HALT state and there must be no effect outside of the boundaries of the IFD subsystem.

The table 3-5 indicates the current protocol information exposed by the slot logical device to higher layers.

Table 3-4. Codes for Enumerating Interface Device Protocol Options

Data Element	TAG	Max. Length	Read Only	Comments
Current Protocol Type	0x0201	4 bytes	X	SCARD_PROTOCOL_RAW
Current RF frequency	0x0202	4 bytes	X	Current RF frequency in kHz encoded as little endian integer value. This is the fixed frequency at 13.56 MHz encoded as the integer value 13560.
Current D (bit rate conversion factor)	0x0204	4 bytes	X	D consists of D_S and D_R as follows. • D_S is encoded by the most significant 2 bytes as a little endian integer, which is bit rate factor for the direction PICC/VICC to PCD/VCD. • D_R is encoded by the least significant 2 bytes as a little endian intger, which is bit rate factor for the direction IFD to PICC (May be modified through PPS with Type A or ATTRIB command with Type B.)
Current FSC (maximum frame size which the PICC is able to receive)	0x0207	4 bytes	X	FSC encoded as a Little Endian integer, if it applies.
Current FSD (maximum frame size which the IFD is able to receive)	0x0208	4 bytes		FSD encoded as a Little Endian integer, if it applies.
Current FWT (frame waiting time)	0x0209	4 bytes	X	FWT encoded as a Little Endian integer, if it applies.

3.1.3.2.2 Error Recovery

From the ICC Service Provider perspective, an IFD Subsystem shall accept APDUs, construct the necessary ISO/IEC 14443/15693 compliant protocol blocks out of them and send them to the PICC/VICC as well as receive replied protocol blocks and put them together to response APDUs.

The IFD Subsystem is responsible for detecting protocol errors and for applying the recovery procedure specified in ISO/IEC 14443 or 15693.

When unrecoverable errors are encountered, ISO/IEC 14443-4 requires that the PICC has to be deactivated or ignored. In this case, the IFD subsystem must generate a card removal notification. The same would apply to ISO/IEC 15693 compliant VICCs.

3.1.3.2.3 ATR

For contactless ICCs, the IFD subsystem must construct an ATR from the fixed elements that identify the cards.

3.1.3.2.3.1 Contactless Smart Cards

Table 3-5 ATR Returned by IFD Handler for Contactless Smart Card ICCs

Byte Nr	Value	Designation	Description
0	3B	Initial Header	
1	8n	T0	Higher nibble 8 means no TA1, TB1, TC1 only TD1 is following. Lower nibble n is the number of historical bytes (HistByte 0 to HistByte n-1)
2	80	TD1	Higher nibble 8 means no TA2, TB2, TC2 only TD2 is following. Lower nibble 0 means T = 0
3	01	TD2	Higher nibble 0 means no TA3, TB3, TC3, TD3 following Lower nibble 1 means T = 1
4 to 3+n	XX XX XX	T1 … .. Tk	Historical bytes: ISO14443A: The historical bytes from ATS response. Refer to the ISO14443-4 specification. ISO14443B: The higher layer response from the ATTRIB response. Refer to the ISO14443-3 specification.
4+n	XX	TCK	Exclusive-OR of bytes T0 to Tk

The Contactless Smart Card exposes its ATS or information bytes not directly, but via a specific ATR mapping. For those cards that provide such information, optionally with Historical Bytes (or Application Information respectively), the mapping in the table above applies.

3.1.3.2.3.2 Contactless Storage Cards

Table 3-6 ATR Returned by IFD Handler for Contactless Smart Card ICCs

Byte Nr	Value	Designation	Description
0	3B	Initial Header	
1	8n	T0	Higher nibble 8 means: no TA1, TB1, TC1 only TD1 is following. Lower nibble n is the number of historical bytes (HistByte 0 to HistByte n-1)
2	80	TD1	Higher nibble 8 means: no TA2, TB2, TC2 only TD2 is following.

					Lower nibble 0 means T = 0		
3	01		TD2		Higher nibble 0 means no TA3, TB3, TC3, TD3 following. Lower nibble 1 means T = 1		
4 to 2+n	XX XX XX	Hist bytes	T1 .. .. .. Tk	80	Category indicator byte, 80 means A status indicator may be present in an optional COMPACT-TLV data object.		
				4F	Application identifier Presence indicator		
				YY	length		
				YY YY .. .. YY	5 bytes	Registered application provider identifier (RID)	
					PIX maximum 7 bytes PP PP .. .. PP	SS	Byte for Standard
						NN NN	Bytes for Card Name
						RR RR RR RR	RFU: Shall be set to zero. Assigned by PC/SC for future extensions.
3+n	UU		TCK		Exclusive-oring of all the bytes T0 to Tk		

The ATR of a Contactless Storage Card is composed as described in the table above. In order to allow the application to identify a storage card type properly, its Standard and Card Name describing bytes must be interpreted according to the Part 3 Supplemental Document, maintained by PC/SC.

3.1.3.3 ICC Events

In a ISO/IEC 14443 contactless environment, support for card insertion detection is already provided by the standards. Card removal, however, is only detected when a card no longer answers a request. A polling mechanism must be used by the IFD subsystem to detect ICC removal. The IFD subsystem must ensure that states of the ICCs present in the field are not affected by this mechanism. Section 4.2.1 of Part 2 proposes an implementation of this polling process.

The specific circumstances, which imply a card removal event notification are:

- When an IFD subsystem detects, through a polling timeout, that a PICC/VICC has left the energizing field, it must generate a card removal notification on the appropriate slot logical device.

- The Resource Manager may instruct the IFD subsystem to power off a card.

  In a contactless ISO/IEC 14443-4 environment, this command maps to the PCD sending a DESELECT command to the PICC, in order to halt it. The CID assigned to the PICC must then be freed (and can subsequently be assigned to other PICCs).

The PCD subsystem must not generate a card removal notification since the card is still logically inserted and can be polled at ISO 14443-3 level by periodically attempting to reactivate it.

ISO/IEC 14443-3 and 15693 cards are in general in the HALT or QUIET state respectively. This state allows for card polling. When a Slot Logical Device where such a card is “inserted” receives a command to power down the ICC it needn’t perform any action. Polling can continue. The slot is still sensitive to card events. For communication, one halted/quiet ISO/IEC 14443.3 / 15693 card is exclusively reactivated.

- The Resource Manager may instruct the IFD subsystem to perform a warm reset on card. In a contactless environment, this command maps to the following actions:

  1. The reader sends the corresponding command to the card, in order to halt it.

  2. The IFD sends a WAKE-UP command to allow for this card to enter the ready state and be reactivated.

  3. The IFD explicitly selects and activates the card, and associates it with the same slot logical device that it previously occupied.

  If the PICC/VICC was physically removed from the energizing field anywhere in steps 1 to 3, it will not respond to the activation request in step 3. The IFD subsystem must then generate a card removal notification on the slot logical device occupied by the card and free that slot for subsequent use.

3.2 Optional Functionality

IFD subsystems may implement optional functionality, as described in the following sections. Again provisions that apply to both contact and contactless environments are presented first. Specific provisions follow.

3.2.1 Common Functionality

3.2.1.1 Vendor-Specific Features

Compliant Interface Devices may implement vendor-specific features and functionality not identified in this specification. Such features must be isolated so that they do not impact required functionality identified herein nor allow such functionality to be circumvented.

If such features are implemented, they shall be accessible through a general purpose “escape” interface, where “escape” is used to indicate a data exchange that is outside the normal ICC Service Provider to ICC data exchange. The escape interface shall provide a means for an ICC Service Provider or an application to request a specific Service by supplying a Command Code parameter, and an optional variable-length data structure. The response to an escape request shall be an escape response specifying a completion code and optional variable-length data structure. These vendor-specific features can either be accessed through the slot logical device or by a specific functional logical device defined by the vendor.

3.2.2 Contact/Contactless Environment Specifics

3.2.2.1 Storage Card Functionality Support

This section defines commands for Storage Cards:

- The IFD subsystem has to, for Storage Cards that do not natively understand ISO 7816-4 APDUs, transform the APDUs into a specific command (-sequence). In other words, it has to filter or map requests and responses.

- The IFD subsystem must *not* filter or map any kind of APDU for cards that understand them since this would be a severe undesired limitation. Such cards are, for example ISO 14443-4 cards.

3.2.2.1.1 General Aspects

If an IFD implements synchronous protocols, it must implement both types 1 and 2 as specified in ISO 7816-10. A device capable of operating ISO/IEC 14443.3 and / or ISO/IEC 15693 cards falls into the scope of this specification. In addition, an IFD can support other synchronous protocols such as I^2C or other contactless protocols / cards.

In this subchapter all these cards / protocols are referred to as *storage cards/ storage card protocol*.

The IFD Subsystem is responsible for mapping the APDUs that consist of Inter Industry commands, and the exposed data structures, to storage card commands and the associated data structures as defined for the particular storage card.

The available commands, data structures and the link protocol are closely dependent to the particular storage card. There is a wide variety from storage only cards without any security mechanisms to highly sophisticated cards providing cryptographic authentication schemes.

The commands are defined by the industry and are not covered by ISO or other standardization bodies. IFD vendors, who intend to implement support for storage cards, have to gather detailed information from the different ICC manufacturers.

Compliant IFD subsystems must support the following commands (see table 3-6):

- **Read Binary:** Read data from the card.
- **Update Binary**: Write data to the card.
- **Load Keys:** Load keys to the IFD system.
- **Autentication:** Perform an authentication between IFD and card.

For IC Cards with security code (CHV), the following additional commands must be supported:

- **Verify**

Furthermore, a command for retrieving the UID/PUPI/serial number of a storage card is included into this specification:

- **Get UID**: This command retrieves the UID of the card in the reader or current Slot Logical Device.

The data mapping of synchronous cards is indicated by four header bytes (H1-H4) which are returned during ATR to the IFD. The mapping is defined either:

implicitly, i.e. known for a specific card and/or application or

explicitly, i.e. compliant to the ISO 7816-10.

Le and Lc semantics are borrowed from ISO/IEC 7816-4.

The memory layout of most cards reflects either file or block-oriented alignment.

- For the first case, file, the storage card memory shall be exposed as a single transparent file, where *Offset* in the following table specifies the byte location:

  ByteAddress = MSB * 256 + LSB.

  This implies that the maximum storage capacity is 64 kByte.

- The second variant are cards with a block-oriented layout. For these types, *Offset* specifies the location of a *Block*, not a single byte.

  BlockAddress = MSB * 256 + LSB.

  The maximum number of blocks is therefore 65536. The length of data to be written depends on the size of the block.

In order to support a given storage card type, reader manufacturers must then implement in the IFD subsystem all the necessary logic to map these high level commands and parameters to the actual electrical signal sequence.

3.2.2.1.2 Common Error Code Definition

Table 3-7 Common Error Codes for Storage Cards

	SW1	SW2	Meaning
Error	'67'	'00'	Wrong length
	'68'	'00'	Class byte is not correct
	'6A'	'81'	Function not supported
	'6B'	'00'	Wrong parameter P1-P2

The error codes defined in the Table above are valid for all commands defined within this section. Moreover command specific errors have been introduced as required in individual subsections.

3.2.2.1.3 Get UID Command

This Get UID command will retrieve the UID or SNR or PUPI of the present card.

Table 3-8a: Get UID Command APDU

Command	Class	INS	P1	P2	Lc	Data in	Le
Get UID	0x00	0xCA	0x00	0x00	-	-	XX

Table 3-8b: Get UID Command Output

Data Out
UID + SW1 SW2

Le = 0x00, this means: Return full length of the UID (e.g. for ISO14443A single 4 bytes, double 7 bytes, triple 10 bytes, for ISO14443B 4 bytes PUPI, for 15693 8 bytes UID).

UID Format:

The format of the UID has to be the same for all IFD subsystem implementations:

- The UID has to be exposed as a string of the expected length. If the expected length is greater than the actual length the rest of the string has to be filled with *zero*-value padding bytes.
- No cast must be done over the UID or parts of it. For example, casting four bytes of the UID to a 32-bit Integer is illegal.
- The order of the bytes within the string must match the order of bytes received from the card during initial contact (e.g. Anti-collision). Consequently, the first byte received will be at index *zero*.
- The bit order of the string bytes must be such that the LSB (MSB) matches with the LSB (MSB) of the card-defined UID.

The following table introduces some examples of SW1SW2 and their meaning.

When the card does not support a UID (e.g. a 7816-10 contact card) the code ‘Function not supported’ will be returned.

Table 3-9: Get UID Command Error Codes

	SW1	SW2	Meaning
Warning	'62'	'82'	End of UID reached before Le bytes (Le is greater than UID Length) In this case the IFD subsystem has to insert zero-value padding bytes up until the length of the UID expected by the caller.
Error	'6C'	'XX'	Wrong length (wrong number Le; 'XX' encodes the exact number) if Le is less than the available UID length)

3.2.2.1.4 Load Keys Command

The 'Load keys' command will just load (write) the keys in the IFD's designated memory. The key will be of two different types; the reader key and the card key.

Reader Key: This key will be used to protect the transmission of secured data e.g. card key from the application to the reader. Example: The application may encrypt the data with one of the reader keys: It has to tell the reader that it has done so, and the number of the key used.

Card Key: This is the card specific key (e.g. for Mifare it is Mifare key). This key can be volatile or non-volatile.

The coding of the command provides the following mechanisms:

• Load keys in either container: Reader key container to be used for transmission protection and card key container for card authentication.

• Transmission of the loaded key in plain or encrypted using a key out of the reader key container: The key to be used is indicated in P1 by its number. P2 is indicating the address within the container, where the key shall be stored.

• The containers can be located in volatile or non-volatile memory.

Table 3-10a: Load Keys Command APDU

Command	Class	INS	P1	P2	Lc	Data In	Le
Load Keys	0x00	0x82	Key Structure	Key number	Key Length	Key	-

Table 3-10b: Load Keys Command Output

Data Out
SW1 SW2

P1 Structure:

Table 3-11: Load Keys Command Error Codes

b7	b6	b5	b4	b3	b2	b1	b0	Description
x								**0**: Card Key; **1** Reader Key
	X							**0**: Plain Transmission, **1**: Secured Transmission
		x						**0**: Keys are loaded into the IFD volatile memory **1**: Keys are loaded into the IFD non-volatile memory.
			x					RFU
				xxxx				If **b6** is **set**, it is the Reader Key number that has been used for the encryption, else it is ignored by the IFD.

					The maximum of 16-reader keys is possible. Typically an IFD uses two reader keys only.

P2 (Key Number):

The application does not need to know the storage location of the keys. Rather, it will just submit the key type (reader or card key) and key number. The IFD decides where and how to store the keys.

The following table introduces some examples of SW1SW2 and their meaning.

Table 3-12: Load Keys Error Codes

	SW1	SW2	Meaning
Warning	'63'	'00'	No information is given
Error		'82'	Card key not supported
		'83'	Reader key not supported
		'84'	Plain transmission not supported
		'85'	Secured transmission not supported
		'86'	Volatile memory is not available
		'87'	Non volatile memory is not available
		'88'	Key number not valid
		'89'	Key length is not correct

3.2.2.1.5 Authenticate Command

The application provides the number of the key used for the authentication. The specific key must be already in the reader.

Table 3-13a Authenticate Command APDU

Command	Class	INS	P1	P2	P3	Data In	Le
Authenticate	0x00	0x88	Address MSB	Address LSB	Key Type	Key Nr.	-

Table 3-13b: Authenticate Command Output

Data Out
SW1 SW2

P1 & P2 (Address):

These items represent the block number or the starting byte number of the card to be authenticated.

P3 (Key Type):

The type of the key: E.g. for Mifare KEY_A (0x60) or KEY_B (0x61), for PicoTag and PicoPass Debit_key or Credit_key.

Key Nr.:

The card key number, which will be used for this authentication

The following table introduces some examples of SW1SW2 and their meaning:

Table 3-14: Authenticate Command Error Codes

	SW1	SW2	Meaning
Warning	'63'	'00'	No information is given
Error	'65'	'81'	Memory failure, addressed by P1-P2 is does not exist
	'69'	'82'	Security status not satisfied
		'83'	Authentication cannot be done
		'84'	Reference key not useable
		'82'	Key type not known
		'88'	Key number not valid

3.2.2.1.6 Verify Command

The PIN or password transmitted from the application to the reader can be also encrypted: The 'Load Keys' command explains the procedure.

Table 3-15a: Verify Command APDU

Command	Class	INS	P1	P2	Lc	Data In	Le
Verify	0x00	0x20	PIN structure	Reference data	PIN length	PIN	-

Table 3-15b: Verify Command Output

Data Out
SW1 SW2

Table 3-16: Verify Command P1 Definition

b7	b6	b5	b4	b3	b2	b1	b0	Description
x								RFU
	x							**0**: Plain transmission, **1**: Secured transmission
		x						RFU
			x					RFU
				xxxx				If **b6** is **set**, it is the Reader Key number that has been used for the encryption, else it is ignored by IFD. The maximum of 16-reader keys is possible. Typically an IFD uses two reader keys only.

Table 3-17: Verify Command Error Codes

	SW1	SW2	Meaning
Warning	'63'	'00' 'CX'	No information is given. Counter (verification failed; 'X' encodes the number of further allowed retries).
Error	'65'	'81'	Memory failure (unsuccessful storing).
	'69'	'82' '83' '84'	Security status not satisfied. Verify method blocked. Reference data not usable.
	'6A'	'88'	Reference data not found.

3.2.2.1.7 Read Binary Command

If the Le field contains only bytes set to '00', then all the bytes until the end of the file shall be read within the limit of 256 for a short Le field or 65 536 for an extended Le field.

Table 3-18a: Read Binary Command APDU

Command	Class	INS	P1	P2	Lc	Data in	Le
Read Binary	0x00	0xB0	Address MSB	Address LSB	-	-	XX

Table 3-18b: Read Binary Command Output

Data Out
Data + SW1 SW2

Table 3-19: Read Binary Command Error Codes

	SW1	SW2	Meaning
Warning	'62'	'81' '82'	Part of returned data may be corrupted. End of file reached before reading expected number of bytes.
Error	'69'	'81' '82' '86'	Command incompatible. Security status not satisfied. Command not allowed.
	'6A'	'81' '82'	Function not supported. File not found / Addressed block or byte does not exist.
	'6C'	'XX'	Wrong length (wrong number Le; 'XX' is the exact number).

3.2.2.1.8 Update Binary Command

Table 3-20a: Update Binary Command APDU

Command	Class	INS	P1	P2	Lc	Data in	Le
Read Binary	0x00	0xD6	Address MSB	Address LSB	XX	Data	-

Table 3-20b: Update Binary Command Output

Data Out
SW1 SW2

Table 3-21: Update Binary Command Error Codes

	SW1	SW2	Meaning
Warning	'62'	'81' '82'	A part of the returned data may be corrupted. End of file reached before writing Lc bytes.
Error	'65'	'81'	Memory failure (unsuccessful writing).
	'69'	'81' '82' '86'	Command incompatible. Security status not satisfied. Command not allowed.
	'6A'	'81' '82'	Function not supported. File not found / Addressed block or byte does not exist.

3.2.2.2 Mechanical Characteristics

Interface Devices may optionally support different mechanical characteristics such as ICC swallowing and ejection. Some Interface Devices may also offer the possibility of confiscating the card (this would typically be the case for PC-based ATM's).

The IFD Handler Interface will expose the necessary entry points enabling an ICC Service Provider to trigger those mechanisms.

3.2.2.3 Security Assurance Devices

The IFD subsystem may optionally implement various Security Assurance features. This may include:

- Isolated numeric keypad (that is, PIN pad) for entry of numeric user authentication data presented to the ICC.
- Isolated keyboard for entry of alphanumeric authentication data presented to the ICC.
- Isolated biometric measurement device for user authentication to the ICC.
- Isolated display for presentation of ICC-generated security-critical data.

The IFD subsystem may also implement user authentication devices designed to deliver authenticating data to the PC and associated applications. These could include:

- Isolated numeric keypad (that is, PIN pad) for entry of numeric user authentication data presented to the ICC.
- Isolated keyboard for entry of alphanumeric authentication data presented to the ICC.
- Isolated biometric measurement device for user authentication to the ICC.

Note that the same type of data collection devices may be used to deliver authentication data to PC applications and to an ICC. Hence, compliant Interface Devices implementing such devices may support both functions. The routing of this data must be under the control of logic contained in the Interface Device such that security-critical operations cannot be circumvented by software running on the PC.

4 Logical Device Interface Reference

This appendix summarizes the features described in detail in Chapter 3 and describes, from a functional point of view, the different services a compliant slot or functional logical device interface must support. It focuses on the definition of the services and their required parameters. Implementations may alter the naming conventions and parameters as required to adapt to their environment. Exact values for the return codes of the different functions are also operating-system dependent and will not be defined in this appendix.

The methods common to all logical devices are detailed first, followed by the specific ones for slot logical devices.

4.1 Syntax

Each service will be described, using the following template:

```
RESPONSECODE Name_Of_Service (
   IN           DWORD        param1
   IN/OUT       BYTE[]       param2
   OUT          WORD         param3)
```

In this template:

- `RESPONSECODE` stands for a `DWORD` quantity.
- Each parameter is specified as either Incoming (`IN`), outgoing (`OUT`) or both (`IN/OUT`).
- Each parameter is data typed.

4.2 Common Methods

Both slot logical devices and functional logical devices that comply with revision 2.0 of this specification must support the following methods.


```
RESPONSECODE Device_List_Devices (
      OUT  DEVICE_DESCRIPTOR [] DeviceList  // list of all the device
                                            // descriptors exposed by the
                                            // given IFD
   )
```

The addressed logical device (slot or functional) lists all the descriptors of its sibling logical devices (slot and functional) exposed for the same physical device (IFD), including itself. This method is useful for an IFD Service Provider when present. It is via this method that the Resource Manager can get access to all the functional logical devices.

The DEVICE_DESCRIPTOR structure is defined as follows:

```
STRUCTURE {
      STR DeviceName;    // system specific name
      DWORD DeviceType; // device type
   } DEVICE_DESCRIPTOR;
```

where DeviceType can have one of the following values:

```
DEVICE_TYPE_SLOT       1

DEVICE_TYPE_FUNCTIONAL 2
```

***ResponseCode* can be one of the following:**

Device_Success:	the request was successful
Device_Error_Not_Supported:	function not supported

```
RESPONSECODE Device_Control (
   IN     DWORD ControlCode         // vendor-defined control code
   IN     BYTE[] InBuffer           // Input Data Buffer
   OUT    BYTE[] OutBuffer          // Output Data Buffer
   )
```

This method supports a direct and generic communication channel with any slot or functional logical device. It is the responsibility of the vendor to define *ControlCode* values and input/ouput data associated with these features.

***ResponseCode* can be one of the following:**

Device_Success:	the request was successful
Device_Error_Not_Supported:	function not supported
Device_Wrong_Parameter:	a parameter is not supported or invalid

4.3 Slot Logical Device Methods

The following functions target the slot logical device even though that, for the sake of compatibility with version 1.0, these functions are prefixed by “IFD_”.

4.3.1 Administrative Services

The slot logical device interface will expose the administrative services listed below. As described in paragraph 3.1.1, those services will appear to the calling ICC Service Provider as simple synchronous function calls. Given the low level of complexity required by those services, in terms of operating system features, they should be quite similar from one platform to another.

4.3.1.1 Retrieving an IFD Capability

```
RESPONSECODE     IFD_Get_Capabilities(
   IN     DWORD         Tag
   OUT    BYTE[]        Value
   )
```

Expected Behavior and Results

This instructs the IFD Handler to retrieve the value corresponding to the specified `Tag` parameter. This enables the calling application to retrieve any of the information described from the following TLV structures:

- Reader Capabilities (cf. Table 3-1)
- ICC Interface State (cf. Table 3-2)
- Protocol Parameters (cf. Table 3-3, 3-4, 3-5, 3-6)

ResponseCode can be one of the following:

`IFD_Success`	: Value was successfully retrieved.
`IFD_Error_Tag`	: Tag does not exist.

Support

Mandatory

4.3.1.2 Setting an IFD Capability

```
RESPONSECODE     IFD_Set_Capabilities(
   IN     DWORD         Tag
   OUT    BYTE[]        Value
   )
```

Expected Behavior and Results

The IFD Handler will attempt to set the parameter specified by `Tag` to `Value`. This function can be used by a calling service provider to set parameters such as the current IFSD, or to request an extension of the Block Waiting Time.

`ResponseCode` can be one of the following:

`IFD_Success`	: Parameter was successfully set.
`IFD_Error_Set_Failure`	: Operation failed.
`IFD_Error_Tag`	: Tag does not exist.
`IFD_Error_Value_Read_Only`	: The value cannot be modified.

Support

Mandatory

4.3.1.3 Protocol Information and Negotiation

```
RESPONSECODE     IFD_Set_Protocol_Parameters(
   IN     DWORD         ProtocolType
   IN     BYTE          SelectionFlags
   IN     BYTE          PPS1   // 7816: Encodes Clock Conversion and bit
                               //       duration factors
                               // 14443: most significant nibble ; DS
                               //      : least significant nibble; DR
   IN     BYTE          PPS2
   IN     BYTE          PPS3
   )
```

Expected Behavior and Results

This function should only be called by the ICC Resource Manager layer. A Service Provider simply specifies its preferred protocols and protocol parameters, if any, when connecting to an ICC, and lets the ICC Resource Manager check if it is able to negotiate one of the preferred protocols based on the ATR received from the ICC, and the IFD capabilities.

The protocol type parameter can be

- a list of protocol types, coded in the same way as for tag 0x0120 and 0x0126
- the special value IFD_DEFAULT_PROTOCOL (defined as 0x80000000)

`SelectionFlags` indicates which of the optional parameters (`PPS1`, `PPS2` and `PPS3`), if any, have to be negotiated and included in the PPS request. It is obtained by performing a bitwise OR operation on the following flags:

```
IFD_NEGOTIATE_PPS1 1
IFD_NEGOTIATE_PPS2 2
IFD_NEGOTIATE_PPS3 4
```

`ResponseCode` can be one of the following:

`IFD_Success`	: PTS succeeded.
`IFD_Error_PTS_Failure`	: PTS failed.
`IFD_Error_Not_supported`	: PTS not supported.
`IFD_Protocol_Not_supported`	: Protocol not supported.

Support

Required.

4.3.1.4 ICC Power Management

```
RESPONSECODE     IFD_Power_ICC (
   IN     WORD          ActionRequested
   )
```

Expected Behavior and Results

This function is used to power up, power down, or reset the ICC. The desired action is specified by the `ActionRequested` parameter. The following actions are permitted:

Table 3-22: Power Management Actions

Command	ISO/IEC 7816-compatible environment	Contactless environment
`IFD_POWER_UP` (deprecated)	Requests activation of the contact	No effect if a PICC is active in the slot logical device. Error condition otherwise.
`IFD_POWER_DOWN`	Requests deactivation of the contact	• ISO 14443-4: DESELECT the PICC. No card removal notification must be generated. Poll for the PICC at ISO 14443-3 level. Keep the slot logically occupied. • ISO 15693 / ISO 14443-3: No state change since these cards are only active during communication and halted otherwise. Do not generate a removal event.
`IFD_RESET`	Requests a warm reset of the ICC	• ISO 14443-4: DESELECT the PICC and reactivate it in the same slot logical device. No card removal notification must be generated. When already powered down (see previous): Reactivate the PICC. • ISO 15693 / 14443-3: No action required since these cards remain in HALT and are only reactivated for communication.
`IFD_COLD_RESET`	Requests a cold reset of the ICC	Same as above.

`ResponseCode` can be one of the following:

IFD_Success

`IFD_Error_Power_Action` : The requested action could not be carried out.

`IFD_Error_Not_supported` : One of the requested actions is not supported.

If the function reports success and the action requested was either a reset or a power-up, then the ATR returned by the card and the Protocol Parameters can be accessed through the `IFD_Get_Capabilities` function.

Support

Mandatory. .

4.3.1.5 Mechanical Characteristics

`RESPONSECODE IFD_Swallow_ICC ()`

Expected Behavior and Results

This function causes a mechanical swallow of the ICC, if the IFD supports such a feature.

`ResponseCode` can be one of the following:

`IFD_Success`	: Card successfully swallowed
`IFD_Error_Swallow`	: Card not swallowed
`IFD_Error_Not_supported`	: Function not supported


```
RESPONSECODE            IFD_Eject_ICC ()
```

Expected Behavior and Results

This function causes a mechanical ejection of the ICC, if the IFD supports such a feature.

ResponseCode can be one of the following:

`IFD_Success`	: Card successfully ejected
`IFD_Error_Eject`	: Card not ejected
`IFD_Error_Not_supported`	: Function not supported

```
RESPONSECODE     IFD_Confiscate_ICC ()
```

Expected Behavior and Results

This function causes the IFD to confiscate the ICC, if the IFD supports such a feature.

`ResponseCode` can be one of the following:

`IFD_Success`	: Card successfully confiscated
`IFD_Error_Confiscate`	: Card not confiscated
`IFD_Error_Not_supported`	: Function not supported

Support

The support for these three functions is optional.

4.3.2 Communication Services

The slot logical device Interface provides communication services in order to support the data exchange between the ICC and the ICC Service Provider. Unlike the administrative services, communication services may differ greatly in their implementation, from one platform to another, depending on the mechanisms provided by the OS to support asynchronous communications between processes. Detailed description of these services will be found in the reference designs provided by OS manufacturers.

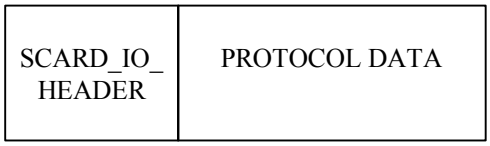
4.3.2.1 Data Exchange with the ICC

```
RESPONSECODE            IFD_Transmit_to_ICC (
      IN        BYTE []             CommandData
      OUT       BYTE []             ResponseData)
```

Expected Behavior and Results

This instructs the IFD Handler to send to the ICC the command specified in the *CommandData* parameter and return the response of the ICC in the *ResponseData* parameter.

1) The *CommandData* parameter is a binary array structured as follows :



where :

- SCARD_IO_HEADER is defined as follows :

```
STRUCTURE {
      DWORD Protocol;
      DWORD Length;
}SCARD_IO_HEADER;
```

- PROTOCOL DATA contains the data to be sent to the card. The Length parameter is the size in bytes of the SCARD_IO_HEADER structure.

2) *ResponseData* contains the reply to the command returned by the card

The following table shows the allowed values for the *Protocol* parameter according to the effective protocol agreed upon by the ICC and the IFD subsystem:

Table 3-23: Protocol Identifiers

Effective Protocol	Allowed Protocol parameter values
ISO/IEC 7816-3 T=0	SCARD_PROTOCOL_T0
ISO/IEC 7816-3 T=1	SCARD_PROTOCOL_T1 or SCARD_PROTOCOL_APPDATA
ISO/IEC 7816-10	SCARD_PROTOCOL_APPDATA
ISO/IEC 14443-4 / 14443-3, 15693	SCARD_PROTOCOL_T0, SCARD_PROTOCOL_T1, or SCARD_PROTOCOL_RAW

If the parameter does not match a valid entry in the table, the IFD handler must fail the call with the return code IFD_Communication_Error.

Provisions for SCARD_PROTOCOL_T0 communication

The *CommandData* needs to be formatted as an APDU as described in the ISO/IEC 7816-4. The following cases are supported:

- Case 1: Command with no incoming or outgoing data byte: CLA INS P1 P2
- Case 2: Command with outgoing data bytes: CLA INS P1 P2 Le
- Case 3: Command with incoming data bytes: CLA INS P1 P2 Lc [Data bytes]

Specific remarks:

- Only the short format of Lc and Le is supported, i.e. one byte long.
- An incoming order with no data byte « CLA INS P1 P2 00 » should be sent to the card as TPDU through a case 1 APDU.
- A case 4 APDU cannot be used. It is the responsibility of the application or the service provider to generate a « get response » if they need it.

The *ResponseData* parameter will contain the optional data returned by the ICC, followed by the 2 status words SW1-SW2.

Provisions for SCARD_PROTOCOL_T1 communication

The `CommandData` needs to be formatted as an APDU as described in the ISO/IEC 7816-4. The 4 general cases are supported:

- Case 1: Command with no incoming or outgoing data byte: CLA INS P1 P2
- Case 2: Command with outgoing data bytes: CLA INS P1 P2 Le
- Case 3: Command with incoming data bytes: CLA INS P1 P2 Lc [Data bytes]
- Case 4: Command with incoming and outgoing data bytes: CLA INS P1 P2 Lc [Data bytes] Le

The *ResponseData* parameter will contain the optional data returned by the ICC, followed by the 2 status words SW1-SW2.

Provisions for SCARD_PROTOCOL_APPDATA / RAW communication

It is assumed that the IFD subsystem and the ICC have negotiated a protocol that can transport arbitrary application data. It is the IFD subsystem responsibility to transport the command to the ICC and obtain its response.

ResponseCode can be one of the following:

`IFD_Success` : the request was successfully sent to the ICC,

`IFD_Communication_Error` : the request could not be sent to the ICC.

`IFD_Response_TimeOut` : the IFD timed-out waiting the response from the ICC.

4.3.2.2 ICC Insertion and Removal

```
RESPONSECODE     IFD_Is_ICC_Present ()
```

Expected Behavior and Results

Asynchronously signals the insertion of an ICC in the Interface Device.

```
RESPONSECODE     IFD_Is_ICC_Absent ()
```

Expected Behavior and Results

Asynchronously signals the removal of the ICC from the Interface Device.

4.3.3 IFDSP Related Methods

The following methods apply to IFDs that support extended capabilities, as defined in part 9 of this specification. Implementation of these methods is required only by IFD subsystems that do support extended capabilities.

The repository of all information relative to the extended capabilities of an IFD is managed directly by the IFD subsystem itself. This relieves the Resource Manager from such duties and provides for more flexibility.

```
RESPONSECODE IFD_List_Contexts(
   OUT    ACID[] ApplicationContextList
   )
```

Returns the list of the application contexts by their ACIDs that a IFD supports

ResponseCode can be one of the following:

Device_Success:	the request was successful
Device_Error_Not_Supported:	function not supported

```
RESPONSECODE IFD_Is_Context_Supported (
   IN     ACID ApplicationContext   // Application Context
   OUT    BOOL SupportResult        // true if supported, false otherwise
   )
```

Returns if a particular application context referenced by its ACID is supported or not by the IFD.

ResponseCode can be one of the following:

Device_Success:	the request was successful
Device_Error_Not_Supported:	function not supported

```
RESPONSECODE IFD_GetIFDSP (
   OUT    GUID IFDSP_ID    // Unique identifier for the IFD Service Provider
   )
```

Returns the reference to the IFD Service Provider if the reader supports it.

ResponseCode can be one of the following:

Device_Success:	the request was successful
Device_Error_Not_Supported:	function not supported

```
RESPONSECODE IFD_ListInterfaces(
   OUT    IID[] Interfaces    // List of interfaces supported by the IFDSP
```


)

Lists the interfaces (referenced by GUID) which are supported by the IFD and exposed by its IFD Service Provider.

ResponseCode can be one of the following:

Device_Success:	the request was successful
Device_Error_Not_Supported:	function not supported

