INTERNATIONAL STANDARD

ISO/IEC 7816-4

First edition
1995-09-01

Information technology — Identification cards — Integrated circuit(s) cards with contacts —

Part 4:
Interindustry commands for interchange

Technologies de l'information — Cartes d'identification — Cartes à circuit(s) intégré(s) à contacts —

Partie 4: Commandes intersectorielles pour les échanges

Reference number
ISO/IEC 7816-4:1995(E)

Contents

Annexes

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 7816-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

ISO/IEC 7816 consists of the following parts, under the general title *Information technology — Identification cards — Integrated circuit(s) cards with contacts*.

— *Part 1 : Physical characteristics,*

— *Part 2 : Dimensions and location of the contacts,*

— *Part 3 : Electronic signals and transmission protocols,*

— *Part 4 : Interindustry commands for interchange,*

— *Part 5 : Numbering system and registration procedure for application identifiers,*

— *Part 6 : Interindustry data elements.*

Annexes A and B form an integral part of this part of ISO/IEC 7816. Annexes C, D, E and F are for information only.

Introduction

This part of ISO/IEC 7816 is one of a series of standards describing the parameters for integrated circuit(s) cards with contacts and the use of such cards for international interchange.

These cards are identification cards intended for information exchange negotiated between the outside and the integrated circuit in the card. As a result of an information exchange, the card delivers information (computation results, stored data), and/or modifies its content (data storage, event memorization).

Information technology — Identification cards — Integrated circuit(s) cards with contacts —

Part 4:
Interindustry commands for interchange

1 Scope

This part of ISO/IEC 7816 specifies

— the content of the messages, commands and responses, transmitted by the interface device to the card and conversely,

— the structure and content of the historical bytes sent by the card during the answer to reset,

— the structure of files and data, as seen at the interface when processing interindustry commands for interchange,

— access methods to files and data in the card,

— a security architecture defining access rights to files and data in the card,

— methods for secure messaging,

— access methods to the algorithms processed by the card. It does not describe these algorithms.

It does not cover the internal implementation within the card and/or the outside world.

It allows further standardization of additional interindustry commands and security architectures.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 7816. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 7816 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 3166: 1993, *Codes for the representation of names of countries.*

ISO/IEC 7812-1 : 1993, *Identification cards — Identification of issuers — Part 1 : Numbering system.*

ISO/IEC 7816-3 : 1989, *Identification cards — Integrated circuit(s) cards with contacts — Part 3: Electronic signals and transmission protocols.*

Amendment 1 : 1992 to ISO/IEC 7816-3 : 1989, *Protocol type T=1, asynchronous half duplex block transmission protocol.*

Amendment 2 : 1994 to ISO/IEC 7816-3 : 1989, *Revision of protocol type selection.*

ISO/IEC 7816-5 : 1994, *Identification cards — Integrated circuit(s) cards with contacts — Part 5 : Numbering system and registration procedure for application identifiers.*

ISO/IEC 7816-6 : —[1], *Identification cards — Integrated circuit(s) cards with contacts — Part 6 : Interindustry data elements.*

ISO/IEC 8825 : 1990 [2], *Information technology — Open Systems Interconnection — Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).*

ISO/IEC 9796 : 1991, *Information technology — Security techniques — Digital signature scheme giving message recovery.*

ISO/IEC 9797 : 1994, *Information technology — Security techniques — Data integrity mechanism using a cryptographic check function employing a block cipher algorithm.*

ISO/IEC 9979 : 1991, *Data cryptographic techniques — Procedures for the registration of cryptographic algorithms.*

ISO/IEC 10116:1991, *Information technology — Modes of operation for an n-bit block cipher algorithm.*

ISO/IEC 10118-1 : 1994, *Information technology — Security techniques — Hash-functions — Part 1 : General.*

ISO/IEC 10118-2 : 1994, *Information technology — Security techniques — Hash-functions — Part 2 : Hash-functions using an n-bit block cipher algorithm.*

3 Definitions

For the purposes of this part of ISO/IEC 7816, the following definitions apply.

3.1 Answer-to-Reset file: Elementary file which indicates operating characteristics of the card.

3.2 command-response pair: Set of two messages: a command followed by a response.

3.3 data unit: The smallest set of bits which can be unambiguously referenced.

3.4 data element: Item of information seen at the interface for which are defined a name, a description of logical content, a format and a coding.

3.5 data object: Information seen at the interface which consists of a tag, a length and a value (i.e., a data element). In this part of ISO/IEC 7816, data objects are referred to as BER-TLV, COMPACT-TLV and SIMPLE-TLV data objects.

3.6 dedicated file: File containing file control information and, optionally, memory available for allocation. It may be the parent of EFs and/or DFs.

3.7 DF name: String of bytes which uniquely identifies a dedicated file in the card.

3.8 directory file: Elementary file defined in part 5 of ISO/IEC 7816.

3.9 elementary file: Set of data units or records which share the same file identifier. It cannot be the parent of another file.

3.10 file control parameters: Logical, structural and security attributes of a file.

3.11 file identifier: A 2-bytes binary value used to address a file.

3.12 file management data: Any information about a file except the file control parameters (e.g., expiration date, application label).

3.13 internal elementary file: Elementary file for storing data interpreted by the card.

3.14 master file: The mandatory unique dedicated file representing the root of the file structure.

3.15 message: String of bytes transmitted by the interface device to the card or vice-versa, excluding transmission-oriented characters as defined in part 3 of ISO/IEC 7816.

3.16 parent file: The dedicated file immediately preceding a given file within the hierarchy.

3.17 password: Data which may be required by the application to be presented to the card by its user.

3.18 path: Concatenation of file identifiers without delimitation. If the path starts with the identifier of the master file, it is an absolute path.

3.19 provider: Authority who has or who obtained the right to create a dedicated file in the card.

3.20 record: String of bytes which can be handled as a whole by the card and referenced by a record number or by a record identifier.

3.21 record identifier: Value associated with a record that can be used to reference that record. Several records may have the same identifier within an elementary file.

3.22 record number: Sequential number assigned to each record which uniquely identifies the record within its elementary file.

3.23 working elementary file: Elementary file for storing data not interpreted by the card.

---
1) To be published.
2) Currently under revision.

4 Abbreviations and notation

For the purposes of this part of ISO/IEC 7816, the following abbreviations apply.

APDU	Application protocol data unit
ATR	Answer to reset
BER	Basic encoding rules of ASN.1 (see annex D)
CLA	Class byte
DIR	Directory
DF	Dedicated file
EF	Elementary file
FCI	File control information
FCP	File control parameter
FMD	File management data
INS	Instruction byte
MF	Master file
P1-P2	Parameter bytes
PTS	Protocol type selection
RFU	Reserved for future use
SM	Secure messaging
SW1-SW2	Status bytes
TLV	Tag, length, value
TPDU	Transmission protocol data unit

For the purposes of this part of ISO/IEC 7816, the following notation applies.

'0' to '9' and 'A' to 'F'	The sixteen hexadecimal digits
(B_1)	Value of byte B_1
$B_1 \parallel B_2$	Concatenation of bytes B_1 (the most significant byte) and B_2 (the least significant byte)
$(B_1 \parallel B_2)$	Value of the concatenation of bytes B_1 and B_2
#	Number

5 Basic organizations

5.1 Data structures

This clause contains information on the logical structure of data as seen at the interface, when processing interindustry commands for interchange. The actual storage location of data and structural information beyond what is described in this clause are outside the scope of ISO/IEC 7816.

5.1.1 File organization

This part of ISO/IEC 7816 supports the following two categories of files.

— Dedicated file (DF).

— Elementary file (EF).

The logical organization of data in a card consists of the following structural hierarchy of dedicated files.

— The DF at the root is called the master file (MF). The MF is mandatory.

— The other DFs are optional.

The following two types of EFs are defined.

— Internal EF — Those EFs are intended for storing data interpreted by the card, i.e., data analyzed and used by the card for management and control purposes.

— Working EF — Those EFs are intended for storing data not interpreted by the card, i.e., data to be used by the outside world exclusively.

Figure 1 illustrates an example of the logical file organization in a card.

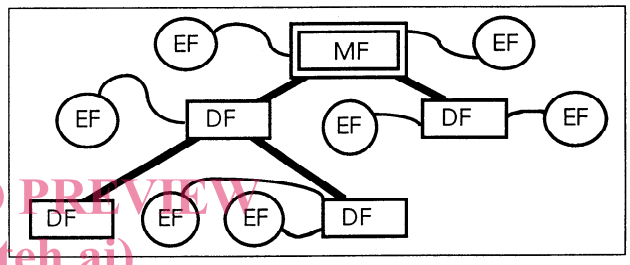

Figure 1 — Logical file organization (example)

5.1.2 File referencing methods

When a file cannot be implicitly selected, it shall be possible to select it by at least one of the following methods.

— **Referencing by file identifier** — Any file may be referenced by a file identifier coded on 2 bytes. If the MF is referenced by a file identifier, '3F00' shall be used (reserved value). The value 'FFFF' is reserved for future use. The value '3FFF' is reserved (see referencing by path). In order to select unambiguously any file by its identifier, all EFs and DFs immediately under a given DF shall have different file identifiers.

— **Referencing by path** — Any file may be referenced by a path (concatenation of file identifiers). The path begins with the identifier of the MF or of the current DF and ends with the identifier of the file itself. Between those two identifiers, the path consists of the identifiers of the successive parent DFs if any. The order of the file identifiers is always in the direction parent to child. If the identifier of the current DF is not known, the value '3FFF' (reserved value) can be used at the beginning of the path. The path allows an unambiguous selection of any file from the MF or from the current DF.

— **Referencing by short EF identifier** — Any EF may be referenced by a short EF identifier coded on 5 bits valued in the range from 1 to 30. The value 0 used as a short EF identifier references the currently selected EF. Short EF identifiers cannot be used in a path or as a file identifier (e.g., in a SELECT FILE command).

— **Referencing by DF name** — Any DF may be referenced by a DF name coded on 1 to 16 bytes. In order to select unambiguously by DF name (e.g., when selecting by means of application identifiers as defined in part 5 of ISO/IEC 7816), each DF name shall be unique within a given card.

5.1.3 Elementary file structures

The following structures of EFs are defined.

— Transparent structure — The EF is seen at the interface as a sequence of data units.

— Record structure — The EF is seen at the interface as a sequence of individually identifiable records.

The following attributes are defined for EFs structured in records.

— Size of the records : either fixed or variable.

— Organization of the records: either as a sequence (linear structure) or as a ring (cyclic structure).

The card shall support at least one of the following four methods for structuring EFs.

— Transparent EF.

— Linear EF with records of fixed size.

— Linear file with records of variable size.

— Cyclic EF with records of fixed size.

Figure 2 shows those four EF structures.

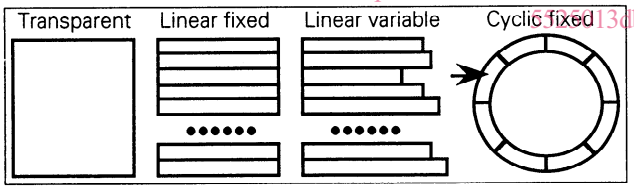

Figure 2 — EF structures

NOTE — The arrow on the figure references the most recently written record.

5.1.4 Data referencing methods

Data may be referenced as records, as data units or as data objects. Data is considered to be stored in a single continuous sequence of records (within an EF of record structure) or of data units (within an EF of transparent structure). Reference to a record or to a data unit outside an EF is an error.

Data referencing method, record numbering method and data unit size are EF-dependent features. The card can provide indications in the ATR, in the ATR file and in any file control information. When the card provides indications in several places, the indication valid for a given EF is the closest one to that EF within the path from the MF to that EF.

5.1.4.1 Record referencing

Within each EF of record structure, each record can be referenced by a record identifier and/or by a record number. Record identifiers and record numbers are unsigned 8-bit integers with values in the range from '01' to 'FE'. The value '00' is reserved for special purposes. The value 'FF' is RFU.

Referencing by record identifier shall induce the management of a record pointer. A reset of the card, a SELECT FILE and any command carrying a valid short EF identifier can affect the record pointer. Referencing by record number shall not affect the record pointer.

— **Referencing by record identifier** — Each record identifier is provided by an application. If a record is a SIMPLE-TLV data object in the data field of a message (see 5.4.4), then the record identifier is the first byte of the data object. Within an EF of record structure, records may have the same record identifier, in which case data contained in the records may be used for discriminating between them.

Each time a reference is made with a record identifier, an indication shall specify the logical position of the target record : the first or last occurrence, the next or previous occurrence relative to the record pointer.

— Within each EF of linear structure, the logical positions shall be sequentially assigned when writing or appending, i.e., in the order of creation. Therefore the first created record is in the first logical position.

— Within each EF of cyclic structure, the logical positions shall be sequentially assigned in the opposite order, i.e., the most recently created record is in the first logical position.

The following additional rules are defined for linear structures and for cyclic structures.

— The first occurrence shall be the record with the specified identifier and in the first logical position ; the last occurrence shall be the record with the specified identifier and in the last logical position.

— When there is no current record, the next occurrence shall be equivalent to the first occurrence ; the previous occurrence shall be equivalent to the last occurrence.

— When there is a current record, the next occurrence shall be the closest record with the specified identifier but in a greater logical position than the current record ; the previous occurrence shall be the closest record with the specified identifier but in a smaller logical position than the current record.

— The value '00' shall refer to the first, last, next or previous record in the numbering sequence, independently from the record identifier.

— **Referencing by record number** — Within each EF of record structure, the record numbers are unique and sequential.

— Within each EF of linear structure, the record numbers shall be sequentially assigned when writing or appending, i.e., in the order of creation. Therefore the first record (record number one, # 1) is the first created record.

— Within each EF of cyclic structure, the record numbers shall be sequentially assigned in the opposite order, i.e., the first record (record number one, # 1) is the most recently created record.

The following additional rule is defined for linear structures and for cyclic structures.

— The value '00' shall refer to the current record, i.e., that record fixed by the record pointer.

5.1.4.2 Data unit referencing

Within each EF of transparent structure, each data unit can be referenced by an offset (e.g., in READ BINARY command, see 6.1). It is an unsigned integer, limited to either 8 or 15 bits according to an option in the respective command. Valued to 0 for the first data unit of the EF, the offset is incremented by 1 for every subsequent data unit.

By default, i.e., if the card gives no indication, the size of the data unit is one byte.

NOTES

1 An EF of record structure may support data unit referencing and, in case it does, data units may contain structural information along with data, e.g., record numbers in a linear structure.

2 Within an EF of record structure, data unit referencing may not provide the intended result because the storage order of the records in the EF is not known, e.g., storage order in a cyclic structure.

5.1.4.3 Data object referencing

Each data object (as defined in 5.4.4) is headed by a tag which references it. Tags are specified in this part and other parts of ISO/IEC 7816.

5.1.5 File control information

The file control information (FCI) is the string of data bytes available in response to a SELECT FILE command. The file control information may be present for any file.

Table 1 introduces 3 templates intended for conveying file control information when coded as BER-TLV data objects.

— The FCP template is intended for conveying file control parameters (FCP), i.e., any BER-TLV data objects defined in table 2.

— The FMD template is intended for conveying file management data (FMD), i.e., BER-TLV data objects specified in other clauses of this part or in other parts of ISO/IEC 7816 (e.g., application label as defined in part 5 and application expiration date as defined in part 6).

— The FCI template is intended for conveying file control parameters and file management data.

Table 1 — Templates relevant to FCI

Tag	Value
'62'	File control parameters (FCP template)
'64'	File management data (FMD template)
'6F'	File control information (FCI template)

The 3 templates may be retrieved according to selection options of the SELECT FILE command (see table 59). If the FCP or FMD option is set, then the use of the corresponding template is mandatory. If the FCI option is set, then the use of the FCI template is optional.

Part of the file control information may additionally be present in a working EF under control of an application and referenced under tag '87'. The use of the FCP or FCI template is mandatory for the coding of file control information in such an EF.

File control information not coded according to this part of ISO/IEC 7816 may be introduced as follows.

— '00' or any value higher than '9F' — The coding of the subsequent string of bytes is proprietary.

— Tag = '53' — The value field of the data object consists of discretionary data not coded in TLV.

— Tag = '73' — The value field of the data object consists of discretionary BER-TLV data objects.

Table 2 — File control parameters

Tag	L	Value	Applies to
'80'	2	Number of data bytes in the file, excluding structural information	Transparent EFs
'81'	2	Number of data bytes in the file, including structural information if any	Any file
'82'	1	File descriptor byte (see table 3)	Any file
	2	File descriptor byte followed by data coding byte (see table 86)	Any file
	3 or 4	File descriptor byte followed by data coding byte and maximum record length	EFs with record structure
'83'	2	File identifier	Any file
'84'	1 to 16	DF name	DFs
'85'	var.	Proprietary information	Any file
'86'	var.	Security attributes (coding outside the scope of this part of ISO/IEC 7816)	Any file
'87'	2	Identifier of an EF containing an extension of the FCI	Any file
'88' to '9E'		RFU	
'9FXY'		RFU	

Table 3 — File descriptor byte

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	x	-	-	-	-	-	-	**File accessibility**
0	0	-	-	-	-	-	-	— Not shareable file
0	1	-	-	-	-	-	-	— Shareable file
0	-	x	x	x	-	-	-	**File type**
0	-	0	0	0	-	-	-	— Working EF
0	-	0	0	1	-	-	-	— Internal EF
0	-	0	1	0	-	-	-	— Reserved
0	-	0	1	1	-	-	-	for
0	-	1	0	0	-	-	-	proprietary
0	-	1	0	1	-	-	-	types
0	-	1	1	0	-	-	-	of EFs
0	-	1	1	1	-	-	-	— DF
0	-	-	-	-	x	x	x	**EF structure**
0	-	-	-	-	0	0	0	— No information given
0	-	-	-	-	0	0	1	— Transparent
0	-	-	-	-	0	1	0	— Linear fixed, no further info
0	-	-	-	-	0	1	1	— Linear fixed, SIMPLE-TLV
0	-	-	-	-	1	0	0	— Linear variable, no further info
0	-	-	-	-	1	0	1	— Linear variable, SIMPLE-TLV
0	-	-	-	-	1	1	0	— Cyclic, no further info
0	-	-	-	-	1	1	1	— Cyclic, SIMPLE-TLV
1	x	x	x	x	x	x	x	RFU

"Shareable" means that the file supports at least concurrent access on different logical channels.

5.2 Security architecture of the card

This clause describes the following features:

— security status,

— security attributes,

— security mechanisms.

Security attributes are compared with the security status to execute commands and/or to access files.

5.2.1 Security status

Security status represents the current state possibly achieved after completion of

— answer to reset (ATR) and possible protocol type selection (PTS) and/or

— a single command or a sequence of commands, possibly performing authentication procedures.

The security status may also result from the completion of a security procedure related to the identification of the involved entities, if any, e.g.

— by proving the knowledge of a password (e.g., using a VERIFY command),

— by proving the knowledge of a key (e.g., using a GET CHALLENGE command followed by an EXTERNAL AUTHENTICATE command).

— by secure messaging (e.g., message authentication).

Three security statuses are considered.

— Global security status — It may be modified by the completion of an MF-related authentication procedure (e.g., entity authentication by a password or by a key attached to the MF).

— File-specific security status — It may be modified by the completion of a DF-related authentication procedure (e.g., entity authentication by a password or by a key attached to the specific DF); it may be maintained, recovered or lost by file selection (see 6.10.2); this modification may be relevant only for the application to which the authentication procedure belongs.

— Command-specific security status — It only exists during the execution of a command involving authentication using secure messaging (see 5.6); such a command may leave the other security status unchanged.

If the concept of logical channels is applied, the file specific security status may depend on the logical channel (see 5.5.1).

5.2.2 Security attributes

The security attributes, when they exist, define the allowed actions and the procedures to be performed to complete such actions.

Security attributes may be associated with each file and fix the security conditions that shall be satisfied to allow operations on the file. The security attributes of a file depend on

— its category (DF or EF),

— optional parameters in its file control information and/or in that of its parent file(s).

NOTE — Security attributes may also be associated to other objects (e.g., keys).

5.2.3 Security mechanisms

This part of ISO/IEC 7816 defines the following security mechanisms.

— **Entity authentication with password** — The card compares data received from the outside world with secret internal data. This mechanism may be used for protecting the rights of the user.

— **Entity authentication with key** — The entity to be authenticated has to prove the knowledge of the relevant key in an authentication procedure (e.g., using a GET CHALLENGE command followed by an EXTERNAL AUTHENTICATE command).

— **Data authentication** — Using internal data, either secret or public, the card checks redundant data received from the outside world. Alternately, using secret internal data, the card computes a data element (cryptographic checksum or digital signature) and inserts it in the data sent to the outside world. This mechanism may be used for protecting the rights of a provider.

— **Data encipherment** — Using secret internal data, the card deciphers a cryptogram received in a data field. Alternately, using internal data, either secret or public, the card computes a cryptogram and inserts it in a data field, possibly together with other data. This mechanism may be used to provide a confidentiality service, e.g., for key management and conditional access. In addition to the cryptogram mechanism, data confidentiality can be achieved by data concealment. In this case, the card computes a string of concealing bytes and adds it by exclusive-or to data bytes received from or sent to the outside world. This mechanism may be used for protecting privacy and for reducing the possibilities of message filtering.

The result of an authentication may be logged in an internal EF according to the requirements of the application.

5.3 APDU message structure

A step in an application protocol consists of sending a command, processing it in the receiving entity and sending back the response. Therefore a specific response corresponds to a specific command, referred to as a command-response pair.

An application protocol data unit (APDU) contains either a command message or a response message, sent from the interface device to the card or conversely.

In a command-response pair, the command message and the response message may contain data, thus inducing four cases which are summarized by table 4.

Table 4 — Data within a command-response pair

Case	Command data	Expected response data
1	No data	No data
2	No data	Data
3	Data	No data
4	Data	Data

5.3.1 Command APDU

Illustrated by figure 3 (see also table 6), the command APDU defined in this part of ISO/IEC 7816 consists of

— a mandatory header of 4 bytes (CLA INS P1 P2),

— a conditional body of variable length.

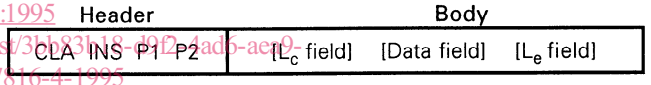

Figure 3 — Command APDU structure

The number of bytes present in the data field of the command APDU is denoted by L_c.

The maximum number of bytes expected in the data field of the response APDU is denoted by L_e (length of expected data). When the L_e field contains only zeroes, the maximum number of available data bytes is requested.

Figure 4 shows the 4 structures of command APDUs according to the 4 cases defined in table 4.

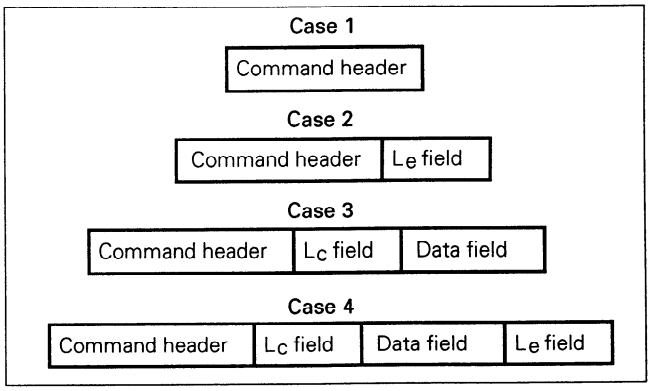

Figure 4 — The 4 structures of command APDUs

In case 1, the length L_c is null; therefore the L_c field and the data field are empty. The length L_e is also null; therefore the L_e field is empty. Consequently, the body is empty.

In case 2, the length L_c is null; therefore the L_c field and the data field are empty. The length L_e is not null; therefore the L_e field is present. Consequently, the body consists of the L_e field.

In case 3, the length L_c is not null; therefore the L_c field is present and the data field consists of the L_c subsequent bytes. The length L_e is null; therefore the L_e field is empty. Consequently, the body consists of the L_c field followed by the data field.

In case 4, the length L_c is not null; therefore the L_c field is present and the data field consists of the L_c subsequent bytes. The length L_e is also not null; therefore the L_e field is also present. Consequently, the body consists of the L_c field followed by the data field and the L_e field.

5.3.2 Decoding conventions for command bodies

In case 1, the body of the command APDU is empty. Such a command APDU carries no length field.

In cases 2, 3 and 4, the body of the command APDU consists of a string of L bytes denoted by B_1 to B_L as illustrated by figure 5. Such a body carries 1 or 2 length fields; B_1 is [part of] the first length field.

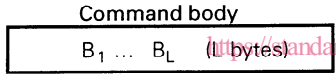

Figure 5 — Not empty body

In the card capabilities (see 8.3.6), the card states that, within the command APDU, the L_c field and the L_e field

— either shall be short (one byte, default value),

— or may be extended (explicit statement).

Consequently, the cases 2, 3 and 4 are either short (one byte for each length field) or extended (B_1 is valued to '00' and the value of each length is coded on 2 other bytes).

Table 5 shows the decoding of the command APDUs according to the four cases defined in table 4 and figure 4 and according to the possible extension of L_c and L_e.

Table 5 — Decoding of the command APDUs

Conditions			Case	
L = 0	-	-	1	
L = 1	-	-	Short 2	(2S)
L = 1+(B_1);	(B_1) ≠ 0;	-	Short 3	(3S)
L = 2+(B_1);	(B_1) ≠ 0;	-	Short 4	(4S)
L = 3;	(B_1) = 0;	-	Extended 2	(2E)
L = 3+(B_2 ‖ B_3);	(B_1) = 0;	(B_2 ‖ B_3) ≠ 0	Extended 3	(3E)
L = 5+(B_2 ‖ B_3);	(B_1) = 0;	(B_2 ‖ B_3) ≠ 0	Extended 4	(4E)

Any other command APDU is invalid.

Decoding conventions for L_e

If the value of L_e is coded on 1 (or 2) byte(s) where the bits are not all null, then the value of L_e is equal to the value of the byte(s) which lies in the range from 1 to 255 (or 65 535); the null value of all the bits means the maximum value of L_e: 256 (or 65 536).

The first 4 cases apply to all cards.

Case 1 — L = 0; the body is empty.

- No byte is used for L_c valued to 0.
- No data byte is present.
- No byte is used for L_e valued to 0.

Case 2S — L = 1.

- No byte is used for L_c valued to 0.
- No data byte is present.
- B_1 codes L_e valued from 1 to 256.

Case 3S — L = 1 + (B_1) and (B_1) ≠ 0.

- B_1 codes L_c (≠ 0) valued from 1 to 255.
- B_2 to B_L are the L_c bytes of the data field.
- No byte is used for L_e valued to 0.

Case 4S — L = 2 + (B_1) and (B_1) ≠ 0.

- B_1 codes L_c (≠ 0) valued from 1 to 255.
- B_2 to B_{L-1} are the L_c bytes of the data field.
- B_L codes L_e from 1 to 256.

For cards indicating the extension of L_c and L_e (see 8.3.6, card capabilities), the next 3 cases also apply.

Case 2E — L = 3 and (B_1) = 0.

- No byte is used for L_c valued to 0.
- No data byte is present.
- The L_e field consists of the 3 bytes where B_2 and B_3 code L_e valued from 1 to 65 536.

Case 3E — L = 3 + (B_2 ‖ B_3), (B_1) = 0 and (B_2 ‖ B_3) ≠ 0.

- The L_c field consists of the first 3 bytes where B_2 and B_3 code L_c (≠ 0) valued from 1 to 65 535.
- B_4 to B_L are the L_c bytes of the data field.
- No byte is used for L_e valued to 0.

Case 4E — L = 5 + (B_2 ‖ B_3), (B_1) = 0 and (B_2 ‖ B_3) ≠ 0.

- The L_c field consists of the first 3 bytes where B_2 and B_3 code L_c (≠ 0) valued from 1 to 65 535.
- B_4 to B_{L-2} are the L_c bytes of the data field.
- The L_e field consists of the last 2 bytes B_{L-1} and B_L which code L_e valued from 1 to 65 536.

For each transmission protocol defined in part 3 of ISO/IEC 7816, an annex attached to this part (one per protocol) specifies the transport of the APDUs of a command-response pair for each of the previous 7 cases.

5.3.3 Response APDU

Illustrated by figure 6 (see also table 7), the response APDU defined in this part of ISO/IEC 7816 consists of

— a conditional body of variable length,

— a mandatory trailer of 2 bytes (SW1 SW2).

Body	Trailer
[Data field]	SW1 SW2

Figure 6 — Response APDU structure

The number of bytes present in the data field of the response APDU is denoted by L_r.

The trailer codes the status of the receiving entity after processing the command-response pair.

NOTE — If the command is aborted, then the response APDU is a trailer coding an error condition on 2 status bytes.

5.4 Coding conventions for command headers, data fields and response trailers

Table 6 shows the contents of the command APDU.

Table 6 — Command APDU contents

Code	Name	Length	Description
CLA	Class	1	Class of instruction
INS	Instruction	1	Instruction code
P1	Parameter 1	1	Instruction parameter 1
P2	Parameter 2	1	Instruction parameter 2
L_c field	Length	variable 1 or 3	Number of bytes present in the data field of the command
Data field	Data	variable $= L_c$	String of bytes sent in the data field of the command
L_e field	Length	variable ≤ 3	Maximum number of bytes expected in the data field of the response to the command

Table 7 shows the contents of the response APDU.

Table 7 — Response APDU contents

Code	Name	Length	Description
Data field	Data	variable $= L_r$	String of bytes received in the data field of the response
SW1	Status byte 1	1	Command processing status
SW2	Status byte 2	1	Command processing qualifier

The subsequent clauses specify coding conventions for the class byte, the instruction byte, the parameter bytes, the data field bytes and the status bytes.

Unless otherwise specified, in those bytes, RFU bits are coded zero and RFU bytes are coded '00'.

5.4.1 Class byte

According to table 8 used in conjunction with table 9, the class byte CLA of a command is used to indicate

— to what extent the command and the response comply with this part of ISO/IEC 7816,

— and when applicable (see table 9), the format of secure messaging and the logical channel number.

Table 8 — Coding and meaning of CLA

Value	Meaning
'0X'	Structure and coding of command and response according to this part of ISO/IEC 7816 (for coding of 'X', see table 9)
'10' to '7F'	RFU
'8X', '9X'	Structure of command and response according to this part of ISO/IEC 7816. Except for 'X' (for coding, see table 9), the coding and meaning of command and response are proprietary
'AX'	Unless otherwise specified by the application context, structure and coding of command and response according to this part of ISO/IEC 7816 (for coding of 'X', see table 9)
'B0' to 'CF'	Structure of command and response according to this part of ISO/IEC 7816
'D0' to 'FE'	Proprietary structure and coding of command and response
'FF'	Reserved for PTS

Table 9 — Coding and meaning of nibble 'X' when CLA = '0X', '8X', '9X' or 'AX'

b4	b3	b2	b1	Meaning
x	x	-	-	Secure messaging (SM) format
0	x	-	-	• No SM or SM not according to 5.6
0	0	-	-	— No SM or no SM indication
0	1	-	-	— Proprietary SM format
1	x	-	-	• Secure messaging according to 5.6
1	0	-	-	— Command header not authenticated
1	1	-	-	— Command header authenticated (see 5.6.3.1 for command header usage)
-	-	x	x	Logical channel number (according to 5.5) (b2 b1 = 00 when logical channels are not used or when logical channel # 0 is selected)

5.4.2 Instruction byte

The instruction byte INS of a command shall be coded to allow transmission with any of the protocols defined in part 3 of ISO/IEC 7816. Table 10 shows the INS codes that are consequently invalid.

Table 10 — Invalid INS codes

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	x	x	x	x	x	1	— Odd values
0	1	1	0	x	x	x	x	— '6X'
1	0	0	1	x	x	x	x	— '9X'

Table 11 shows the INS codes defined in this part of ISO/IEC 7816. When the value of CLA lies within the range from '00' to '7F', the other values of INS codes are to be assigned by ISO/IEC JTC 1 SC17.

Table 11 — INS codes defined in this part of ISO/IEC 7816

Value	Command name	Clause
'0E'	ERASE BINARY	6.4
'20'	VERIFY	6.12
'70'	MANAGE CHANNEL	6.16
'82'	EXTERNAL AUTHENTICATE	6.14
'84'	GET CHALLENGE	6.15
'88'	INTERNAL AUTHENTICATE	6.13
'A4'	SELECT FILE	6.11
'B0'	READ BINARY	6.1
'B2'	READ RECORD(S)	6.5
'C0'	GET RESPONSE	7.1
'C2'	ENVELOPE	7.2
'CA'	GET DATA	6.9
'D0'	WRITE BINARY	6.2
'D2'	WRITE RECORD	6.6
'D6'	UPDATE BINARY	6.3
'DA'	PUT DATA	6.10
'DC'	UPDATE RECORD	6.8
'E2'	APPEND RECORD	6.7

5.4.3 Parameter bytes

The parameter bytes P1-P2 of a command may have any value. If a parameter byte provides no further qualification, then it shall be set to '00'.

5.4.4 Data field bytes

Each data field shall have one of the following three structures.

— Each TLV-coded data field shall consist of one or more TLV-coded data objects.

— Each non TLV-coded data field shall consist of one or more data elements, according to the specifications of the respective command.

— The structure of the proprietary-coded data fields is not specified in ISO/IEC 7816.

This part of ISO/IEC 7816 supports the following two types of TLV-coded data objects in the data fields.

— BER-TLV data object.

— SIMPLE-TLV data object.

ISO/IEC 7816 uses neither '00' nor 'FF' as tag value.

Each BER-TLV data object shall consist of 2 or 3 consecutive fields (see ISO/IEC 8825 and annex D).

— The tag field T consists of one or more consecutive bytes. It encodes a class, a type and a number.

— The length field consists of one or more consecutive bytes. It encodes an integer L.

— If L is not null, then the value field V consists of L consecutive bytes. If L is null, then the data object is empty : there is no value field.

Each SIMPLE-TLV data object shall consist of 2 or 3 consecutive fields.

— The tag field T consists of a single byte encoding only a number from 1 to 254 (e.g., a record identifier). It codes no class and no construction-type.

— The length field consists of 1 or 3 consecutive bytes. If the leading byte of the length field is in the range from '00' to 'FE', then the length field consists of a single byte encoding an integer L valued from 0 to 254. If the leading byte is equal to 'FF', then the length field continues on the two subsequent bytes which encode an integer L with a value from 0 to 65 535.

— If L is not null, then the value field V consists of L consecutive bytes. If L is null, then the data object is empty : there is no value field.

The data fields of some commands (e.g., SELECT FILE), the value fields of the SIMPLE-TLV data objects and the value fields of the some primitive BER-TLV data objects are intended for encoding one or more data elements.

The data fields of some other commands (e.g., record-oriented commands) and the value fields of the other primitive BER-TLV data objects are intended for encoding one or more SIMPLE-TLV data objects.

The data fields of some other commands (e.g., object-oriented commands) and the value fields of the constructed BER-TLV data objects are intended for encoding one or more BER-TLV data objects.

NOTE — Before, between or after TLV-coded data objects, '00' or 'FF' bytes without any meaning may occur (e.g., due to erased or modified TLV-coded data objects).

5.4.5 Status bytes

The status bytes SW1-SW2 of a response denote the processing state in the card. Figure 7 shows the structural scheme of the values defined in this part of ISO/IEC 7816.

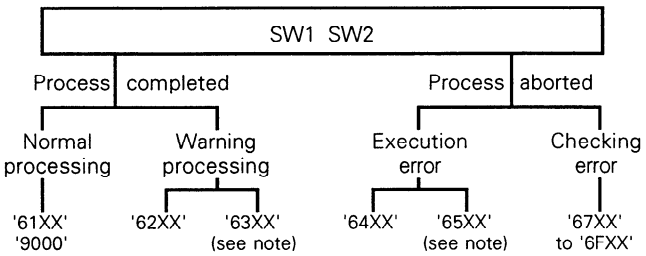

Figure 7 — Structural scheme of status bytes

NOTE — When SW1 = '63' or '65', the state of the non-volatile memory is changed. When SW1 = '6X' except '63' and '65', the state of the non-volatile memory is unchanged.

Due to specifications in part 3 of ISO/IEC 7816, this part does not define the following values of SW1-SW2:

— '60XX';

— '67XX', '6BXX', '6DXX', '6EXX', '6FXX', in each case if 'XX' ≠ '00';

— '9XXX', if 'XXX' ≠ '000'.

The following values of SW1-SW2 are defined whichever protocol is used (see examples in annex A).

— If a command is aborted with a response where SW1 = '6C', then SW2 indicates the value to be given to the short L_e field (exact length of requested data) when re-issuing the same command before issuing any other command.

— If a command (which may be of case 2 or 4, see table 4 and figure 4) is processed with a response where SW1 = '61', then SW2 indicates the maximum value to be given to the short L_e field (length of extra data still available) in a GET RESPONSE command issued before issuing any other command.

NOTE — A functionality similar to that offered by '61XX' may be offered at application level by '9FXX'. However, applications may use '9FXX' for other purposes.

Table 12 completed by tables 13 to 18 shows the general meanings of the values of SW1-SW2 defined in this part of ISO/IEC 7816. For each command, an appropriate clause provides more detailed meanings.

Tables 13 to 18 specify values of SW2 when SW1 is valued to '62', '63', '65', '68', '69' and '6A'. The values of SW2 not defined in tables 13 to 18 are RFU, except the values from 'F0' to 'FF' which are not defined in this part of ISO/IEC 7816.

Table 12 — Coding of SW1-SW2

SW1-SW2	Meaning
	Normal processings
'9000'	— No further qualification
'61XX'	— SW2 indicates the number of response bytes still available (see text below)
	Warning processings
'62XX'	— State of non-volatile memory unchanged (further qualification in SW2, see table 13)
'63XX'	— State of non-volatile memory changed (further qualification in SW2, see table 14)
	Execution errors
'64XX'	— State of non-volatile memory unchanged (SW2 = '00', other values are RFU)
'65XX'	— State of non-volatile memory changed (further qualification in SW2, see table 15)
'66XX'	Reserved for security-related issues (not defined in this part of ISO/IEC 7816)
	Checking errors
'6700'	— Wrong length
'68XX'	— Functions in CLA not supported (further qualification in SW2, see table 16)
'69XX'	— Command not allowed (further qualification in SW2, see table 17)
'6AXX'	— Wrong parameter(s) P1-P2 (further qualification in SW2, see table 18)
'6B00'	— Wrong parameter(s) P1-P2
'6CXX'	— Wrong length L_e: SW2 indicates the exact length (see text below)
'6D00'	— Instruction code not supported or invalid
'6E00'	— Class not supported
'6F00'	— No precise diagnosis

Table 13 — Coding of SW2 when SW1 = '62'

SW2	Meaning
'00'	No information given
'81'	Part of returned data may be corrupted
'82'	End of file /record reached before reading L_e bytes
'83'	Selected file invalidated
'84'	FCI not formatted according to 5.1.5

Table 14 — Coding of SW2 when SW1 = '63'

SW2	Meaning
'00'	No information given
'81'	File filled up by the last write
'CX'	Counter provided by 'X' (valued from 0 to 15) (exact meaning depending on the command)

Table 15 — Coding of SW2 when SW1 = '65'

SW2	Meaning
'00'	No information given
'81'	Memory failure