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.

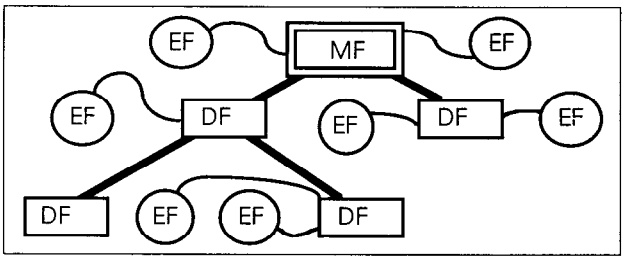


Figure 1 — Logical file organization (example)

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.

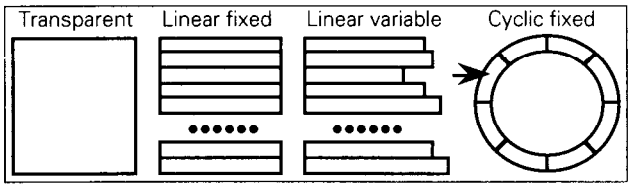


Figure 2 — EF structures

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.

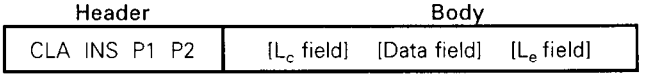


Figure 3 — Command APDU structure

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.

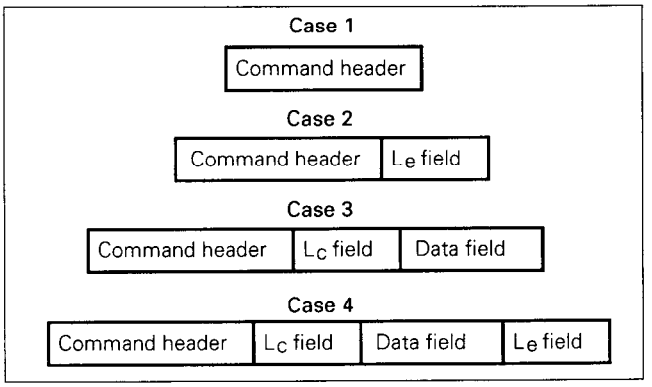


Figure 4 — The 4 structures of command APDUs

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.

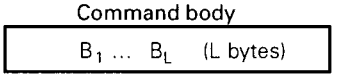


Figure 5 — Not empty body

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).

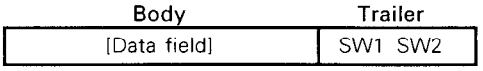


Figure 6 — Response APDU structure

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.

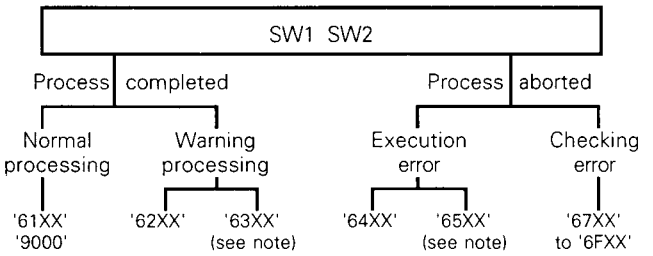


Figure 7 — Structural scheme of status bytes

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

Table 16 — Coding of SW2 when SW1 = '68'

SW2	Meaning
'00'	No information given
'81'	Logical channel not supported
'82'	Secure messaging not supported

Table 17 — Coding of SW2 when SW1 = '69'

SW2	Meaning
'00'	No information given
'81'	Command incompatible with file structure
'82'	Security status not satisfied
'83'	Authentication method blocked
'84'	Referenced data invalidated
'85'	Conditions of use not satisfied
'86'	Command not allowed (no current EF)
'87'	Expected SM data objects missing
'88'	SM data objects incorrect

Table 18 — Coding of SW2 when SW1 = '6A'

SW2	Meaning
'00'	No information given
'80'	Incorrect parameters in the data field
'81'	Function not supported
'82'	File not found
'83'	Record not found
'84'	Not enough memory space in the file
'85'	L_c inconsistent with TLV structure
'86'	Incorrect parameters P1-P2
'87'	L_c inconsistent with P1-P2
'88'	Referenced data not found

5.5 Logical channels

5.5.1 General concept

A logical channel, as seen at the interface, works as a logical link to a DF.

There shall be independence of activity on one logical channel from activity on another one. That is, command interdependencies on one logical channel shall be independent of command interdependencies on another logical channel. However, logical channels may share application-dependent security status and therefore may have security-related command interdependencies across logical channels (e.g., password verification).

Commands referring to a certain logical channel carry the respective logical channel number in the CLA byte (see tables 8 and 9). Logical channels are numbered from 0 to 3. If a card supports the logical channel mechanism, then the maximum number of available logical channels is indicated in the card capabilities (see 8.3.6).

Command-response pairs work as currently described. This part of ISO/IEC 7816 supports only command-response pairs which shall be completed before initiating a subsequent command-response pair. There shall be no interleaving of commands and their responses across logical channels; between the receipt of a command and the sending of the response to that command only one logical channel shall be active. When a logical channel is opened, it remains open until explicitly closed by a MANAGE CHANNEL command.

NOTES

1 More than one logical channel may be opened to the same DF, if not excluded (see file accessibility in 5.1.5).

2 More than one logical channel may select the same EF, if not excluded (see file accessibility in 5.1.5).

3 A SELECT FILE command on any logical channel will open a current DF and possibly a current EF. Therefore, there is one current DF and possibly one current EF per logical channel as a result of the behavior of the SELECT FILE command and file accessing commands using a short EF identifier.

5.5.2 Basic logical channel

The basic logical channel is permanently available. When numbered, its number is 0. When the class byte is coded according to tables 8 and 9, the bits b1 and b2 code the logical channel number.

5.5.3 Opening a logical channel

A logical channel is opened by successful completion of

— either the SELECT FILE command referencing a DF by assigning a logical channel number other than 0 in the class byte;

— or the open function of the MANAGE CHANNEL command either assigning a logical channel number other than 0 in the command APDU or requesting a logical channel number to be assigned by the card and returned in the response.

5.5.4 Closing a logical channel

The close function of the MANAGE CHANNEL command may be used to explicitly close a logical channel using the logical channel number. After closing, the logical channel number will be available for re-use. The basic logical channel shall not be closed.

5.6 Secure messaging

The goal of secure messaging (SM) is to protect [part of] the messages to and from a card by ensuring two basic security functions: data authentication and data confidentiality.

Secure messaging is achieved by applying one or more security mechanisms. Each security mechanism involves an algorithm, a key, an argument and often, initial data.

- The transmission and reception of data fields may be interleaved with the execution of security mechanisms. This specification does not preclude the determination by sequential analysis of which mechanisms and which security items shall be used for processing the remaining part of the data field.
- Two or more security mechanisms may use the same algorithm with different modes of operation (see ISO/IEC 10116). The present specifications of the padding rules do not preclude such a feature.

This clause defines 3 types of SM-related data objects:

— plain value data objects, intended for carrying plain data,

— security mechanism data objects, intended for carrying computational results of security mechanisms,

— auxiliary security data objects, intended for carrying control references and response descriptors.

5.6.1 SM format concept

In each message involving security mechanisms based on cryptography, the data field shall comply with the basic encoding rules of ASN.1 (see ISO/IEC 8825 and annex D), unless otherwise indicated by the class byte (see 5.4.1).

In the data field, the present SM format may be selected

— implicitly, i.e., known before issuing the command,

— explicitly, i.e., fixed by the class byte (see table 9).

The SM format defined in this part of ISO/IEC 7816 is BER-TLV coded.

- The context-specific class of tags (range from '80' to 'BF') is reserved for SM.
- Data objects of the other classes may be present (e.g., data objects of the application-specific class).
- Some SM-related data objects are recursive: their plain value field is still BER-TLV coded and there, the context-specific class is still reserved for SM.

In the context-specific class, the bit b1 of the tag fixes whether the SM-related data object shall (b1=1) or not (b1=0) be integrated in the computation of a data object for authentication. If present, the data objects of the other classes shall be integrated in such a computation.

5.6.2 Plain value data objects

Encapsulation is mandatory for data not coded in BER-TLV and for BER-TLV, including SM-related data objects. Encapsulation is optional for BER-TLV, not including SM-related data objects. Table 19 shows plain data objects for encapsulation.

Table 19 — Plain value data objects

Tag	Value
	The plain value consists of
'B0', 'B1'	— BER-TLV, including SM-related data objects
'B2', 'B3'	— BER-TLV, but not SM-related data objects
'80', '81'	— not BER-TLV-coded data
'99'	— SM status information (e.g., SW1-SW2)

5.6.3 Data objects for authentication

5.6.3.1 Cryptographic checksum data object

The computation of cryptographic checksums (see ISO/IEC 9797) involves an initial check block, a secret key and a block cipher algorithm that need not be reversible. The algorithm under control of the related key basically transforms a current input block of k bytes (typically 8 or 16) into a current output block of the same length.

The computation of a cryptographic checksum is performed in the following consecutive stages.

— **Initial stage** — The initial stage sets the initial check block which shall be one of the following blocks:

- the null block, i.e., k bytes valued to '00',
- the chaining block, i.e., a result from former computations, namely for a command, the final check block of the previous command and for a response, the final check block of the previous response.
- the initial value block provided e.g., by the outside world,
- the auxiliary block resulting from transforming auxiliary data under the related key. If the auxiliary data is less than k bytes, then it is headed by bits set to 0, up to the block length.

— **Sequential stage** — When table 9 is applicable (CLA = '0X', '8X', '9X' or 'AX'), if bits b4 and b3 of the class byte are set to 1, then the first data block consists of the header of the command APDU (CLA INS P1 P2) followed by one byte valued to '80' and k-5 bytes valued to '00'.

The cryptographic checksum shall integrate any SM-related data object having a tag where b1=1 and any data object with a tag outside the range from '80' to 'BF'. Those data objects shall be integrated data block by data block in the current check block. The splitting into data blocks shall be performed in the following way.

— The blocking shall be continuous at the border between adjacent data objects to be integrated.

— The padding shall apply at the end of each data object to be integrated followed either by a data object not to be integrated or by no further data object.

The padding consists of one mandatory byte valued to '80' followed, if needed, by 0 to k–1 bytes set to '00', until the respective data block is filled up to k bytes. Padding for authentication has no influence on transmission as the padding bytes shall not be transmitted.

The mode of operation is "cipher block chaining" (see ISO/IEC 10116). The first input is the exclusive-or of the initial check block with the first data block. The first output results from the first input. The current input is the exclusive-or of the previous output with the current data block. The current output results from the current input. The final check block is the last output.

— **Final stage** — The final stage extracts a cryptographic checksum (first m bytes, at least 4) from the final check block.

Table 20 shows the cryptographic checksum data object.

Table 20 — Cryptographic checksum data object

Tag	Value
'8E'	Cryptographic checksum (at least 4 bytes)

5.6.3.2 Digital signature data object

The digital signature computation is typically based upon asymmetric cryptographic techniques. There are two types of digital signatures:

— digital signature with appendix,

— digital signature giving message recovery.

The computation of a digital signature with appendix implies the use of a hash function (see ISO/IEC 10118). The data input either consists of the value of the digital signature input data object (see table 21), or is determined by the mechanism defined in 5.6.3.1.

The computation of a digital signature giving message recovery (see ISO/IEC 9796) does not imply the use of a hash function. However, according to the needs of the application, a hash code may be present as a part of the recovered message which may itself be BER-TLV-coded.

Table 21 shows digital signature related data objects.

Table 21 — Digital signature related data objects

Tag	Value
'9A', 'BA'	Digital signature input data
'9E'	Digital signature

5.6.4 Data objects for confidentiality

Data objects for confidentiality are intended for carrying a cryptogram which plain value consists of one of the following 3 cases:

— BER-TLV, including SM-related data objects,

— BER-TLV, not including SM-related data objects,

— not BER-TLV coded data.

Padding has to be indicated when the plain value consists of not BER-TLV coded data. When padding is applied but not indicated, the rules defined in 5.6.3.1 shall apply.

Table 22 shows the data objects for confidentiality.

Table 22 — Data objects for confidentiality

Tag	Value
'82', '83' '84', '85'	**Cryptogram, the plain value consisting of** — BER-TLV, including SM-related data objects — BER-TLV, but not SM-related data objects
'86', '87'	Padding indicator byte (see table 23) followed by cryptogram (plain value not coded in BER-TLV)

Every data object for confidentiality may use any cryptographic algorithm and any mode of operation, owing to an appropriate algorithm reference (see 5.6.5.1). In the absence of an algorithm reference and when no mechanism is implicitly selected for confidentiality, a default mechanism shall apply.

For the computation of a cryptogram which is preceded by the padding indicator, the default mechanism is a block cipher in "electronic code book" mode (see ISO/IEC 10116). The use of a block cipher may involve padding. Padding for confidentiality has an influence on transmission, the cryptogram (one or more blocks) is longer than the plain text.

Table 23 shows the padding indicator byte.

Table 23 — Padding indicator byte

Value	Meaning
'00'	— No further indication
'01'	— Padding as defined in 5.6.3.1
'02'	— No padding
'80' to '8E'	— Proprietary
	Other values are RFU

For the computation of a cryptogram not preceded by a padding indicator byte, the default mechanism is a stream cipher with an exclusive-or. In this case, the cryptogram is the exclusive-or of the string of data bytes to be concealed with a concealing string of the same length. Concealment thus requires no padding and the data objects concealed in the value field are recovered by the same operation.

5.6.5 Auxiliary security data objects

An algorithm, a key and, possibly, initial data may be selected for each security mechanism

— implicitly, i.e., known before issuing the command,

— explicitly, by control references nested in a control reference template.

Each command message may carry a response descriptor template fixing the data objects required in response. Inside the response descriptor, the security mechanisms are not yet applied ; the receiving entity shall apply them for constructing the response.

5.6.5.1 Control references

Table 24 shows the control reference templates.

Table 24 — Control reference templates

Tag	Meaning
'B4', 'B5' 'B6', 'B7' 'B8', 'B9'	— Template valid for cryptographic checksum — Template valid for digital signature — Template valid for confidentiality

The last possible position of a control reference template is just before the first data object to which the referred mechanism applies. For example, the last possible position of a template for cryptographic checksum is just before the first data object integrated in the computation.

Each control reference remains valid until a new control reference is provided for the same mechanism. For example, a command may fix control references for the next command.

Each control reference template is intended for carrying control reference data objects (see table 25) : an algorithm reference, a file reference, a key reference, an initial data reference and, only in a control reference template for confidentiality, a cryptogram contents reference.

The algorithm reference fixes an algorithm and its mode of operation (see ISO/IEC 9979 and 10116). Structure and coding of the algorithm reference are not defined in this part of ISO/IEC 7816.

The file reference denotes the file where the key reference is valid. If no file reference is present, then the key reference is valid in the current DF.

The key reference identifies the key to be used.

The initial data reference, when applied to cryptographic checksums, fixes the initial check block. If no initial data reference is present and no initial check block is implicitly selected, then the null block shall be used. Moreover, before transmitting the first data object for confidentiality using a stream cipher, a template for confidentiality shall provide auxiliary data for initializing the computation of the string of concealing bytes.

The cryptogram contents reference specifies the content of the cryptogram (e.g., secret keys, initial password, control words). The first byte of the value field is named the cryptogram descriptor byte and is mandatory. The range '00' to '7F' is RFU. The range '80' to 'FF' is proprietary.

Table 25 — Control reference data objects

Tag	Value
'80'	**Algorithm reference**
'81' '82'	**File reference** — file identifier or path — DF name
'83' '84'	**Key reference** — for direct use — for computing a session key
'85' '86' '87'	**Initial data reference** • **Initial check block** — L=0, null block — L=0, chaining block — L=0, previous initial value block plus one L=k, initial value block
'88' '89' to '8D'	• **Auxiliary data** — L=0, previous exchanged challenge plus one L≠0, no further indication — L=0, index of a proprietary data element L≠0, value of a proprietary data element
'8E'	**Cryptogram contents reference**

5.6.5.2 Response descriptor

The response descriptor template, if present in the data field of the command APDU, shall fix the structure of the corresponding response. Empty data objects shall list all data needed for producing the response.

The security items (algorithms, keys and initial data) used for processing the data field of a command message may be different from those used for producing the data field of the subsequent response message.

The following rules shall apply.

— The card shall fill each empty primitive data object.

— Each control reference template present in the response descriptor shall be present in the response at the same place with the same control references for algorithm, file and key. If the response descriptor provides auxiliary data, then the respective data object shall be empty in the response. If an empty reference data object for auxiliary data is present in the response descriptor, then it shall be full in the response.

— By the relevant security mechanisms, with the selected security items, the card shall produce all the requested security mechanism data objects.

Table 26 shows the response descriptor template.

Table 26 — Response descriptor template

Tag	Value
'BA', 'BB'	Response descriptor

5.6.6 SM status conditions

In any command using secure messaging, the following specific error conditions may occur.

— SW1 = '69' with SW2 =
• '87' : Expected SM data objects missing.
• '88' : SM data objects incorrect.

6 Basic interindustry commands

It shall not be mandatory for all cards complying to this part of ISO/IEC 7816 to support all the described commands or all the options of a supported command.

When international interchange is required, a set of card system services and related commands and options shall be used as defined in clause 9.

Table 11 provides a summary of the commands defined in this part of ISO/IEC 7816.

The impact of secure messaging (see 5.6) on the message structure is not described in this clause.

The list of error and warning conditions given in each clause 6.X.5 is not exhaustive (see 5.4.5).

6.1 READ BINARY command

6.1.1 Definition and scope

The READ BINARY response message gives [part of] the content of an EF with transparent structure.

6.1.2 Conditional usage and security

When the command contains a valid short EF identifier, it sets the file as current EF.

The command is processed on the currently selected EF. The command can be performed only if the security status satisfies the security attributes defined for this EF for the read function.

The command shall be aborted if it is applied to an EF without transparent structure.

6.1.3 Command message

Table 27 — READ BINARY command APDU

CLA	As defined in 5.4.1
INS	'B0'
P1-P2	See text below
L_c field	Empty
Data field	Empty
L_e field	Number of bytes to be read

If b8=1 in P1, then b7 and b6 of P1 are set to 0 (RFU bits), b5 to b1 of P1 are a short EF identifier and P2 is the offset of the first byte to be read in data units from the beginning of the file.

If b8=0 in P1, then P1 ll P2 is the offset of the first byte to be read in data units from the beginning of the file.

6.1.4 Response message (nominal case)

If the L_e field contains only zeroes, then within the limit of 256 for short length or 65 536 for extended length, all the bytes until the end of the file should be read.

Table 28 — READ BINARY response APDU

Data field	Data read (L_e bytes)
SW1-SW2	Status bytes

6.1.5 Status conditions

The following specific warning conditions may occur.

— SW1 = '62' with SW2 =
• '81' : Part of returned data may be corrupted.
• '82' : End of file reached before reading L_e bytes.

The following specific error conditions may occur.

— SW1 = '67' with SW2 =
• '00' : Wrong length (wrong L_e field).

— SW1 = '69' with SW2 =
• '81' : Command incompatible with file structure.
• '82' : Security status not satisfied.
• '86' : Command not allowed (no current EF).

— SW1 = '6A' with SW2 =
• '81' : Function not supported.
• '82' : File not found.

— SW1 = '6B' with SW2 =
• '00' : Wrong parameters (offset outside the EF).

— SW1 = '6C' with SW2 =
• 'XX' : Wrong length (wrong L_e field ; 'XX' indicates the exact length).

6.2 WRITE BINARY command

6.2.1 Definition and scope

The WRITE BINARY command message initiates the writing of binary values into an EF.

Depending upon the file attributes, the command shall perform one of the following operations:

— the logical OR of the bits already present in the card with the bits given in the command APDU (logical erased state of the bits of the file is 0),

— the logical AND of the bits already present in the card with the bits given in the command APDU (logical erased state of the bits of the file is 1),

— the one-time write in the card of the bits given in the command APDU.

When no indication is given in the data coding byte (see table 86), the logical OR behavior shall apply.

6.2.2 Conditional usage and security

When the command contains a valid short EF identifier, it sets the file as current EF.

The command is processed on the currently selected EF. The command can be performed only if the security status satisfies the security attributes for the write functions.

Once a WRITE BINARY has been applied to a data unit of a one-time write EF, any further write operation referring to this data unit will be aborted if the content of the data unit or the logical erased state indicator (if any) attached to this data unit is different from the logical erased state.

The command shall be aborted if it is applied to an EF without transparent structure.

6.2.3 Command message

Table 29 — WRITE BINARY command APDU

CLA	As defined in 5.4.1
INS	'D0'
P1-P2	See text below
L_C field	Length of the subsequent data field
Data field	String of data units to be written
L_e field	Empty

If b8=1 in P1, then b7 and b6 of P1 are set to 0 (RFU bits), b5 to b1 of P1 are a short EF identifier and P2 is the offset of the first byte to be written in data units from the beginning of the file.

If b8=0 in P1, then P1 ll P2 is the offset of the first byte to be written in data units from the beginning of the file.

6.2.4 Response message (nominal case)

Table 30 — WRITE BINARY response APDU

Data field	Empty
SW1-SW2	Status bytes

6.2.5 Status conditions

The following specific warning condition may occur.

— SW1 = '63' with SW2 =
• 'CX': Counter (successful writing, but after using an internal retry routine, 'X' ≠ '0' indicates the number of retries; 'X' = '0' means that no counter is provided).

The following specific error conditions may occur.

— SW1 = '65' with SW2 =
• '81': Memory failure (unsuccessful writing).

— SW1 = '67' with SW2 =
• '00': Wrong length (wrong L_C field).

— SW1 = '69' with SW2 =
• '81': Command incompatible with file structure.
• '82': Security status not satisfied.
• '86': Command not allowed (no current EF).

— SW1 = '6A' with SW2 =
• '81': Function not supported.
• '82': File not found.

— SW1 = '6B' with SW2 =
• '00': Wrong parameters (offset outside the EF).

6.3 UPDATE BINARY command

6.3.1 Definition and scope

The UPDATE BINARY command message initiates the update of the bits already present in an EF with the bits given in the command APDU.

6.3.2 Conditional usage and security

When the command contains a valid short EF identifier, it sets the file as current EF.

The command is processed on the currently selected EF. The command can be performed only if the security status satisfies the security attributes for the update function.

The command shall be aborted if it is applied to an EF without transparent structure.

6.3.3 Command message

Table 31 — UPDATE BINARY command APDU

CLA	As defined in 5.4.1
INS	'D6'
P1-P2	See text below
L_c field	Length of the subsequent data field
Data field	String of data units to be updated
L_e field	Empty

If b8=1 in P1, then b7 and b6 of P1 are set to 0 (RFU bits), b5 to b1 of P1 are a short EF identifier and P2 is the offset of the first byte to be updated in data units from the beginning of the file.

If b8=0 in P1, then P1 ll P2 is the offset of the first byte to be updated in data units from the beginning of the file.

6.3.4 Response message (nominal case)

Table 32 — UPDATE BINARY response APDU

Data field	Empty
SW1-SW2	Status bytes

6.3.5 Status conditions

The following specific warning condition may occur.

— SW1 = '63' with SW2 =
• 'CX' : Counter (successful updating, but after using an internal retry routine, 'X' ≠ '0' indicates the number of retries ; 'X' = '0' means that no counter is provided).

The following specific error conditions may occur.

— SW1 = '65' with SW2 =
• '81' : Memory failure (unsuccessful updating).

— SW1 = '67' with SW2 =
• '00' : Wrong length (wrong L_c field).

— SW1 = '69' with SW2 =
• '81' : Command incompatible with file structure.
• '82' : Security status not satisfied.
• '86' : Command not allowed (no current EF).

— SW1 = '6A' with SW2 =
• '81' : Function not supported.
• '82' : File not found.

— SW1 = '6B' with SW2 =
• '00' : Wrong parameters (offset outside the EF).

6.4 ERASE BINARY command

6.4.1 Definition and scope

The ERASE BINARY command message sets [part of] the content of an EF to its logical erased state, sequentially, starting from a given offset.

6.4.2 Conditional usage and security

When the command contains a valid short EF identifier, it sets the file as current EF.

The command is processed on the currently selected EF. The command can be performed only if the security status satisfies the security attributes for the erase function.

The command shall be aborted if it is applied to an EF without transparent structure.

6.4.3 Command message

Table 33 — ERASE BINARY command APDU

CLA	As defined in 5.4.1
INS	'0E'
P1-P2	See text below
L_c field	Empty or '02'
Data field	See text below
L_e field	Empty

If b8=1 in P1, then b7 and b6 of P1 are set to 0 (RFU bits), b5 to b1 of P1 are a short EF identifier and P2 is the offset of the first byte to be erased in data units from the beginning of the file.

If b8=0 in P1, then P1 ll P2 is the offset of the first byte to be erased in data units from the beginning of the file.

If the data field is present, it codes the offset of the first data unit not to be erased. This offset shall be higher than the one coded in P1-P2. When the data field is empty, the command erases up to the end of the file.

6.4.4 Response message (nominal case)

Table 34 — ERASE BINARY response APDU

Data field	Empty
SW1-SW2	Status bytes

6.4.5 Status conditions

The following specific warning condition may occur.

— SW1 = '63' with SW2 =
• 'CX' : Counter (successful erasing, but after using an internal retry routine, 'X' ≠ '0' indicates the number of retries ; 'X' = '0' means that no counter is provided).

The following specific error conditions may occur.

— SW1 = '65' with SW2 =
• '81' : Memory failure (unsuccessful erasing).

— SW1 = '67' with SW2 =
• '00' : Wrong length (wrong L_c field).

— SW1 = '69' with SW2 =
• '81' : Command incompatible with file structure.
• '82' : Security status not satisfied.
• '86' : Command not allowed (no current EF).

— SW1 = '6A' with SW2 =
• '81' : Function not supported.
• '82' : File not found.

— SW1 = '6B' with SW2 =
• '00' : Wrong parameters (offset outside the EF).

6.5 READ RECORD(S) command

6.5.1 Definition and scope

The READ RECORD(S) response message gives the contents of the specified record(s) [or the beginning part of one record] of an EF.

6.5.2 Conditional usage and security

The command can be performed only if the security status satisfies the security attributes for this EF for the read function.

If an EF is currently selected at the time of issuing the command, then this command may be processed without identification of this file.

When the command contains a valid short EF identifier, it sets the file as current EF and resets the current record pointer.

The command shall be aborted if applied to an EF without record structure.

6.5.3 Command message

Table 35 — READ RECORD(S) command APDU

CLA	As defined in 5.4.1
INS	'B2'
P1	Record number or record identifier of the first record to be read ('00' indicates the current record)
P2	Reference control, according to table 36
L_c field	Empty
Data field	Empty
L_e field	Number of bytes to be read

Table 36 — Coding of the reference control P2

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	-	-	-	— Currently selected EF
x	x	x	x	x	-	-	-	— Short EF identifier
		(not all equal)						
1	1	1	1	1	-	-	-	RFU
-	-	-	-	-	1	x	x	**Usage of record number in P1**
-	-	-	-	-	1	0	0	— Read record P1
-	-	-	-	-	1	0	1	— Read all records from P1 up to the last
-	-	-	-	-	1	1	0	— Read all records from the last up to P1
-	-	-	-	-	1	1	1	RFU
-	-	-	-	-	0	x	x	**Usage of record identifier in P1**
-	-	-	-	-	0	0	0	— Read first occurrence
-	-	-	-	-	0	0	1	— Read last occurrence
-	-	-	-	-	0	1	0	— Read next occurrence
-	-	-	-	-	0	1	1	— Read previous occurrence

6.5.4 Response message (nominal case)

If the L_e field contains only zeroes, then depending on b3b2b1 of P2 and within the limit of 256 for short length or 65 536 for extended length, the command should read completely

— either the single requested record,

— or the requested sequence of records.

Table 37 — READ RECORD(S) response APDU

Data field	L_r (may be equal to L_e) bytes, see table 38
SW1-SW2	Status bytes

When the records are SIMPLE-TLV data objects (see 5.4.4), table 38 illustrates the format of the data field of the response message.

Table 38-1 — Data field of the response when reading for one record

Case a — Partial read of one record

T_n 1 byte	L_n 1 or 3 bytes	First data bytes of the record

<---------- L_e bytes ---------->

This case applies when the L_e field does not contain only zeroes.

Case b — Complete read of one record

T_n 1 byte	L_n 1 or 3 bytes	Whole data bytes of the record L_n bytes

This case applies when the L_e field contains only zeroes.

Table 38-2 — Data field of the response when reading for several records

Case c — Partial read of a record sequence

Record # n $T_n \| L_n \| V_n$	... ...	First bytes of record # n+m $T_{n+m} \| L_{n+m} \| V_{n+m}$

<---------- L_e bytes ---------->

This case applies when the L_e field does not contain only zeroes.

Case d — Read multiple records up to the file end

Record # n $T_n \| L_n \| V_n$	... ...	Record # n+m $T_{n+m} \| L_{n+m} \| V_{n+m}$

This case applies when the L_e field contains only zeroes.

The comparison of the length of the data field with its TLV structure gives the nature of the data: the unique record (read one record) or the last record (read all records) is incomplete, complete or padded.

NOTE — If TLV coding is not used, then the read-all-records function results in receiving several records without standard delimitation of the records.

6.5.5 Status conditions

The following specific warning conditions may occur.

— SW1 = '62' with SW2 =
- '81' : Part of returned data may be corrupted.
- '82' : End of record reached before L_e bytes.

The following specific error conditions may occur.

— SW1 = '67' with SW2 =
- '00' : Wrong length (empty L_e field).

— SW1 = '69' with SW2 =
- '81' : Command incompatible with file structure.
- '82' : Security status not satisfied.

— SW1 = '6A' with SW2 =
- '81' : Function not supported.
- '82' : File not found.
- '83' : Record not found.

— SW1 = '6C' with SW2 =
- 'XX' : Wrong length (wrong L_e field ; 'XX' indicates the exact length).

6.6 WRITE RECORD command

6.6.1 Definition and scope

The WRITE RECORD command message initiates one of the following operations :

— the write once of a record ;

— the logical OR of the data bytes of a record already present in the card with the data bytes of the record given in the command APDU ;

— the logical AND of the data bytes of a record already present in the card with the data bytes of the record given in the command APDU.

When no indication is given in the data coding byte (see table 86), the logical OR operation shall apply.

When using current record addressing, the command shall set the record pointer on the successfully written record.

6.6.2 Conditional usage and security

The command can be performed only if the security status satisfies the security attributes for this EF for the write functions.

If an EF is currently selected at the time of issuing the command, then this command may be processed without identification of this file.

When the command contains a valid short EF identifier, it sets the file as current EF and resets the current record pointer.

The command shall be aborted if applied to an EF without record structure.

The "previous" option of the command (P2 = xxxxx011), applied to a cyclic file, has the same behavior as APPEND RECORD.

6.6.3 Command message

Table 39 — WRITE RECORD command APDU

CLA	As defined in 5.4.1
INS	'D2'
P1	P1= '00' designates the current record. P1≠ '00' is the number of the specified record.
P2	According to table 40
L_c field	Length of the subsequent data field
Data field	Record to be written
L_e field	Empty

Table 40 — Coding of the reference control P2

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	-	-	-	— Currently selected EF
x	x	x	x	x	-	-	-	— Short EF identifier
(not all equal)								
-	-	-	-	-	0	0	0	— First record
-	-	-	-	-	0	0	1	— Last record
-	-	-	-	-	0	1	0	— Next record
-	-	-	-	-	0	1	1	— Previous record
-	-	-	-	-	1	0	0	— Record number given in P1
Any other value								RFU

When the records are SIMPLE-TLV data objects (see 5.4.4), table 41 illustrates the format of the data field of the command message.

Table 41 — Data field of the command
Complete write of one record

T_n 1 byte	L_n 1 or 3 bytes	Whole data bytes of the record L_n bytes

6.6.4 Response message (nominal case)

Table 42 — WRITE RECORD response APDU

Data field SW1-SW2	Empty Status bytes

6.6.5 Status conditions

The following specific warning condition may occur.

— SW1 = '63' with SW2 =
• 'CX' : Counter (successful writing, but after using an internal retry routine, 'X' ≠ '0' indicates the number of retries ; 'X' = '0' means that no counter is provided).

The following specific error conditions may occur.

— SW1 = '65' with SW2 =
• '81' : Memory failure (unsuccessful writing).

— SW1 = '67' with SW2 =
• '00' : Wrong length (empty L_c field).

— SW1 = '69' with SW2 =
• '81' : Command incompatible with file structure.
• '82' : Security status not satisfied.
• '86' : Command not allowed (no current EF).

— SW1 = '6A' with SW2 =
• '81' : Function not supported.
• '82' : File not found.
• '83' : Record not found.
• '84' : Not enough memory space in the file.
• '85' : L_c inconsistent with TLV structure.

6.7 APPEND RECORD command

6.7.1 Definition and scope

The APPEND RECORD command message initiates either the appending of a record at the end of an EF of linear structure or the writing of record number 1 in an EF of cyclic structure (see 5.1.4).

The command shall set the record pointer on the successfully appended record.

6.7.2 Conditional usage and security

The command can be performed only if the security status satisfies the security attributes for this EF for the append function.

If an EF is currently selected at the time of issuing the command, then this command may be processed without identification of this file.

When the command contains a valid short EF identifier, it sets the file as current EF and resets the current record pointer.

The command shall be aborted if applied to an EF without record structure.

NOTE — If this command is applied to an EF of cyclic structure full of records, then the record with the highest record number is replaced. This record becomes record number 1.

6.7.3 Command message

Table 43 — APPEND RECORD command APDU

CLA	As defined in 5.4.1
INS	'E2'
P1	Only P1 = '00' is valid
P2	According to table 44
L_c field	Length of the subsequent data field
Data field	Record to be appended
L_e field	Empty

Table 44 — Coding of the reference control P2

b8 b7 b6 b5 b4 b3 b2 b1	Meaning
0 0 0 0 0 0 0 0	— Currently selected EF
x x x x x 0 0 0 (not all equal)	— Short EF identifier
Any other value	RFU

When the records are SIMPLE-TLV data objects (see 5.4.4), table 45 illustrates the format of the data field of the command message.

Table 45 — Data field of the command
Complete append of one record

T_n 1 byte	L_n 1 or 3 bytes	Whole data bytes of the record L_n bytes

6.7.4 Response message (nominal case)

Table 46 — APPEND RECORD response APDU

Data field SW1-SW2	Empty Status bytes

6.7.5 Status conditions

The following specific warning condition may occur.

— SW1 = '63' with SW2 =
• 'CX' : Counter (successful appending, but after using an internal retry routine, 'X' ≠ '0' indicates the number of retries ; 'X' = '0' means that no counter is provided).

The following specific error conditions may occur.

— SW1 = '65' with SW2 =
• '81' : Memory failure (unsuccessful appending).

— SW1 = '67' with SW2 =
• '00' : Wrong length (empty L_c field).

— SW1 = '69' with SW2 =
• '81' : Command incompatible with file structure.
• '82' : Security status not satisfied.
• '86' : Command not allowed (no current EF).

— SW1 = '6A' with SW2 =
• '81' : Function not supported.
• '82' : File not found.
• '84' : Not enough memory space in the file.
• '85' : L_c inconsistent with TLV structure.

6.8 UPDATE RECORD command

6.8.1 Definition and scope

The UPDATE RECORD command message initiates the updating of a specific record with the bits given in the command APDU.

When using current record addressing, the command shall set the record pointer on the successfully updated record.

6.8.2 Conditional usage and security

The command can be performed only if the security status satisfies the security attributes for this EF for the update function.

If an EF is currently selected at the time of issuing the command, then this command may be processed without identification of this file.

When the command contains a valid short EF identifier, it sets the file as current EF and resets the current record pointer.

The command shall be aborted if applied to an EF without record structure.

When the command applies to an EF with linear fixed or cyclic structure, then it shall be aborted if the record length is different from the length of the existing record.

When the command applies to an EF with linear variable structure, then it may be carried out when the record length is different from the length of the existing record.

The "previous" option of the command (P2 = xxxxx011), applied to a cyclic file, has the same behavior as APPEND RECORD.

6.8.3 Command message

Table 47 — UPDATE RECORD command APDU

CLA	As defined in 5.4.1
INS	'DC'
P1	P1 = '00' designates the current record P1 ≠ '00' is the number of the specified record
P2	According to table 48
L_c field	Length of the subsequent data field
Data field	Record to be updated
L_e field	Empty

Table 48 — Coding of the reference control P2

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	-	-	-	— Currently selected EF
x	x	x	x	x	-	-	-	— Short EF identifier (not all equal)
-	-	-	-	-	0	0	0	— First record
-	-	-	-	-	0	0	1	— Last record
-	-	-	-	-	0	1	0	— Next record
-	-	-	-	-	0	1	1	— Previous record
-	-	-	-	-	1	0	0	— Record number given in P1
Any other value								RFU

When the records are SIMPLE-TLV data objects (see 5.4.4), table 49 illustrates the format of the data field of the command message.

Table 49 — Data field of the command
Complete update of one record

T_n 1 byte	L_n 1 or 3 bytes	Whole data bytes of the record L_n bytes

6.8.4 Response message (nominal case)

Table 50 — UPDATE RECORD response APDU

Data field	Empty
SW1-SW2	Status bytes

6.8.5 Status conditions

The following specific warning condition may occur.

— SW1 = '63' with SW2 =
• 'CX' : Counter (successful updating, but after using an internal retry routine, 'X' ≠ '0' indicates the number of retries ; 'X' = '0' means that no counter is provided).

The following specific error conditions may occur.

— SW1 = '65' with SW2 =
• '81' : Memory failure (unsuccessful updating).

— SW1 = '67' with SW2 =
• '00' : Wrong length (empty L_c field).

— SW1 = '69' with SW2 =
• '81' : Command incompatible with file structure.
• '82' : Security status not satisfied.
• '86' : Command not allowed (no current EF).

— SW1 = '6A' with SW2 =
• '81' : Function not supported.
• '82' : File not found.
• '83' : Record not found.
• '84' : Not enough memory space in the file.
• '85' : L_C inconsistent with TLV structure.

6.9 GET DATA command

6.9.1 Definition and scope

The GET DATA command is used for the retrieval of one primitive data object, or the retrieval of one or more data objects contained in a constructed data object, within the current context (e.g., application-specific environment or current DF).

6.9.2 Conditional usage and security

The command can be performed only if the security status satisfies the security conditions defined by the application within the context for the function.

6.9.3 Command message

Table 51 — GET DATA command APDU

CLA	As defined in 5.4.1
INS	'CA'
P1-P2	See table 52
L_C field	Empty
Data field	Empty
L_e field	Number of bytes expected in response

Table 52 — Coding of the parameters P1-P2

Value	Meaning
'0000' to '003F'	RFU
'0040' to '00FF'	BER-TLV tag (1 byte) in P2
'0100' to '01FF'	Application data (proprietary coding)
'0200' to '02FF'	SIMPLE-TLV tag in P2
'0300' to '3FFF'	RFU
'4000' to 'FFFF'	BER-TLV tag (2 bytes) in P1-P2

Get application data

• When the value of P1-P2 lies in the range from '0100' to '01FF', the value of P1-P2 shall be an identifier reserved for card internal tests and for proprietary services meaningful within a given application context.

Get data objects

• When the value of P1-P2 lies in the range from '0040' to '00FF', the value of P2 shall be a BER-TLV tag on a single byte. The value '00FF' is reserved for obtaining all the common BER-TLV data objects readable in the context.

• When the value of P1-P2 lies in the range from '0200' to '02FF', the value of P2 shall be a SIMPLE-TLV tag. The value '0200' is RFU. The value '02FF' is reserved for obtaining all the common SIMPLE-TLV data objects readable in the context.

• When the value of P1-P2 lies in the range from '4000' to 'FFFF', the value of P1-P2 shall be a BER-TLV tag on two bytes. The values '4000' and 'FFFF' are RFU.

When a primitive data object is requested, the data field of the response message shall contain the value of the corresponding primitive data object.

When a constructed data object is requested, the data field of the response message shall contain the value of the constructed data object, i.e., data objects including their tag, length and value.

6.9.4 Response message (nominal case)

If the L_e field contains only zeroes, then within the limit of 256 for short length or 65 536 for extended length, all the required information should be returned.

Table 53 — GET DATA response APDU

Data field	L_r (may be equal to L_e) bytes
SW1-SW2	Status bytes

6.9.5 Status conditions

The following specific warning condition may occur.

— SW1 = '62' with SW2 =
• '81' : Part of returned data may be corrupted.

The following specific error conditions may occur.

— SW1 = '67' with SW2 =
• '00' : Wrong length (empty L_e field).

— SW1 = '69' with SW2 =
• '82' : Security status not satisfied.
• '85' : Conditions of use not satisfied.

— SW1 = '6A' with SW2 =
• '81' : Function not supported.
• '88' : Referenced data (data objects) not found.

— SW1 = '6C' with SW2 =
• 'XX' : Wrong length (wrong L_e field ; 'XX' indicates the exact length).

6.10 PUT DATA command

6.10.1 Definition and scope

The PUT DATA command is used for storing one primitive data object, or one or more data objects contained in a constructed data object, within the current context (e.g., application-specific environment or current DF). The exact storing functions (writing once and/or updating and/or appending) are to be induced by the definition or the nature of the data objects.

NOTE — The command could be used, for example, to update data objects.

6.10.2 Conditional usage and security

The command can be performed only if the security status satisfies the security conditions defined by the application within the context for the function(s).

6.10.3 Command message

Table 54 — PUT DATA command APDU

CLA	As defined in 5.4.1
INS	'DA'
P1-P2	See table 55
L_c field	Length of the subsequent data field
Data field	Parameters and data to be written
L_e field	Empty

Table 55 — Coding of the parameters P1-P2

Value	Meaning
'0000' to '003F'	RFU
'0040' to '00FF'	BER-TLV tag (1 byte) in P2
'0100' to '01FF'	Application data (proprietary coding)
'0200' to '02FF'	SIMPLE-TLV tag in P2
'0300' to '3FFF'	RFU
'4000' to 'FFFF'	BER-TLV tag (2 bytes) in P1-P2

Store application data

• When the value of P1-P2 lies in the range from '0100' to '01FF', the value of P1-P2 shall be an identifier reserved for card internal tests and for proprietary services meaningful within a given application context.

Store data objects

• When the value of P1-P2 lies in the range from '0040' to '00FF', the value of P2 shall be a BER-TLV tag on a single byte. The value '00FF' is reserved for indicating that the data field carries BER-TLV data objects.

• When the value of P1-P2 lies in the range from '0200' to '02FF', the value of P2 shall be a SIMPLE-TLV tag. The value '0200' is RFU. The value '02FF' is reserved for indicating that the data field carries SIMPLE-TLV data objects.

• When the value of P1-P2 lies in the range from '4000' to 'FFFF', the value of P1-P2 shall be a BER-TLV tag on two bytes. The values '4000' and 'FFFF' are RFU.

When a primitive data object is provided, the data field of the command message shall contain the value of the corresponding primitive data object.

When a constructed data object is provided, the data field of the command message shall contain the value of the constructed data object, i.e., data objects including their tag, length and value.

6.10.4 Response message (nominal case)

Table 56 — PUT DATA response APDU

Data field	Empty
SW1-SW2	Status bytes

6.10.5 Status conditions

The following specific warning conditions may occur.

— SW1 = '63' with SW2 =
• 'CX': Counter (successful storing, but after using an internal retry routine, 'X' ≠ '0' indicates the number of retries; 'X' = '0' means that no counter is provided).

The following specific error conditions may occur.

— SW1 = '65' with SW2 =
• '81': Memory failure (unsuccessful storing).

— SW1 = '67' with SW2 =
• '00': Wrong length (wrong L_c field).

— SW1 = '69' with SW2 =
• '82': Security status not satisfied.
• '85': Conditions of use not satisfied.

— SW1 = '6A' with SW2 =
• '80': Incorrect parameters in the data field.
• '81': Function not supported.
• '84': Not enough memory space in the file.
• '85': L_c inconsistent with TLV structure.

6.11 SELECT FILE command

6.11.1 Definition and scope

A successful SELECT FILE sets a current file within a logical channel (see 5.5). Subsequent commands may implicitly refer to the current file through that logical channel.

Selecting a DF (which may be the MF) sets it as current DF. After such a selection, an implicit current EF may be referred to through that logical channel.

Selecting an EF sets a pair of current files : the EF and its parent file.

After the answer to reset, the MF is implicitly selected through the basic logical channel (see 5.5.2), unless specified differently in the historical bytes (see 8) or in the initial data string (see 9).

NOTE — A direct selection by DF name can be used for selecting applications registered according to part 5 of ISO/IEC 7816.

6.11.2 Conditional usage and security

The following conditions shall apply to each open logical channel.

Unless otherwise specified, the correct execution of the command modifies the security status (see 5.2.1) according to the following rules.

— When the current EF is changed, or when there is no current EF, the security status, if any, specific to a former current EF is lost.

— When the current DF is a descendant of, or identical to the former current DF, the security status specific to the former current DF is maintained.

— When the current DF is neither a descendant of, nor identical to the former current DF, the security status specific to the former current DF is lost. The security status common to all common ancestors of the previous and new current DF is maintained.

6.11.3 Command message

Table 57 — SELECT FILE command APDU

CLA	As defined in 5.4.1
INS	'A4'
P1	Selection control, see table 58
P2	Selection options, see table 59
L_c field	Empty or length of the subsequent data field
Data field	If present, according to P1-P2, — file identifier — path from the MF — path from the current DF — DF name
L_e field	Empty or maximum length of data expected in response

Table 58 — Coding of the selection control P1

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	**0**	**0**	**0**	**0**	**0**	**x**	**x**	**Selection by file identifier**
0	0	0	0	0	0	0	0	— Select MF, DF or EF (data field = identifier or empty)
0	0	0	0	0	0	0	1	— Select child DF (data field = DF identifier)
0	0	0	0	0	0	1	0	— Select EF under current DF (data field = EF identifier)
0	0	0	0	0	0	1	1	— Select parent DF of the current DF (empty data field)
0	**0**	**0**	**0**	**0**	**1**	**x**	**x**	**Selection by DF name**
0	0	0	0	0	1	0	0	— Direct selection by DF name (data field = DF name)
0	0	0	0	0	1	0	1	RFU
0	0	0	0	0	1	1	0	RFU
0	0	0	0	0	1	1	1	RFU
0	**0**	**0**	**0**	**1**	**0**	**x**	**x**	**Selection by path (see 5.1.2)**
0	0	0	0	1	0	0	0	— Select from MF (data field = path without the identifier of the MF)
0	0	0	0	1	0	0	1	— Select from current DF (data field = path without the identifier of the current DF)
0	0	0	0	1	0	1	0	RFU
0	0	0	0	1	0	1	1	RFU
Any other value								RFU

When P1 = '00', the card knows either because of a specific coding of the file identifier or because of the context of execution of the command if the file to select is the MF, a DF or an EF.

When P1-P2 = '0000', if a file identifier is provided, then it shall be unique in the following environments:

— the immediate children of the current DF,

— the parent DF,

— the immediate children of the parent DF.

If P1-P2 = '0000' and if the data field is empty or equal to '3F00', then select the MF.

When P1 = '04', the data field is a DF name, possibly right truncated. When supported, successive such commands with the same data field shall select DFs whose names match with the data field, i.e., start with the command data field. If the card accepts the SELECT FILE command with an empty data field, then all or a subset of the DFs can be successively selected.

NOTE — See 8.3.6 for the selection methods supported by the card.

Table 59 — Coding of the selection options P2

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	-	-	0	0	— First or only occurrence
0	0	0	0	-	-	0	1	— Last occurrence
0	0	0	0	-	-	1	0	— Next occurrence
0	0	0	0	-	-	1	1	— Previous occurrence
0	**0**	**0**	**0**	**x**	**x**	**-**	**-**	**File control information option (see 5.1.5)**
0	0	0	0	0	0	-	-	— Return FCI, optional template
0	0	0	0	0	1	-	-	— Return FCP template
0	0	0	0	1	0	-	-	— Return FMD template
Any other value								RFU

6.11.4 Response message (nominal case)

If the L_e field contains only zeroes, then within the limit of 256 for short length or 65 536 for extended length, all the bytes corresponding to the selection option should be returned.

Table 60 — SELECT FILE response APDU

Data field	Information according to P2 (at most L_e bytes)
SW1-SW2	Status bytes

6.11.5 Status conditions

The following specific warning conditions may occur.

— SW1 = '62' with SW2 =
- '83': Selected file invalidated.
- '84': FCI not formatted according to 5.1.5.

The following specific error conditions may occur.

— SW1 = '6A' with SW2 =
- '81': Function not supported.
- '82': File not found.
- '86': Incorrect parameters P1-P2.
- '87': L_c inconsistent with P1-P2.

6.12 VERIFY command

6.12.1 Definition and scope

The VERIFY command initiates the comparison in the card of the verification data sent from the interface device with the reference data stored in the card (e.g., password).

6.12.2 Conditional usage and security

The security status may be modified as a result of a comparison. Unsuccessful comparisons may be recorded in the card (e.g., to limit the number of further attempts of the use of the reference data).

6.12.3 Command message

Table 61 — VERIFY command APDU

CLA	As defined in 5.4.1
INS	'20'
P1	'00' (other values are RFU)
P2	Qualifier of the reference data, see table 62
L_c field	Empty or length of the subsequent data field
Data field	Empty or verification data
L_e field	Empty

Table 62 — Coding of the reference control P2

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	0	0	0	— No information is given
0	-	-	-	-	-	-	-	— Global reference data (e.g., card password)
1	-	-	-	-	-	-	-	— Specific reference data (e.g., DF specific password)
-	x	x	-	-	-	-	-	00 (other values are RFU)
-	-	-	x	x	x	x	x	— Reference data number

NOTES

1 P2 = '00' is reserved to indicate that no particular qualifier is used, in those cards where the VERIFY command references the secret data unambiguously.

2 The reference data number may be for example a password number or a short EF identifier.

3 When the body is empty, the command may be used either to retrieve the number 'X' of further allowed retries (SW1-SW2 = '63CX') or to check whether the verification is not required (SW1-SW2 = '9000').

6.12.4 Response message (nominal case)

Table 63 — VERIFY response APDU

Data field	Empty
SW1-SW2	Status bytes

6.12.5 Status conditions

The following specific warning conditions may occur.

— SW1 = '63' with SW2 =
- '00': No information given (verification failed).

• 'CX': Counter (verification failed; 'X' indicates the number of further allowed retries).

The following specific error conditions may occur.

— SW1 = '69' with SW2 =
• '83': Authentication method blocked.
• '84': Referenced data invalidated.

— SW1 = '6A' with SW2 =
• '86': Incorrect parameters P1-P2.
• '88': Referenced data not found.

6.13 INTERNAL AUTHENTICATE command

6.13.1 Definition and scope

The INTERNAL AUTHENTICATE command initiates the computation of the authentication data by the card using the challenge data sent from the interface device and a relevant secret (e.g., a key) stored in the card.

When the relevant secret is attached to the MF, the command may be used to authenticate the card as a whole.

When the relevant secret is attached to another DF, the command may be used to authenticate that DF.

6.13.2 Conditional usage and security

The successful execution of the command may be subject to successful completion of prior commands (e.g., VERIFY, SELECT FILE) or selections (e.g., the relevant secret).

If a key and an algorithm are currently selected when issuing the command, then the command may implicitly use the key and the algorithm.

The number of times the command is issued may be recorded in the card to limit the number of further attempts of using the relevant secret or the algorithm.

6.13.3 Command message

Table 64 — INTERNAL AUTHENTICATE command APDU

CLA	As defined in 5.4.1
INS	'88'
P1	Reference of the algorithm in the card
P2	Reference of the secret, see table 65
L_c field	Length of the subsequent data field
Data field	Authentication related data (e.g., challenge)
L_e field	Maximum number of bytes expected in response

P1 = '00' indicates that no information is given. The reference of the algorithm is known either before issuing the command or is provided in the data field.

P2 = '00' indicates that no information is given. The reference of the secret is known either before issuing the command or is provided in the data field.

Table 65 — Coding of the reference control P2

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	0	0	0	— No information is given
0	-	-	-	-	-	-	-	— Global reference data (e.g., an MF specific key)
1	-	-	-	-	-	-	-	— Specific reference data (e.g., a DF specific key)
-	x	x	-	-	-	-	-	00 (other values are RFU)
-	-	-	x	x	x	x	x	— Number of the secret

NOTE — The number of the secret may be for example a key number or a short EF identifier.

6.13.4 Response message (nominal case)

Table 66 — INTERNAL AUTHENTICATE response APDU

Data field	Authentication related data (e.g., response to the challenge)
SW1-SW2	Status bytes

NOTE — The response message may include data useful for further application security functions (e.g., random number).

6.13.5 Status conditions

The following specific error conditions may occur.

— SW1 = '69' with SW2 =
• '84': Referenced data invalidated.
• '85': Conditions of use not satisfied.

— SW1 = '6A' with SW2 =
• '86': Incorrect parameters P1-P2.
• '88': Referenced data not found.

6.14 EXTERNAL AUTHENTICATE command

6.14.1 Definition and scope

The EXTERNAL AUTHENTICATE command conditionally updates the security status using the result (yes or no) of the computation by the card based on a challenge previously issued by the card (e.g., by a GET CHALLENGE command), a key possibly secret stored in the card and authentication data transmitted by the interface device.

6.14.2 Conditional usage and security

The successful execution of the command requires that the last challenge obtained from the card is valid.

Unsuccessful comparisons may be recorded in the card (e.g., to limit the number of further attempts of the use of the reference data).

6.14.3 Command message

Table 67 — EXTERNAL AUTHENTICATE command APDU

CLA	As defined in 5.4.1
INS	'82'
P1	Reference of the algorithm in the card
P2	Reference of the secret, see table 68
L_c field	Empty or length of the subsequent data field
Data field	Empty or authentication related data (e.g., response to the challenge)
L_e field	Empty

P1 = '00' indicates that no information is given. The reference of the algorithm is known either before issuing the command or is provided in the data field.

P2 = '00' indicates that no information is given. The reference of the secret is known either before issuing the command or is provided in the data field.

Table 68 — Coding of the reference control P2

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	0	0	0	— No information is given
0	-	-	-	-	-	-	-	— Global reference data (e.g., an MF specific key)
1	-	-	-	-	-	-	-	— Specific reference data (e.g., a DF specific key)
-	x	x	-	-	-	-	-	00 (other values are RFU)
-	-	-	x	x	x	x	x	— Number of the secret

NOTES

1 The number of the secret may be for example a key number or a short EF identifier.

2 When the body is empty, the command may be used either to retrieve the number 'X' of further allowed retries (SW1-SW2 = '63CX') or to check whether the verification is not required (SW1-SW2 = '9000').

6.14.4 Response message (nominal case)

Table 69 — EXTERNAL AUTHENTICATE response APDU

Data field	Empty
SW1-SW2	Status bytes

6.14.5 Status conditions

The following specific warning conditions may occur.

— SW1 = '63' with SW2 =
- '00' : No information given (authentication failed).
- 'CX' : Counter (authentication failed; 'X' indicates the number of further allowed retries).

The following specific error conditions may occur.

— SW1 = '67' with SW2 =
- '00' : Wrong length (the L_c field is incorrect).

— SW1 = '69' with SW2 =
- '83' : Authentication method blocked.
- '84' : Referenced data invalidated.
- '85' : Conditions of use not satisfied (the command is not allowed in the context).

— SW1 = '6A' with SW2 =
- '86' : Incorrect parameters P1-P2.
- '88' : Referenced data not found.

6.15 GET CHALLENGE command

6.15.1 Definition and scope

The GET CHALLENGE command requires the issuing of a challenge (e.g., random number) for use in a security related procedure (e.g., EXTERNAL AUTHENTICATE command).

6.15.2 Conditional usage and security

The challenge is valid at least for the next command. No further condition is specified in this part of ISO/IEC 7816.

6.15.3 Command message

Table 70 — GET CHALLENGE command APDU

CLA	As defined in 5.4.1
INS	'84'
P1-P2	'0000' (other values are RFU)
L_c field	Empty
Data field	Empty
L_e field	Maximum length of the expected response

6.15.4 Response message (nominal case)

Table 71 — GET CHALLENGE response APDU

Data field	Challenge
SW1-SW2	Status bytes

6.15.5 Status conditions

The following specific error conditions may occur.

— SW1 = '6A' with SW2 =
- '81' : Function not supported.
- '86' : Incorrect parameters P1-P2.

6.16 MANAGE CHANNEL command

6.16.1 Definition and scope

The MANAGE CHANNEL command opens and closes logical channels.

The open function opens a new logical channel other than the basic one. Options are provided for the card to assign a logical channel number, or for the logical channel number to be supplied to the card.

The close function explicitly closes a logical channel other than the basic one. After the successful closing, the logical channel shall be available for re-use.

6.16.2 Conditional usage and security

When the open function is performed from the basic logical channel, then after a successful open, the MF shall be implicitly selected as the current DF and the security status for the new logical channel should be the same as for the basic logical channel after ATR. The security status of the new logical channel should be separate from that of any other logical channel.

When the open function is performed from a logical channel which is not the basic one, then after a successful open, the current DF of the logical channel from which the command was issued shall be selected as the current DF and the security status for the new logical channel should be the same as for the logical channel from which the open function was performed.

After a successful close function, the security status related to this logical channel is lost.

6.16.3 Command message

Table 72 — MANAGE CHANNEL command APDU

CLA	As defined in 5.4.1
INS	'70'
P1	P1 = '00' to open a logical channel P1 = '80' to close a logical channel (other values are RFU)
P2	'00', '01', '02' or '03' (other values are RFU)
L_c field	Empty
Data field	Empty
L_e field	'01' if P1-P2 = '0000' Empty if P1-P2 ≠ '0000'

Bit b8 of P1 is used to indicate the open function or the close function; if b8 is 0 then MANAGE CHANNEL shall open a logical channel and if b8 is 1 then MANAGE CHANNEL shall close a logical channel.

For the open function (P1 = '00'), the bits b1 and b2 of P2 are used to code the logical channel number in the same manner as in the class byte (see 5.4.1); the other bits of P2 are RFU.

— When b1 and b2 of P2 are null, then the card will assign a logical channel number that will be returned in bits b1 and b2 of the data field.

— When b1 and/or b2 of P2 are not null, they code a logical channel number other than the basic one; then the card will open the externally assigned logical channel number.

6.16.4 Response message (nominal case)

Table 73 — MANAGE CHANNEL response APDU

Data field	Logical channel number if P1-P2 = '0000' Empty if P1-P2 ≠ '0000'
SW1-SW2	Status bytes

6.16.5 Status conditions

The following specific warning conditions may occur.

— SW1 = '62' with SW2 =
• '00' : No information is given.

7 Transmission-oriented interindustry commands

It shall not be mandatory for all cards complying to this part of ISO/IEC 7816 to support all the described commands or all the options of a supported command.

When international interchange is required, a set of card system services and related commands and options shall be used as defined in clause 9.

Table 11 provides a summary of the commands defined in this part of ISO/IEC 7816.

The impact of secure messaging (see 5.6) on the message structure is not described in this clause.

The list of error and warning conditions given in each clause 7.X.5 is not exhaustive (see 5.4.5).

7.1 GET RESPONSE command

7.1.1 Definition and scope

The GET RESPONSE command is used to transmit from the card to the interface device APDU(s) (or part of APDUs) which otherwise could not be transmitted by the available protocols.

7.1.2 Conditional usage and security

No condition.

7.1.3 Command message

Table 74 — GET RESPONSE command APDU

CLA	As defined in 5.4.1
INS	'C0'
P1-P2	'0000' (other values are RFU)
L_c field	Empty
Data field	Empty
L_e field	Maximum length of data expected in response

7.1.4 Response message (nominal case)

If the L_e field contains only zeroes, then within the limit of 256 for short length or 65 536 for extended length, all the available bytes should be returned.

Table 75 — GET RESPONSE response APDU

Data field	(Part of) APDU according to L_e
SW1-SW2	Status bytes

7.1.5 Status conditions

The following specific normal processing may occur.

— SW1 = '61' with SW2 =
• 'XX' : Normal processing: more data bytes are available ('XX' indicates a number of extra data bytes still available by a subsequent GET RESPONSE).

The following specific warning condition may occur.

— SW1 = '62' with SW2 =
• '81' : Part of returned data may be corrupted.

The following specific error conditions may occur.

— SW1 = '67' with SW2 =
• '00' : Wrong length (incorrect L_c field).

— SW1 = '6A' with SW2 =
• '86' : Incorrect parameters P1-P2.

— SW1 = '6C' with SW2 =
• 'XX' : Wrong length (wrong L_e field ; 'XX' indicates the exact length).

7.2 ENVELOPE command

7.2.1 Definition and scope

The ENVELOPE command is used to transmit APDU(s), or part of APDUs, or any data string, which otherwise could not be transmitted by the available protocols.

NOTE — The usage of ENVELOPE for SM is shown in annex F.

7.2.2 Conditional usage and security

No condition.

7.2.3 Command message

Table 76 — ENVELOPE command APDU

CLA	As defined in 5.4.1
INS	'C2'
P1-P2	'0000' (other values are RFU)
L_c field	Length of the subsequent data field
Data field	(Part of) APDU
L_e field	Empty or length of expected data

When the ENVELOPE command is used under T=0 for transmitting data strings, an empty data field in an ENVELOPE command APDU means "end of data string".

7.2.4 Response message (nominal case)

Table 77 — ENVELOPE response APDU

Data field	Empty or (part of) APDU according to L_e
SW1-SW2	Status bytes

NOTE — The status bytes belong to the ENVELOPE command. Status bytes of a command transmitted in the data field of the ENVELOPE command may be found in the data field of the ENVELOPE response.

7.2.5 Status conditions

The following specific error condition may occur.

— SW1 = '67' with SW2 =
• '00' : Wrong length (incorrect L_c field).

8 Historical bytes

8.1 Purpose and general structure

The historical bytes tell the outside world how to use the card when the transport protocol is ascertained according to part 3 of ISO/IEC 7816.

The number of historical bytes (at most 15 bytes) is specified and coded as defined in part 3 of ISO/IEC 7816.

The information carried by the historical bytes may also be found in an ATR file (default EF identifier = '2F01').

If present, the historical bytes are made up of three fields:

— a mandatory category indicator (1 byte),

— optional COMPACT-TLV data objects,

— a conditional status indicator (3 bytes).

8.2 Category indicator (mandatory)

The category indicator is the first historical byte. If the category indicator is equal to '00', '10' or '8X', then the format of the historical bytes shall be according to this part of ISO/IEC 7816.

Table 78 — Coding of the category indicator

Value	Meaning
'00'	Status information shall be present at the end of the historical bytes (not in TLV).
'10'	Specified in 8.5
'80'	Status information, if present, is contained in an optional COMPACT-TLV data object.
'81' to '8F'	RFU
Other values	Proprietary

8.3 Optional COMPACT-TLV data objects

The coding of the COMPACT-TLV data objects is deduced from the basic encoding rules of ASN.1 (see ISO/IEC 8825 and annex D) for BER-TLV data objects with tag = '4X' and length = '0Y'. The coding of such data objects is replaced by 'XY' followed by 'Y' bytes of data. In this clause, 'X' is referred to as the tag number and 'Y' as the length.

Besides the data objects defined in this clause, the historical bytes may contain data objects defined in part 5 of ISO/IEC 7816. In this case, the coding of the tags and length fields defined in part 5 shall be modified as above.

When COMPACT-TLV data objects defined in this clause appear in the ATR file, they shall be encoded according to the basic encoding rules of ASN.1 (i.e., tag = '4X', length = '0Y').

All application-class tags not defined in ISO/IEC 7816 are reserved for ISO.

8.3.1 Country/issuer indicator

When present, this data object denotes a country or an issuer.

This data object is introduced by either '1Y' or '2Y'.

Table 79 — Coding of the country/issuer indicator

Tag	Length	Value
'1'	variable	Country code and national data
'2'	variable	Issuer identification number

The tag '1' is followed by the appropriate length (1 nibble) and by three digits denoting the country as defined in ISO 3166. Data which follows (odd number of nibbles) is chosen by the relevant national standardization body.

The tag '2' is followed by the appropriate length (1 nibble) and by the issuer identification number as defined in part 1 of ISO/IEC 7812. If the issuer identification number contains an odd number of digits, then it shall be right padded with a nibble valued to 'F'.

8.3.2 Card service data

This data object denotes the methods available in the card for supporting the services described in clause 9.

This data object is introduced by '31'.

When this data object is not present, the card supports only the implicit application selection.

Table 80 — Card-profile for application-independent card services

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	— Direct application selection by full DF name
-	1	-	-	-	-	-	-	— Selection by partial DF name (see 9.3.2)
								Data objects available
-	-	1	-	-	-	-	-	— in DIR file
-	-	-	1	-	-	-	-	— in ATR file
								File I/O services by
-	-	-	-	1	-	-	-	— READ BINARY command
-	-	-	-	0	-	-	-	— READ RECORD(S) command
-	-	-	-	-	x	x	x	000 (other values are RFU)

NOTE — The contents of the DIR and ATR files may give information on selection methods.

8.3.3 Initial access data

This optional data object allows the retrieval of a string of data objects defined in ISO/IEC 7816. The string retrieved by this data object is called the "initial data string".

This data object is introduced by '41', '42' or '45'.

Any command APDU described in this clause is assumed to be the first command sent after the answer to reset. Consequently, the data available at this point may not be subsequently retrievable.

8.3.3.1 Length = '1'

When only one byte of information is provided, it indicates the length of the command to perform for retrieving the initial data string. The command to perform is a READ BINARY command structured as follows.

Table 81 — Coding of the command when length = '1'

CLA	'00' (see 5.4.1)
INS	'B0'
P1-P2	'0000'
L_c field	Empty
Data field	Empty
L_e field	First and only byte of value field of initial access data (indicating the number of bytes to be read)

8.3.3.2 Length = '2'

When two bytes of information are provided, the first byte indicates the file structure (transparent or record) and the short identifier of the EF to be read. The second byte indicates the length of the READ command to perform for retrieving the initial data string.

Table 82 — Structure of the first byte

b8	= 0 Record oriented file = 1 Transparent file
b7-b6	00 (other values are RFU)
b5-b1	Short EF identifier

- When b8=0, the command to perform is a READ RECORD(S) command structured as follows.

Table 83 — Coding of the command when b8=0

CLA	'00' (see 5.4.1)
INS	'B2'
P1	'01'
P2	Short EF identifier (from the first byte of initial access data) followed by b3-b2-b1 = 110
L_c field	Empty
Data field	Empty
L_e field	Second and last byte of value field of initial access data (indicating the number of bytes to be read)

- When b8=1, the command to perform is a READ BINARY command structured as follows.

Table 84 — Coding of the command when b8=1

CLA	'00' (see 5.4.1)
INS	'B0'
P1	Value of the first byte of initial access data
P2	'00'
L_c field	Empty
Data field	Empty
L_e field	Second and last byte of value field of initial access data (indicating the number of bytes to be read)

8.3.3.3 Length = '5'

The value found in the initial access data object consists of the APDU of a command to perform. When executed, this command provides the initial data string in its response data field.

8.3.4 Card issuer's data

This data object is optional and of variable length. Structure and coding are defined by the card issuer.

This data object is introduced by '5Y'.

8.3.5 Pre-issuing data

This data object is optional and of variable length. Structure and coding are not defined in this part of ISO/IEC 7816. It may be used for indicating

— card manufacturer,

— integrated circuit type,

— integrated circuit manufacturer,

— ROM mask version,

— operating system version.

This data object is introduced by '6Y'.

8.3.6 Card capabilities

This data object is optional and of variable length. Its value field consists of either the first software function table, or the first two software function tables, or the three software function tables.

This data object is introduced by '71', '72' or '73'.

Table 85 shows the first software function table.

Table 85 — First software function table

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
								DF selection
1	-	-	-	-	-	-	-	— by full DF name
-	1	-	-	-	-	-	-	— by partial DF name
-	-	1	-	-	-	-	-	— by path
-	-	-	1	-	-	-	-	— by file identifier
-	-	-	-	1	-	-	-	— implicit
								EF management
-	-	-	-	-	1	-	-	— Short EF identifier supported
-	-	-	-	-	-	1	-	— Record number supported
-	-	-	-	-	-	-	1	— Record identifier supported

Table 86 shows the second software function table which is the data coding byte. The data coding byte may also be present as the second data element in the file control parameter with tag '82' (see table 2 in 5.1.5).

Table 86 — Second software function table (data coding byte)

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	x	x	-	-	-	-	-	**Behavior of write functions**
-	0	0	-	-	-	-	-	— one-time write
-	0	1	-	-	-	-	-	— proprietary
-	1	0	-	-	-	-	-	— write OR
-	1	1	-	-	-	-	-	— write AND
-	-	-	-	-	x	x	x	**Data unit size in nibbles** (power of 2, e.g., 001 = 2 nibbles) (default value = one byte)
x	-	-	x	x	-	-	-	0...00... (other values are RFU)

Table 87 shows the third software function table.

Table 87 — Third software function table

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	-	-	-	-	-	-	-	0 (1 is RFU)
-	1	-	-	-	-	-	-	— Extended L_c and L_e fields
-	-	x	-	-	-	-	-	0 (1 is RFU)
-	-	-	x	x	-	-	-	**Logical channel assignment**
			1					— by the card
				1				— by the interface device
-	-	-	0	0	-	-	-	No logical channel
-	-	-	-	-	x	-	-	0 (1 is RFU)
-	-	-	-	-	-	x	y	— Maximum number of logical channels (= 2x+y+1)

8.4 Status information

The status information consists of 3 bytes: the card life status (1 byte) and the two status bytes SW1-SW2.

The value '00' of the card life status indicates that no card life status is provided. The values '80' to 'FE' are proprietary. All other values are RFU.

The value '9000' of SW1-SW2 indicates normal processing as defined in 5.4.5.

The value '0000' of SW1-SW2 indicates that the status is not indicated.

If the category indicator is valued to '80', then the status information may be present in a COMPACT-TLV data object. In this case, the tag number is '8'. When the length is '1', then the value is the card life status. When the length is '2', then the value is SW1-SW2. When the length is '3', then the value is the card life status followed by SW1-SW2. Other values of the length are reserved for ISO.

8.5 DIR data reference

If the category indicator is '10', then the following byte is the DIR data reference. The coding and meaning of this byte are outside the scope of this part of ISO/IEC 7816.

9 Application-independent card services

9.1 Definitions and scope

This clause describes the application-independent card services, referred to as "card services" in the following text. Their purpose is to provide interchange mechanisms between a card and an interface device knowing nothing about each other except that they both comply with this part of ISO/IEC 7816.

Card services are supported by any combination of

— historical bytes,

— contents of one or more reserved EFs,

— sequences of interindustry commands.

The commands use CLA = '00' (see 5.4.1), i.e., no secure messaging and the basic logical channel.

There is no need for an application to comply with this clause once it has been identified and selected in the card. It is possible for an application to use other mechanisms compatible with this part of ISO/IEC 7816 for achieving similar functions. Therefore such solutions may not guarantee interchange.

The following card services are defined.

— Card identification service — This service allows the interface device to identify the card as well as how to deal with it.

— Application selection service — This service allows the interface device to know what application is active in the card (if any) as well as how to select and start an application in the card.

— Data object retrieval service — This service allows to retrieve data objects defined either in this part or in other parts of ISO/IEC 7816. This clause describes standard mechanisms only for interindustry data objects.

— File selection service — This service allows selection of un-named DFs, and EFs.

— File I/O service — This service allows access to data stored in EFs.

9.2 Card identification service

This function consists of the card providing information to the outside world on its logical content as well as some general data objects all applications might be interested in (e.g., interindustry data objects). The information, called "card identification data", is given by the card in the historical bytes and possibly in a file implicitly selected immediately after the answer to reset.

Access to this file is indicated in the initial access data information (see 8.3.3).

If the initial access data of the historical bytes does not denote a READ command, then the response to the command to perform contains card identification data.

9.3 Application selection service

An application is either implicitly selected in a card or can be explicitly selected by its name.

9.3.1 Implicit application selection

When an application is implicitly selected in a card, the application identifier as defined in part 5 of ISO/IEC 7816 should be indicated in the card identification data. If not present in the card identification data, then it shall be present in the ATR file.

9.3.2 Direct application selection

A card in a multi-application environment shall be able to respond positively to a direct application selection performed by a SELECT FILE command specifying the application identifier as DF name.

The application identifier should be provided completely in the command APDU. In case of an application selection by partial DF name, the next application matching with the name proposed may be selected and the full DF name will be made available in the response message of the SELECT FILE command as the file control parameter with tag '84' (see table 2 in 5.1.5).

The APDU of the command to perform is the following.

Table 88 — Coding of the command for direct application selection

CLA	'00' (see 5.4.1)
INS	'A4'
P1-P2	'0400'
L_c field	Length in bytes the data field
Data field	Full or partial DF name
L_e field	Present, contains only zeroes

9.4 Data object retrieval service

Data objects used for application-independent international interchange are defined in this part and other parts of ISO/IEC 7816.

The retrieval of those data objects relies on one or both of the following methods:

— presence of a data object in the card identification data (see 9.2),

— presence of a data object in the DIR file (path = '3F002F00') or in the ATR file (path = '3F002F01').

The information necessary to retrieve data objects by an indirect method are defined in part 6 of ISO/IEC 7816.

9.5 File selection service

When the path to an EF is known, the number of SELECT FILE commands to be issued equals the length of the path divided by two, minus one (the path always starts with the current DF).

If the path length is more than four bytes, then until all available DF Identifiers of the path have been used, one or more SELECT FILE commands shall be performed with the following command APDU.

Table 89 — Coding of the command to select a DF using a file identifier

CLA	'00' (see 5.4.1)
INS	'A4'
P1-P2	'0100'
L_c field	'02'
Data field	DF identifier (from bytes 3 and 4 of the path)
L_e field	Empty

The last and possibly only selection is an EF selection with the following command APDU.

Table 90 — Coding of the command to select an EF

CLA	'00' (see 5.4.1)
INS	'A4'
P1-P2	'0200'
L_c field	'02'
Data field	EF identifier (last two bytes of the path)
L_e field	Empty

9.6 File I/O service

Once a file used for interindustry interchange has been selected, the contents relevant to interchange shall be returned by one of the following command APDUs.

- If the first software function table is absent, or does not denote the support of record-oriented commands, then the following command shall be performed.

Table 91 — Coding of the command to read a transparent file

CLA	'00' (see 5.4.1)
INS	'B0'
P1-P2	'0000'
L_c field	Empty
Data field	Empty
L_e field	Present, contains only zeroes

- If the first software function table denotes the support of record-oriented commands, then the following command shall be performed.

Table 92 — Coding of a command to read a record-oriented file

CLA	'00' (see 5.4.1)
INS	'B2'
P1-P2	'0005'
L_c field	Empty
Data field	Empty
L_e field	Present, contains only zeroes

Annex A

(normative)

Transportation of APDU messages by T=0

A.1 Case 1

The command APDU is mapped onto the T=0 command TPDU by assigning the value '00' to P3.

Command APDU

CLA	INS	P1	P2

Command TPDU

CLA	INS	P1	P2	P3 = '00'

The response TPDU is mapped onto the response APDU without any change.

Response TPDU

SW1	SW2

Response APDU

SW1	SW2

A.2 Case 2 Short

In this case, L_e is valued from 1 to 256 and coded on byte B_1 (B_1 = '00' means maximum, i.e., L_e = 256).

The command APDU is mapped onto the T=0 command TPDU without any change.

C-APDU

CLA	INS	P1	P2	$L_e = B_1$

C-TPDU

CLA	INS	P1	P2	$P3 = B_1$

The response TPDU is mapped onto the response APDU according to the acceptance of L_e and according to the processing of the command.

Case 2S.1 — L_e accepted

The response TPDU is mapped onto the response APDU without any change.

R-TPDU

L_e bytes	SW1	SW2

R-APDU

L_e bytes	SW1	SW2

Case 2S.2 — L_e definitely not accepted

L_e is not accepted by the card which does not support the service of providing data if the length is wrong.

The response TPDU from the card indicates that the card aborts the command because of wrong length : (SW1) = '67'. The response TPDU is mapped onto the response APDU without any change.

R-TPDU

SW1 = '67'	SW2

R-APDU

SW1 = '67'	SW2

Case 2S.3 — L_e not accepted, L_a indicated

L_e is not accepted by the card and the card indicates the available length L_a.

The response TPDU from the card indicates that the command is aborted due to a wrong length and that the right length is L_a : (SW1) = '6C' and SW2 codes L_a.

If the transmission system does not support the service of re-issuing the same command, it shall map the response TPDU onto the response APDU without any change.

R-TPDU

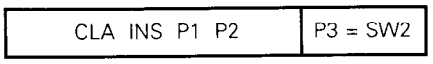
SW1 = '6C'	SW2 = L_a

R-APDU

SW1 = '6C'	SW2 = L_a

If the transmission system supports the service of re-issuing the same command, it shall re-issue the same command TPDU assigning the value L_a to parameter P3.

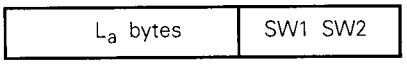
C-TPDU



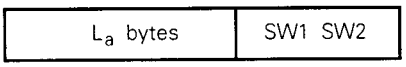
The response TPDU consists of L_a bytes followed by two status bytes.

If L_a is smaller than or equal to L_e, then the response TPDU is mapped onto the response APDU without any change.

R-TPDU



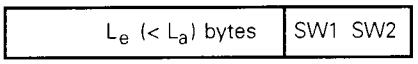
R-APDU



If L_a is greater than L_e, then the response TPDU is mapped onto the response APDU by keeping only the first L_e bytes of the body and the status bytes SW1-SW2.

R-TPDU

R-APDU



Case 2S.4 — SW1-SW2 = '9XYZ', except '9000'

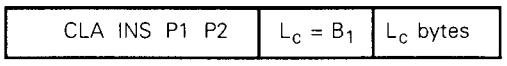
The response TPDU is mapped onto the response APDU without any change.

A.3 Case 3 Short

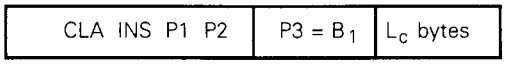
In this case, L_c is valued from 1 to 255 and coded on byte B_1 (≠ '00').

The command APDU is mapped onto the T=0 command TPDU without any change.

C-APDU

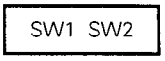


C-TPDU

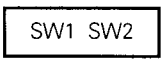


The response TPDU is mapped onto the response APDU without any change.

R-TPDU



R-APDU

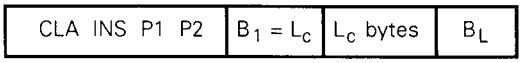


A.4 Case 4 Short

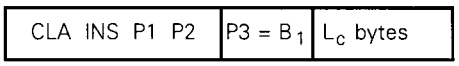
In this case, L_c is valued from 1 to 255 and coded on byte B_1, L_e is valued from 1 to 256 and coded on byte B_L (B_L = '00' means maximum i.e., L_e = 256).

The command APDU is mapped onto the T=0 command TPDU by cutting off the last byte of the body.

C-APDU



C-TPDU

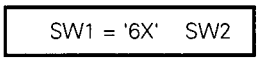


Case 4S.1 — Command not accepted

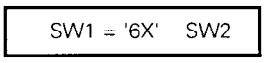
The first response TPDU from the card indicates that the card aborted the command: SW1 = '6X', except '61'.

The response TPDU is mapped onto the response APDU without any change.

R-TPDU



R-APDU



Case 4S.2 — Command accepted

The first response TPDU from the card indicates that the card performed the command: SW1-SW2 = '9000'.

The transmission system shall issue a GET RESPONSE command TPDU to the card by assigning the value L_e to parameter P3.

C-TPDU

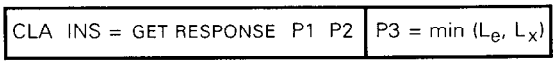
Depending on the second response TPDU from the card, the transmission system shall react as described in cases 2S.1, 2S.2, 2S.3 and 2S.4 above.

Case 4S.3 — Command accepted with information added

The first response TPDU from the card indicates that the card performed the command and gives information on the length of data bytes available: SW1 = '61' and SW2 codes L_x.

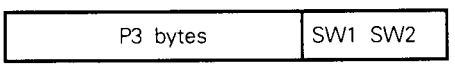
The transmission system shall issue a GET RESPONSE command TPDU to the card by assigning the minimum of L_x and L_e to parameter P3.

TPDU

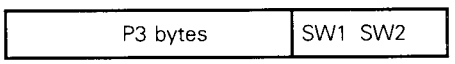


The second response TPDU is mapped onto the response APDU without any change.

R-TPDU



R-APDU



Case 4S.4 — SW1-SW2 = '9XYZ', except '9000'

The response TPDU is mapped onto the response APDU without any change.

A.5 Case 2 Extended

In this case, L_e is valued from 1 to 65 536 and coded on 3 bytes : (B_1) = '00', $(B_2$ II $B_3)$ = any value (B_2 and B_3 valued to '0000' means maximum, i.e., L_e = 65 536).

C-APDU

CLA INS P1 P2	B_1 = '00' $B_2 B_3 = L_e$

Case 2E.1 — $L_e \leq 256$, B_1 = '00, B_2 B_3 from '0001' to '0100'.

The command APDU shall be mapped onto the command TPDU by assigning the value of B_3 to parameter P3. The processing by the transmission system shall be according to case 2S.

C-TPDU

CLA INS P1 P2	P3 = B_3

Case 2E.2 — $L_e > 256$, B_1 = '00, B_2 B_3 = either '0000' or from '0101' to 'FFFF'

The command APDU shall be mapped onto the command TPDU by assigning the value of '00' to parameter P3.

C-TPDU

CLA INS P1 P2	P3 = '00'

a) If the first response TPDU from the card indicates that the card aborted the command because of wrong length (SW1 = '67'), then the response TPDU shall be mapped onto the response APDU without any change.

R-TPDU

SW1 = '67' SW2

R-APDU

SW1 = '67' SW2

b) If the first response TPDU from the card indicates that the command is aborted due to a wrong length and that the right length is L_a (SW1 = '6C' and SW2 = L_a), then the transmission system shall complete the processing as described in case 2S.3.

c) If the first response TPDU is 256 bytes of data followed by SW1-SW2 = '9000', this means that the card has no more than 256 bytes of data, and/or does not support the GET RESPONSE command. The transmission system shall then map the response TPDU onto the response APDU without any change.

R-TPDU

256 bytes	SW1 = '90' SW2 = '00'

R-APDU

256 bytes	SW1 = '90' SW2 = '00'

d) If the first or subsequent response TPDU from the card is SW1 = '61', then SW2 codes L_x which is the extra amount of bytes available from the card (SW2 valued to '00' indicates 256 extra bytes or more), the transmission system shall compute $L_m = L_e$ – (sum of the lengths of the bodies of the previously received response TPDU(s)) to obtain the amount of remaining bytes to be retrieved from the card.

If L_m = 0, then the transmission system shall concatenate the bodies of all received response TPDUs together with the trailer of the last received response TPDU into the response APDU.

If $L_m > 0$, then the transmission system shall issue a GET RESPONSE command TPDU by assigning the minimum of L_x and L_m to parameter P3. The corresponding response TPDU from the card shall be processed

— according to case d), if SW1 = '61'.

— as above when L_m = 0, if SW1 = '90'.

A.6 Case 3 Extended

In this case, L_c is valued from 1 to 65 535 and coded on 3 bytes : (B_1) = '00', $(B_2$ II $B_3) \neq$ '00 00'.

C-APDU

CLA INS P1 P2	B_1 = '00' $B_2 B_3 = L_c$	L_c bytes

Case 3E.1 — $0 < L_c < 256$, B_1 = '00', B_2 = '00', $B_3 \neq$ '00'

The command APDU is mapped onto the command TPDU by assigning the value of B_3 to parameter P3.

C-TPDU

CLA INS P1 P2	P3 = B_3	L_c bytes

In this case, L_c is valued from 1 to 255 and coded on 1 byte.

The response TPDU is mapped onto the response APDU without any change.

R-TPDU

SW1 SW2

R-APDU

SW1 SW2

Case 3E.2 — $L_c > 255$, B_1 = '00', $B_2 \neq$ '00', B_3 = any value

If the transmission system does not support the ENVELOPE command, it shall return an error response APDU meaning that the length is wrong : SW1 = '67'.

R-TPDU

SW1 = '67' SW2

R-APDU

SW1 = '67' SW2

If the transmission system supports the ENVELOPE command, it shall split the APDU into segments of length less than 256, and send those successive segments into the bodies of consecutive ENVELOPE command TPDUs.

C-TPDU

CLA INS = ENVELOPE P1 P2	P3	P3 bytes

If the first response TPDU from the card indicates that the card does not support the ENVELOPE command (SW1 = '6D'), the TPDU shall be mapped onto the response TPDU without any change.

R-TPDU

SW1 = '6D'	SW2

R-APDU

SW1 = '6D'	SW2

If the first response TPDU from the card indicates that the card does support the ENVELOPE command (SW1-SW2 = '9000'), the transmission system shall send further ENVELOPE commands as needed.

R-TPDU

SW1-SW2 = '9000'

C-TPDU

CLA INS = ENVELOPE P1 P2	P3	P3 bytes

The response TPDU corresponding to the last ENVELOPE command is mapped onto the response APDU without any change.

R-TPDU

SW1	SW2

R-APDU

SW1	SW2

A.7 Case 4 Extended

In this case, L_c is valued from 1 to 65 535 and coded on 3 bytes: (B_1) = '00', $(B_2 \| B_3) \neq$ '0000', and L_e is valued from 1 to 65 536 and coded on 2 bytes: $(B_{L-1} \| B_L)$ = any value (B_{L-1} and B_L valued to '0000' means maximum, i.e., L_e = 65 536).

C-APDU

CLA INS P1 P2	B_1 = '00' $B_2B_3 = L_c$	L_c bytes	$B_{L-1}B_L = L_e$

Case 4E.1 — $L_c < 256$, B_1 = '00', B_2 = '00', $B_3 \neq$ '00'

The command APDU is mapped onto the command TPDU by cutting off the last two bytes B_{L-1} and B_L, and by assigning the value of B_3 to parameter P3.

C-TPDU

CLA INS P1 P2	P3 = B_3	L_c bytes

In this case, L_c is valued from 1 to 255 bytes and coded on 1 byte.

a) If SW1 = '6X' in the first response TPDU from the card, then the response TPDU is mapped onto the response APDU without any change.

R-TPDU

SW1 = '6X'	SW2

R-APDU

SW1 = '6X'	SW2

b) If SW1 = '90' in the first response TPDU from the card, then

If $L_e < 257$ (B_{L-1} B_L valued from '0001' to '0100'), then the transmission system shall issue a GET RESPONSE command TPDU by assigning the value of B_L to parameter P3. The subsequent processing by the transmission system shall be according to cases 2S.1, 2S.2, 2S.3 and 2S.4 above.

If $L_e > 256$ (B_{L-1} B_L valued to '0000' or more than '0100'), then the transmission system shall issue a GET RESPONSE command TPDU by assigning the value '00' to parameter P3. The subsequent processing by the transmission system shall be according to case 2E.2 above.

c) If SW1 = '61' in the first response TPDU from the card, then the transmission system shall proceed as specified in case 2E.2 d) above.

Case 4E.2 — $L_c > 255$, B_1 = '00, $B_2 \neq$ '00', B_3 = any value

The transmission system shall go on according to case 3E.2 described above, until the command APDU has been sent completely to the card. It shall then go on as described in case 4E.1 a), b) and c) described above.

Annex B

(normative)

Transportation of APDU messages by T=1

B.1 Case 1

The command APDU is mapped onto the information field of an I-block without any change.

Command APDU

CLA	INS	P1	P2

Information field

CLA	INS	P1	P2

The information field of the I-block received in response is mapped onto the response APDU without any change.

Information field

SW1	SW2

Response APDU

SW1	SW2

B.2 Case 2 (short and extended)

The command APDU is mapped onto the information field of an I-block without any change.

C-APDU

CLA	INS	P1	P2	L_e field

Information field

CLA	INS	P1	P2	L_e field

The response APDU consists of
— either the information field of the I-block received in response,
— or the concatenation of the information fields of successive I-blocks received in response. These blocks shall be chained.

Either information field

Data field	SW1-SW2

or concatenation of information fields

Data	...	...
...	...	...
...	field	SW1-SW2

R-APDU

Data field	SW1-SW2

B.3 Case 3 (short and extended)

The command APDU is mapped without any change onto
— either the information field of one I-block,
— or the concatenation of the information fields of successive I-blocks which shall be chained.

C-APDU

CLA	INS	P1	P2	L_c field	Data field

Either information field

CLA	INS	P1	P2	L_c field	Data field

or concatenation of information fields

CLA	INS	P1	P2	L_c field	Data ...
...	...	...			
...	...	field			

The information field of the I-block received in response is mapped onto the response APDU without any change.

Information field

SW1	SW2

R-APDU

SW1	SW2

B.4 Case 4 (short and extended)

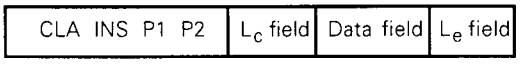
The command APDU is mapped without any change onto

— either the information field of one I-block,

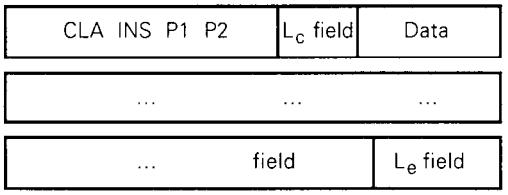
— or the concatenation of the information fields of successive I-blocks which shall be chained.

C-APDU

Either information field



or concatenation of information fields

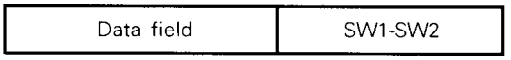


The response APDU consists of

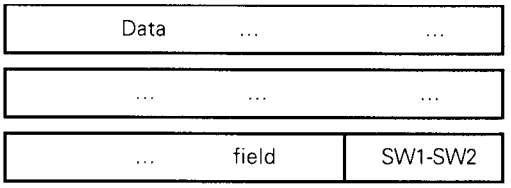
— either the information field of the I-block received in response,

— or the concatenation of the information fields of successive I-blocks received in response. These blocks shall be chained.

Either information field



or concatenation of information fields



R-APDU

Annex C

(informative)

Record pointer management

C.1 Case 1

Case 1 deals with the first command issued after a select function (either explicit or implicit). The current record pointer (CP) is undefined.

Command READ RECORDS	Record in response	Position of CP after command
Next (id=aa)	First with id=aa	Record read
	If not found, then error	Undefined
Previous (id=bb)	Last with id=bb	Record read
	If not found, then error	Undefined
First (id=cc)	First with id=cc	Record read
	If not found, then error	Undefined
Last (id=dd)	Last with id=dd	Record read
	If not found, then error	Undefined
Next (id=00)	First	Record read
Previous (id=00)	Last	Record read
First (id=00)	First	Record read
Last (id=00)	Last	Record read
Record # 00	Error	Undefined
Record # ee	# ee	Undefined
	If not found, then error	Undefined
P1='00', P2=xxxx x101	Error	Undefined
P1='00', P2=xxxx x110	Error	Undefined
# jj, P2=xxxx x101	# jj to last	Undefined
	If # jj not found, then error	Undefined
# kk, P2=xxxx x110	Last to # kk ...	Undefined
	If # kk not found, then error	Undefined

C.2 Case 2

Case 2 deals with a subsequent command. The current record pointer (CP) is defined.

Command READ RECORDS	Record in response	Position of CP after command
Next (id=aa)	Next with id=aa	Record read
	If no next, then error	Unchanged
Previous (id=bb)	Previous with id=bb	Record read
	If no previous, then error	Unchanged
First (id=cc)	First with id=cc	Record read
	If not found, then error	Unchanged
Last (id=dd)	Last with id=dd	Record read
	If not found, then error	Unchanged
Next (id=00)	CP+1	Previous CP+1
	If CP = last, then error	Unchanged
Previous (id=00)	CP–1	Previous CP–1
	If CP = first, then error	Unchanged
First (id=00)	First	First record
Last (id=00)	Last	Last record
Record # 00	CP	Unchanged
Record # ee	# ee	Unchanged
	If not found, then error	Unchanged
P1='00', P2=xxxx x101	CP to last	Unchanged
P1='00', P2=xxxx x110	Last to CP	Unchanged
# jj, P2=xxxx x101	# jj to last	Unchanged
	If # jj not found, then error	Unchanged
# jj, P2=xxxx x110	Last to # kk	Unchanged
	If # kk not found, then error	Unchanged

Annex D

(informative)

Use of the basic encoding rules of ASN.1

D.1 BER-TLV data object

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

— 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.

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

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

D.2 Tag field

The bits b8 and b7 of the leading byte of the tag field shall encode the tag class, i.e., the class of the data object.

— b8-b7=00 introduces a tag of universal class.

— b8-b7=01 introduces a tag of application class.

— b8-b7=10 introduces a tag of context-specific class.

— b8-b7=11 introduces a tag of private class.

The bit b6 of the leading byte of the tag field shall encode the tag type, i.e., the type of the data object.

— b6=0 introduces a primitive data object.

— b6=1 introduces a constructed data object.

If the bits b5 to b1 of the leading byte are not all set to 1, then they shall encode an integer equal to the tag number which therefore lies in the range from 0 to 30. Then the tag field consists of a single byte.

Otherwise (b5 to b1 set to 1 in the leading byte), the tag field shall continue on one or more subsequent bytes.

— The bit b8 of each subsequent byte shall be set to 1, unless it is the last subsequent byte.

— The bits b7 to b1 of the first subsequent byte shall not be all set to 0.

— The bits b7 to b1 of the first subsequent byte, followed by the bits b7 to b1 of each further subsequent byte, up to and including the bits b7 to b1 of the last subsequent byte, shall encode an integer equal to the tag number (thus strictly positive).

D.3 Length field

In short form, the length field consists of a single byte where the bit b8 shall be set to 0 and the bits b7 to b1 shall encode an integer equal to the number of bytes in the value field. Any length from 0 to 127 can thus be encoded by 1 byte.

In long form, the length field consists of a leading byte where the bit b8 shall be set to 1 and the bits b7 to b1 shall not be all equal, thus encoding a positive integer equal to the number of subsequent bytes in the length field. Those subsequent bytes shall encode an integer equal to the number of bytes in the value field. Any length within the APDU limit (up to 65 535) can thus be encoded by 3 bytes.

NOTE — ISO/IEC 7816 does not use the indefinite lengths specified by the basic encoding rules of ASN.1 (see ISO/IEC 8825).

D.4 Value field

In this part of ISO/IEC 7816, the value field of some primitive BER-TLV data objects consists of zero, one or more SIMPLE-TLV data objects.

The value field of any other primitive BER-TLV data object consists of zero, one or more data elements fixed by the specifications of the data object.

The value field of each constructed BER-TLV data object consists of zero, one or more BER-TLV data objects.

Annex E

(informative)

Examples of card profiles

E.1 Introduction

This annex defines a number of card profiles to guide application designers in selecting commands to use in their applications. The profiles may also be used to help specify the features desired in a card. Card profiles may be combined.

E.2 Profile M

Cards of this profile have as a minimum the following features and commands.

— File structures

- Transparent structure.
- Linear structure with records of fixed length.

— Commands

- READ BINARY and UPDATE BINARY with
  P1, b8 = 0,
  Lengths up to 256 bytes.
- READ RECORD(S) and UPDATE RECORD with
  P2, b8 to b4 = 0,
  P2, b3 = 1,
  P2, b3 b2 b1 = 000, 001, 010 or 011 and P1 = 0.
- SELECT FILE with
  P1-P2 = '0000'.
- VERIFY with
  P1-P2 = '0001' or '0002'.
- INTERNAL AUTHENTICATE with
  P1-P2 = '0000'.

E.3 Profile N

This profile is the same as M, plus the additional option P1 = '04' in the SELECT FILE command.

E.4 Profile O

Cards of this profile have as a minimum the following features and commands.

— File structures

- Transparent structure.
- Linear structure with records of fixed length.
- Linear structure with records of variable length.
- Cyclic structure with records of fixed length.

— Commands

- READ BINARY, WRITE BINARY and UPDATE BINARY with
  P1, b8 = 0,
  Lengths up to 256 bytes.
- READ RECORD(S), WRITE RECORD and UPDATE RECORD with
  P2, b8 to b4 = 0,
  P2, b3 = 1,
  P2, b3 b2 b1 = 000, 001, 010 or 011 and P1 = 0.
- APPEND RECORD with
  P1-P2 = '0000'.
- SELECT FILE with
  P1 = '00', '01', '02', '03', '04', '08' or '09',
  P2 = '00'.
- VERIFY with
  P1-P2 = '0001' or '0002'.
- INTERNAL AUTHENTICATE with
  P1-P2 = '0000'.
- EXTERNAL AUTHENTICATE with
  P1-P2 = '0000'.
- GET CHALLENGE with
  P1-P2 = '0000'.

E.5 Profile P

Cards of this profile have as a minimum the following features and commands.

— File structures

- Transparent structure.

— Historical bytes

- Card service data (= '3188').
- Initial access data (= '4164').

— Commands

- READ BINARY and UPDATE BINARY with
  P1, b8 = 0,
  Lengths up to 64 bytes.
- SELECT FILE with
  P1-P2 = '0400'.
- VERIFY with
  P1-P2 = '0001' or '0002'.
- INTERNAL AUTHENTICATE with
  P1-P2 = '0000'.

E.6 Profile Q

Cards of this profile have as a minimum the following features and commands.

— Historical bytes

- Initial access data (= '45'-GET).
- Card capabilities (= '7180').

— Secure messaging

— Commands

- GET DATA and PUT DATA with
  Tag in P1-P2,
- SELECT FILE with
  P1-P2 = '0401, '0402' or '0403'.
- VERIFY with
  P1 = '00'.
- INTERNAL AUTHENTICATE
- EXTERNAL AUTHENTICATE
- GET CHALLENGE

Annex F

(informative)

Use of secure messaging

F.1 Abbreviations

For the purpose of this annex, the following abbreviations apply.

CC	Cryptographic checksum
CG	Cryptogram
CH	Command header (CLA INS P1 P2)
CR	Control reference
FR	File reference
KR	Key reference
L	Length
PB	Padding bytes ('80' followed by 0 to k-1 times '00' where k is the block length)
PI	Padding indicator byte
PV	Plain value
RD	Response descriptor
T	Tag
||	Concatenation

For all the examples, CLA indicates the use of secure messaging by an appropriate value ('0X', '8X', '9X' or 'AX') where bit b4 of CLA is set to 1 (see 5.4.1 and table 9).

F.2 Use of cryptographic checksums

The use of cryptographic checksums (see 5.6.3.1) is shown for the four cases defined in table 4 and figure 4.

— Case 1 — No data, no data

Command data field = T_{CC} || L_{CC} || CC

Data covered by CC (b3=1 in CLA) =
First and only data block = CH || PB

The command of case 1 is transformed into a command of case 3.

— Case 2 — No data, data

Command data field = T_{CC} || L_{CC} || CC

Data covered by CC (b3=1 in CLA) =
First and only data block = CH || PB

Response data field =
T_{PV} (b1=1) || L_{PV} || PV || T_{CC} || L_{CC} || CC

Data covered by CC = Data blocks =
T_{PV} (b1=1) || L_{PV} || PV || PB

— Case 3.a — Data, no data

Command data field =
T_{PV} (b1=1) || L_{PV} || PV || T_{CC} || L_{CC} || CC

Data covered by CC (b3=0 in CLA) = Data blocks =
T_{PV} (b1=1) || L_{PV} || PV || PB

— Case 3.b — Data, no data

Command data field =
T_{PV1} (b1=0) || L_{PV1} || PV1 || T_{PV2} (b1=1) || L_{PV2} || PV2 || T_{CC} || L_{CC} || CC

Data covered by CC (b3=1 in CLA) = Data blocks =
CH || PB || T_{PV2} (b1=1) || L_{PV2} || PV2 || PB

— Case 4 — Data, data

Command data field =
T_{PV} (b1=1) || L_{PV} || PV || T_{CC} || L_{CC} || CC

Data covered by CC (b3=0 in CLA) = Data blocks =
T_{PV} (b1=1) || L_{PV} || PV || PB

Response data field =
T_{PV} (b1=1) || L_{PV} || PV || T_{CC} || L_{CC} || CC

Data covered by CC = Data blocks =
T_{PV} (b1=1) || L_{PV} || PV || PB

F.3 Use of cryptograms

The use of cryptograms (see 5.6.4) is shown with and without padding.

— Case a — Plain data not coded in BER-TLV

Command data field = T_{CG} II L_{CG} II PI II CG

Data carried by CG = Data blocks =
Non BER-TLV coded data and padding bytes, if indicated in PI

— Case b — Plain data coded in BER-TLV

Command data field = T_{CG} II L_{CG} II CG

Data carried by CG = String of concealed bytes =
BER-TLV data objects (padding depending on the algorithm and its mode of operation)

F.4 Use of control references

The use of control references (see 5.6.5.1) is shown.

Command data field = T_{CR} II L_{CR} II CR
Where CR = T_{FR} II L_{FR} II FR II T_{KR} II L_{KR} II KR

F.5 Use of response descriptor

The use of response descriptor (see 5.6.5.2) is shown.

Command data field = T_{RD} II L_{RD} II RD
Where RD = T_{PV} II '00' II T_{CC} II '00'

Response data field =
T_{PV} II L_{PV} II PV II T_{CC} II L_{CC} II CC

F.6 Use of the ENVELOPE command

The use of the ENVELOPE command (see 7.2) is shown.

Command data field = T_{CG} II L_{CG} II PI II CG

Data carried by CG = Command APDU starting by CH and padding bytes according to PI

Response data field = T_{CG} II L_{CG} II PI II CG

Data carried by CG = Response APDU and padding bytes according to PI

ICS 35.240.40

Descriptors: data processing, information interchange, identification cards, IC cards, messages, security techniques, authentication.

Price based on 46 pages