INTERNATIONAL STANDARD

ISO/IEC 7816-4

Third edition
2013-04-15

Identification cards — Integrated circuit cards —

Part 4: Organization, security and commands for interchange

Cartes d'identification — Cartes à circuit intégré —

Partie 4: Organisation, sécurité et commandes pour les échanges

Reference number
ISO/IEC 7816-4:2013(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. 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.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 7816-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 17, *Cards and personal identification*.

This third edition cancels and replaces the second edition (ISO/IEC 7816-4:2005), and incorporates the Amendment ISO/IEC 7816-4:2005/Amd.1:2008.

ISO/IEC 7816 consists of the following parts, under the general title *Identification cards — Integrated circuit cards*:

— *Part 1: Cards with contacts — Physical characteristics*

— *Part 2: Cards with contacts — Dimensions and location of the contacts*

— *Part 3: Cards with contacts — Electrical interface and transmission protocols*

— *Part 4: Organization, security and commands for interchange*

— *Part 5: Registration of application providers*

— *Part 6: Interindustry data elements for interchange*

— *Part 7: Interindustry commands for Structured Card Query Language (SCQL)*

— *Part 8: Commands for security operations*

— *Part 9: Commands for card management*

— *Part 10: Electronic signals and answer to reset for synchronous cards*

— *Part 11: Personal verification through biometric methods*

— *Part 12: Cards with contacts — USB electrical interface and operating procedures*

— *Part 13: Commands for application management in a multi-application environment*

— *Part 15: Cryptographic information application*

Introduction

ISO/IEC 7816 [6] is a series of standards specifying integrated circuit cards and the use of such cards for interchange. These cards are identification cards intended for information exchange negotiated between the outside world and the integrated circuit in the card. As a result of an information exchange, the card delivers information (computation result, stored data), and/or modifies its content (data storage, event memorization).

— Five parts are specific to cards with galvanic contacts and three of them specify electrical interfaces.
  — ISO/IEC 7816-1 specifies physical characteristics for cards with contacts.
  — ISO/IEC 7816-2 specifies dimensions and location of the contacts.
  — ISO/IEC 7816-3 specifies electrical interface and transmission protocols for asynchronous cards.
  — ISO/IEC 7816-10 specifies electrical interface and answer to reset for synchronous cards.
  — ISO/IEC 7816-12 specifies electrical interface and operating procedures for USB cards.
— All the other parts are independent from the physical interface technology. They apply to cards accessed by contacts and/or by radio frequency.
  — ISO/IEC 7816-4 specifies organization, security and commands for interchange.
  — ISO/IEC 7816-5 specifies registration of application providers.
  — ISO/IEC 7816-6 specifies interindustry data elements for interchange.
  — ISO/IEC 7816-7 specifies commands for structured card query language.
  — ISO/IEC 7816-8 specifies commands for security operations.
  — ISO/IEC 7816-9 specifies commands for card management.
  — ISO/IEC 7816-11 specifies personal verification through biometric methods.
  — ISO/IEC 7816-13 specifies commands for handling the life cycle of applications.
  — ISO/IEC 7816-15 specifies cryptographic information application.

ISO/IEC 10536 [15] specifies access by close coupling. ISO/IEC 14443 [18] and ISO/IEC 15693 [20] specify access by radio frequency. Such cards are also known as contactless cards.

ISO and IEC draw attention to the fact that it is claimed that compliance with this document may involve the use of the following patents and the foreign counterparts.

— JPN 2033906, Portable electronic device
— JPN 2557838, Integrated circuit card
— JPN 2537199, Integrated circuit card
— JPN 2856393, Portable electronic device

— JPN 2137026, Portable electronic device

— JPN 2831660, Portable electronic device

— DE 198 55 596, Portable microprocessor-assisted data carrier that can be used with or without contacts

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured ISO and IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applications throughout the world. In this respect, the statements of the holders of these patent rights are registered with ISO and IEC. Information may be obtained from:

Contact	Patent details
Toshiba Corporation Intellectual Property Division 1-1, Shibaura 1-Chome Minato-ku, Tokyo 105-8001, Japan	JPN 2033906 (priority date: 1986-02-18; publication date: 1996-03-19), FRA 8614996, KOR 44664 JPN 2557838 (priority date: 1986-02-18; publication date: 1996-09-05), FRA 8700343, GER 3700504, KOR 42243, USA 4841131 JPN 2537199 (priority date: 1986-06-20; publication date: 1996-07-08), FRA 8708646, FRA 8717770, USA 4833595, USA 4901276 JPN 2856393 (priority date: 1987-02-17; publication date: 1998-11-27), FRA 8801887, KOR 43929, USA 4847803 JPN 2137026 (priority date: 1987-02-20; publication date: 1998-06-26), JPN 3054119, FRA 8802046, KOR 44393, USA 4891506 JPN 2831660 (priority date: 1988-08-26; publication date: 1998-09-25), FRA 8911249, KOR 106290, USA 4988855
Orga Kartensysteme Gmbh Am Hoppenhof 33 D-33104 Paderborn Germany	DE 198 55 596 (priority date: 1998-12-02; publication date: 2000-06-29) Applications pending in Europe, Russia, Japan, China, USA, Brazil, Australia

Identification cards — Integrated circuit cards —

Part 4:
Organization, security and commands for interchange

1 Scope

This part of ISO/IEC 7816 is intended to be used in any sector of activity. It specifies:

— contents of command-response pairs exchanged at the interface,

— means of retrieval of data elements and data objects in the card,

— structures and contents of historical bytes to describe operating characteristics of the card,

— structures for applications and data in the card, as seen at the interface when processing commands,

— access methods to files and data in the card,

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

— means and mechanisms for identifying and addressing applications 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 or the outside world.

This part of ISO/IEC 7816 is independent from the physical interface technology. It applies to cards accessed by one or more of the following methods: contacts, close coupling and radio frequency. If the card supports simultaneous use of more than one physical interface, the relationship between what happens on different physical interfaces is out of the scope of this edition of ISO/IEC 7816-4.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 7816-3, *Identification cards — Integrated circuit cards — Part 3: Cards with contacts: Electrical interface and transmission protocols*

ISO/IEC 7816-6, *Identification cards — Integrated circuit cards — Part 6: Interindustry data elements for interchange*

ISO/IEC 8825-1:2002, *Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
access rule
data element containing an access mode referring to an action and security conditions to fulfil before acting

3.2
Answer-to-Reset file
optional EF indicating operating characteristics of the card, also known as Information file

3.3
application
structures, data elements and program modules needed for performing a specific functionality

3.4
application DF
dedicated file (DF) hosting an application in a card

3.5
application identifier
data element (up to sixteen bytes) that identifies an application

3.6
application label
data element for use at the man-machine interface

3.7
application provider
entity providing the components that make up an application in the card

3.8
application template
set of application-relevant data objects including one application identifier data object

3.9
asymmetric cryptographic technique
cryptographic technique that uses two related operations: a public operation defined by public numbers or by a public key and a private operation defined by private numbers or by a private key (the two operations have the property that, given the public operation, it is computationally infeasible to derive the private operation)

3.10
base template
value field of constructed data object excluding DOs resulting from resolution of indirect referencing

3.11
certificate
digital signature binding a particular person or object and its associated public key (the entity issuing the certificate also acts as tag allocation authority with respect to the data elements in the certificate)

3.12
command-response pair
set of two messages at the interface: a command APDU followed by a response APDU in the opposite direction

3.13
command chaining
means used by the outside world to tell the card that the command data of a sequence of successive command-response pairs must be processed together

3.14
context-specific class
class of a tag with its first or only byte from '80' to 'BF'

3.15
current template
sequence of data objects which can be directly referenced by their sole tag in a command for interchange, i.e. the value field of the current constructed DO (which may be virtual)

3.16
data element
item of information seen at the interface for which are specified a name, a description of logical content, a format and a coding

3.17
data object
information seen at the interface consisting of the concatenation of a mandatory tag field, a mandatory length field and a conditional value field

3.18
data unit
smallest set of bits that can be unambiguously referenced within an EF supporting data units

3.19
dedicated file
structure containing file control information and, optionally, memory available for allocation

3.20
DF name
data element (up to sixteen bytes) that uniquely identifies a DF in the card

3.21
digital signature
data appended to, or cryptographic transformation of, a data string that proves the origin and the integrity of the data string and protects against forgery, e.g. by the recipient of the data string

3.22
directory file
optional EF containing a list of applications supported by the card and optional related data elements

3.23
elementary file
set of data units or records or data objects sharing the same file identifier

3.24
extended header
data element referencing one or several DOs in a constructed DO

3.25
extended header list
concatenation of extended headers

3.26
file
structure for application and/or data in the card, as seen at the interface when processing commands

3.27
file identifier
data element (two bytes) used to address a file

3.28
header list
concatenation of pairs of tag field and length field without delimitation

3.29
information file
optional EF indicating operating characteristics of the card, also known as Answer-to-Reset file

3.30
interindustry
standardized in the ISO/IEC 7816 [6] series of standards

3.31
internal EF
EF for storing data interpreted by the card

3.32
key
sequence of symbols controlling a cryptographic operation (e.g. encipherment, decipherment, a private or a public operation in a dynamic authentication, signature generation, signature verification)

3.33
master file
unique DF representing the root in a card using a hierarchy of DFs

3.34
offset
number sequentially referencing a data unit in an EF supporting data units, or a byte in a record

3.35
oversize payload
payload which exceeds the current size constraints of the APDU

3.36
parent file
DF immediately preceding a given file within a hierarchy of DFs

3.37
password
data that may be required by the application to be presented to the card by its user for authentication purpose

3.38
path
concatenation of file identifiers without delimitation

3.39
payload
data of arbitrary length, to be sent to the card or by the card, in order to be processed together

3.40
private key
that key of an entity's asymmetric key pair that should only be used by that entity

[ISO/IEC 9798-1 [10]]

3.41
provider
authority who has or who obtained the right to create a DF in the card

3.42
public key
that key of an entity's asymmetric key pair that can be made public

[ISO/IEC 9798-1 [10]]

3.43
record
string of bytes referenced and handled by the card within an EF supporting records

3.44
record identifier
number used to reference one or more records within an EF supporting records

3.45
record number
sequential number that uniquely identifies each record within an EF supporting records

3.46
registered application provider identifier
data element (five bytes) that uniquely identifies an application provider

3.47
resetting code
data to be presented to a card in order to modify the value of a counter

3.48
response chaining
means used by the card to tell the outside world that the response data of any command-response pair followed by the response data of a sequence of GET RESPONSE command-response pairs should be processed together

3.49
secret key
key used with symmetric cryptographic techniques by a set of specified entities

[ISO/IEC 11770-3 [17]]

3.50
secure messaging
set of means for cryptographic protection of (parts of) command-response pairs

3.51
security attribute
condition of use of objects in the card including stored data and data processing functions, expressed as a data element containing one or more access rules

3.51
security environment
set of components required by an application in the card for secure messaging or for security operations

3.52
self-controlled DO
constructed DO which nests at least a DO'62' nesting security attributes

3.53
short EF identifier
data element (five bits) used to address an elementary file

3.54
structure
DF, EF, record, DataString or DO

3.55
symmetric cryptographic technique
cryptographic technique using the same secret key for both the originator's and the recipient's operation (without the secret key, it is computationally infeasible to compute either operation)

3.56
tag list
concatenation of tag fields without delimitation

3.57
tagged wrapper
wrapper which provides a tag for local addressing of the DO it references

3.58
template
concatenation of BER-TLV data objects, forming the value field of a constructed BER-TLV data object

3.59
template extension
part of the value field of a constructed DO resulting from automatic resolution of indirect referencing

3.60
transient selection
structure selection needed when performing a C-RP, the success of which will not modify the current validity area

3.61
transient VA
validity area (VA) set transiently during execution of a command which handles DOs

3.62
user
user of the card, also known as cardholder

3.63
validity area
result of all successful selections performed on a logical channel

3.64
virtual root DO
virtual constructed DO'7F70' made current by the selection of a file, a record or a DataString supporting DO handling

3.65
wrapper
concatenation of DOs referencing a DO

3.66
working EF
EF for storing data not interpreted by the card

4 Symbols and abbreviated terms

AID	application identifier
AMB	access mode byte
AMF	access mode field
APDU	application protocol data unit
ARR	access rule reference
ASN.1	abstract syntax notation one (see ISO/IEC 8825-1)
AT	control reference template for authentication
ATR	Answer-to-Reset
BER	basic encoding rules of ASN.1 (see ISO/IEC 8825-1)
CCT	control reference template for cryptographic checksum
CLA	class byte
CRT	control reference template
CT	control reference template for confidentiality
CP	control parameter (file control parameter or data object control parameter)
CP DO	control parameter BER-TLV data object
C-RP	command-response pair
DF	dedicated file
DIR	directory
DO	BER-TLV data object
DO'…'	BER-TLV data object, the tag of which is a hexadecimal value given between simple quotes
DST	control reference template for digital signature
EF	elementary file
EF.ARR	access rule reference file
EF.ATR/INFO	Answer-to-Reset file, or Information file
EF.DIR	directory file
FCI	file control information
FCP	file control parameter
FMD	file management data
HT	control reference template for hash-code
INS	instruction byte
KAT	control reference template for key agreement

L_c field	length field for coding the number N_c
LCS	Life cycle status
L_e field	length field for coding the number N_e
MF	master file
N_c	number of bytes in the command data field
N_e	maximum number of bytes expected in the response data field
N_r	number of bytes in the response data field
OID	object identifier, as defined by ISO/IEC 8825-1
PIX	proprietary application identifier extension
P1-P2	parameter bytes (inserted for clarity, the dash is not significant)
RFU	reserved for future use by ISO/IEC JTC 1/SC 17
RID	registered application provider identifier
SC	security condition
SCB	security condition byte
SCQL	structured card query language
SE	security environment
SEID	security environment identifier
SM	secure messaging
SPT	security parameter template
SW1-SW2	status bytes (inserted for clarity, the dash is not significant)
TLV	tag, length, value
{T-L-V}	data object (inserted for clarity, the dashes and curly brackets are not significant)
VA	validity area
'XY'	notation where any upper case letter from G to Z stands for a hexadecimal digit from '0' to '9' or 'A' to 'F', equal to XY to the base 16. Identical letters do not state that the most and least significant quartets are identical.

5 Command-Response pairs

5.1 Conditions of operation

A physical interface between the card and the outside world shall be enabled to support processing of command-response pairs. The enabling process is defined by interface-specific protocols. With their own words (between double quotes), the following standards define enabling procedures:

— ISO/IEC 7816-3 defines "activation of contacts", "cold reset" or "warm reset", and possible "Protocol and Parameter Selection".

— ISO/IEC 7816-12 defines "electrical connection" of contacts with reference to the USB specification, the "Configured" and "Initial" state of the device, and "ATR".

— ISO/IEC 14443 [18] (all parts) defines how to set a proximity card to an "ACTIVE" state.

Those standards also define procedures and situations which disable the physical interface. A disabled physical interface shall not support processing of command-response pairs.

5.2 Syntax

Table 1 shows a command-response pair (abbreviated in this document by C-RP), namely a command APDU followed by a response APDU in the opposite direction (see ISO/IEC 7816-3). There shall be no interleaving of C-RPs across the interface, i.e. the response APDU shall be received before initiating another C-RP.

In any command APDU comprising both L_c and L_e fields (see ISO/IEC 7816-3), short and extended length fields shall not be combined: either both of them are short, or both of them are extended.

If the card explicitly states its capability of handling "extended L_c and L_e fields" (see Table 119, third software function table) in the historical bytes (see 12.1.1) or in EF.ATR/INFO (see 12.2.2), then the card handles short and extended length fields. Otherwise (default value), the card handles only short length fields. In a command APDU longer than 5 bytes, the use of extended L fields is indicated by the first byte after P2 equal to '00'.

N_c denotes the number of bytes in the command data field. The L_c field encodes N_c.

— If the L_c field is absent, then N_c is zero.

— A short L_c field consists of one byte not set to '00'. From '01' to 'FF', the byte encodes N_c from one to 255.

— An extended L_c field consists of three bytes: one byte set to '00' followed by two bytes not set to '0000'. From '0001' to 'FFFF', the two bytes encode N_c from one to 65 535.

Table 1 — Command-response pair (C-RP)

Field	Description	Number of bytes	Direction
Command header	Class byte denoted CLA	1	To the card
	Instruction byte denoted INS	1	
	Parameter bytes denoted P1-P2	2	
L_c field	Absent for encoding $N_c = 0$, present for encoding $N_c > 0$	0, 1 or 3	
Command data field	Absent if $N_c = 0$, present as a string of N_c bytes if $N_c > 0$	N_c	
L_e field	Absent for encoding $N_e = 0$, present for encoding $N_e > 0$	0, 1, 2 or 3	
Response data field	Absent if $N_r = 0$, present as a string of N_r bytes if $N_r > 0$	N_r (at most N_e)	From the card
Response trailer	Status bytes denoted SW1-SW2	2	

N_e denotes the maximum number of bytes expected in the response data field, regardless of any structure of the data within this field. The L_e field encodes N_e.

— If the L_e field is absent, then N_e is zero.

— A short L_e field consists of one byte with any value.

  — From '01' to 'FF', the byte encodes N_e from one to 255.

  — If the byte is set to '00', then N_e is 256.

— An extended L_e field consists of either three bytes (one byte set to '00' followed by two bytes with any value) if the L_c field is absent, or two bytes (with any value) if an extended L_c field is present.

  — From '0001' to 'FFFF', the two bytes encode N_e from one to 65 535.

  — If the two bytes are set to '0000', then N_e is 65 536.

N_r denotes the number of bytes in the response data field. N_r shall be less than or equal to N_e. Therefore in any C-RP, the absence of L_e field is the standard way for receiving no response data field. If the L_e field contains only bytes set to '00', then N_e is maximum, i.e. within the limit of 256 for a short L_e field, or 65 536 for an extended L_e field, all the available bytes should be returned.

If the process is aborted, then the card may become unresponsive. However, if a response APDU occurs, then the response data field shall be absent and SW1-SW2 shall indicate an error.

P1-P2 indicates controls and options for processing the command. A parameter byte P1 or P2 set to '00' generally provides no further qualification. There is no other general convention for encoding the parameter bytes.

General conventions are specified hereafter for encoding the class byte denoted CLA (see 5.4), the instruction byte denoted INS (see 5.5) and the status bytes denoted SW1-SW2 (see 5.6). In those bytes, the RFU bits shall be set to 0 unless otherwise specified.

5.3 Chaining procedures

5.3.1 General

Chaining procedures are used either to support payload fragmentation or for a process involving several consecutive C-RPs.

5.3.2 Payload fragmentation

A command payload is data of arbitrary length to be sent to the card in order to be processed together. A response payload is data of arbitrary length to be received from the card as requested. A payload is oversize if its length is larger than available in a data field (see NOTE); chaining is needed to transmit such a payload:

— Chaining of commands supports the transmission to the card of an oversize command payload. The payload is fragmented; each fragment is a command data field which complies with size limitations.

— Chaining of responses supports the recovery from the card of an oversize response payload. The payload is fragmented; each fragment is a response data field which complies with size limitations.

The receiver shall concatenate the successively transmitted fragments to recover the payload.

NOTE APDU syntax limits the size of data fields. More size limitations may be stated (see 12.7.1); if an Le field ('0000' or '000000') indicates Ne = '010000', then all the required information should be returned, up to 65 536 bytes. Fragmentation will occur with the short format of length fields if the command payload exceeds 255 bytes and/or the response payload exceeds 256 bytes. Fragmentation should not occur with the extended format of length fields if payloads comply with stated size limitations (see 12.7.1).

5.3.3 Command chaining

This clause specifies a mechanism whereby in the interindustry class (CLA < '80', see Table 2 and Table 3) consecutive C-RPs can be chained. If the card supports the mechanism, then it shall indicate it (see

Table 119, third software function table) in the historical bytes (see 12.1.1) or in EF.ATR/INFO (see 12.2.2).

NOTE The contents of this clause do not depend on the transmission protocol. ISO/IEC 7816-3 describes chaining when using protocol T=0.

For chaining in the interindustry class, bit b5 of CLA shall be used while the other seven bits are constant.

— If bit b5 is set to 0, then the command is the only or last command of a chain.

— If bit b5 is set to 1, then the command is not the last command of a chain.

The mechanism may be used:

— to transmit an oversize command payload:

  — all CLA bytes of the commands shall be the same, except for bit b5 (see above).

    — If bit b5 is set to 1, then the L_e field shall be absent.

    - If bit b5 is set to 0, then an L_e field may be present.
  - all INS P1 P2 bytes of the commands shall be the same.
- As specified elsewhere in this document (see example in Annex C).
- For any application-defined process involving several consecutive C-RPs
  - bit b5 of CLA shall be used as defined above.
  - the logical channel indicated by CLA shall be the same.
  - there is no constraint on the values of INS P1 P2 bytes of the commands.

This document specifies the card behaviour only in the case where, once initiated, a chain of C-RPs is successfully processed before initiating a C-RP not part of the chain. If this condition is not respected, the card behaviour is out of scope of this document, but may be described in a specification.

In response to a command that is not the last command of a chain, SW1-SW2 set to '9000' means that the process has been completed so far; the use of warning indications is defined in see 5.6; the following specific error conditions may occur.

- If SW1-SW2 is set to '6883', then the last command of the chain is expected.
- If SW1-SW2 is set to '6884', then command chaining is not supported.

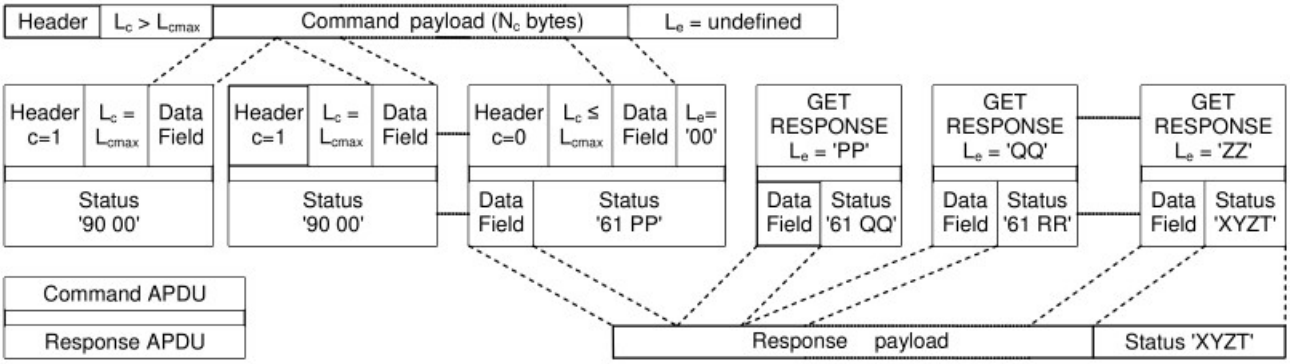
5.3.4 Response chaining

SW1 = '61' and GET RESPONSE support the transmission of an oversized response payload:

- all CLA bytes of the commands involved in response chaining shall be the same. By definition response chaining starts with that C-RP where SW1 is set to '61'.
- with the exception of the first command APDU of the sequence, all INS P1-P2 bytes of the command APDUs shall be 'C0 00 00' (GET RESPONSE).
- as in command chaining, payload transmission shall be interrupted by any C-RP different from GET RESPONSE, or by any C-RP on another logical channel.
- contrary to command chaining, this possible termination is normal when the outside world considers that it has received enough data: the exchange should continue normally with an arbitrary C-RP.
- however, this document does not define the behaviour of the card if the outside world tries to resume response chaining after e.g. the execution of a command on another logical channel. The behaviour may be defined by the application.

If defined by the application, response chaining may be used for checking the availability of data, which are not necessarily required. The card may indicate availability by SW1-SW2 = '61XY', without sending any response data; the outside world will or will not send (a) GET RESPONSE (s).

In Figure 1, c= 0 and c=1 indicates the value of bit b5 of CLA. Lcmax indicates the maximum value of short or extended Lc field the ICC supports.



WARNING — This is an example where the interface device chooses to send the longest command data fields the card supports, and to recover all data made available by the card. None of those choices is mandated.

Figure 1 — Transmission to the card of an oversize command payload using command chaining, followed by the transmission by the card of an oversize response payload using response chaining.

5.4 Class byte

5.4.1 Coding

CLA indicates the class of the command. Bit b8 of CLA distinguishes between the interindustry class and the proprietary class.

— Bit b8 set to 0 indicates the interindustry class.

— Bit b8 set to 1 indicates the proprietary class, except for the value 'FF' which is invalid due to specifications in ISO/IEC 7816-3. The application-context defines the other bits of CLA in proprietary class.

The values 000x xxxx and 01xx xxxx are specified hereafter. The values 001x xxxx are RFU.

Table 2 specifies 000x xxxx as the first interindustry values.

— Bits b8, b7 and b6 are set to 000.

— Bit b5 controls command chaining (see 5.3.3).

— Bits b4 and b3 indicate secure messaging (see clause 10).

— Bits b2 and b1 encode a logical channel number from zero to three (see 5.4.2).

Table 2 — First interindustry values of CLA

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	**x**	-	-	-	-	**Command chaining control** (see 5.3.3)
0	0	0	0	-	-	-	-	— The command is the last or only command of a chain
0	0	0	1	-	-	-	-	— The command is not the last command of a chain
0	0	0	-	**x**	**x**	-	-	**Secure messaging indication**
0	0	0	-	0	0	-	-	— No SM or no indication
0	0	0	-	0	1	-	-	— Proprietary SM format
0	0	0	-	1	0	-	-	— SM according to clause 10, command header not processed according to 10.2.3.1
0	0	0	-	1	1	-	-	— SM according to clause 10, command header authenticated according to 10.2.3.1
0	0	0	-	-	-	**x**	**x**	**Logical channel number from zero to three** (see 5.4.2)

Table 3 — Further interindustry values of CLA

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	1	**x**	-	-	-	-	-	**Secure messaging indication**
0	1	0	-	-	-	-	-	— No SM or no indication
0	1	1	-	-	-	-	-	— SM according to clause 10, command header not processed according to 10.2.3.1
0	1	-	**x**	-	-	-	-	**Command chaining control** (see 5.3.3)
0	1	-	0	-	-	-	-	— The command is the last or only command of a chain
0	1	-	1	-	-	-	-	— The command is not the last command of a chain
0	1	-	-	**x**	**x**	**x**	**x**	**Logical channel number from four to nineteen** (see 5.4.2)

Table 3 specifies 01xx xxxx as further interindustry values.

— Bits b8 and b7 are set to 01.

— Bit b6 indicates secure messaging (see clause 10).

— Bit b5 controls command chaining (see 5.3.3).

Bits b4 to b1 encode a number from zero to fifteen; this number plus four is the logical channel number from four to nineteen (see 5.4.2).

5.4.2 Logical channels

This clause specifies a mechanism whereby in the interindustry class, C-RPs can refer to logical channels. Each logical channel has its own security status (see 9.3) and validity area (see 7.2). The way to share a security status is out of scope of this document.

If the card supports the mechanism, then it shall indicate the maximum number of available logical channels (see

Table 119, third software function table) in the historical bytes (see 12.1.1) or in EF.ATR/INFO (see 12.2.2).

— If the indicated number is four or less, then only Table 2 applies.

— If the indicated number is five or more, then Table 3 also applies.

For referring to logical channels in the interindustry class, the following rules apply.

— CLA encodes the number of the logical channel of the C-RP.

— Enabling the physical interface (see 5.1) opens the basic logical channel, which shall remain open until disabling of the physical interface. It may be reset (see below). The number of the basic logical channel is zero.

— Cards not supporting additional logical channel(s) (default value) shall use only the basic logical channel.

— Any additional logical channel shall be opened by completion of either a SELECT or a SELECT DATA command (see 11.1.1 and 11.4.2) where CLA encodes the number of a logical channel not yet in use, or a MANAGE CHANNEL command with open function (see 11.1.2).

— Any additional logical channel can be closed by the completion of a MANAGE CHANNEL command with close function. (see 11.1.2). After closing, the logical channel shall be available for re-use.

— Even if more than one logical channel is opened there shall be no interleaving of C-RPs (see 5.2).

— If sharability is not explicitly excluded by the file descriptor byte (see bit b7 in Table 11), more than one logical channel may be opened to the same structure (see clause 7), i.e. to a DF, possibly an application DF, and also possibly to an EF.

— If sharability is not explicitly excluded by the data descriptor byte (see bit b7 in Table 13) more than one logical channel may be opened to the same DO.

— Any logical channel can be reset by the completion of a MANAGE CHANNEL command with reset function (see 11.1.2).

5.5 Instruction byte

INS indicates the command to process. Due to specifications in ISO/IEC 7816-3, the values '6X' and '9X' are invalid.

Table 4 lists all the commands specified in ISO/IEC 7816 [6] at the time of publication.

— Table 4.1, i.e. the left side, lists the command names in the alphabetic order.

— Table 4.2, i.e. the right side, lists the INS codes in the numeric order.

ISO/IEC 7816 [6] specifies the use of those commands in the interindustry class.

— This document specifies commands for interchange (see clause 11).

— ISO/IEC 7816-7 specifies commands for structured card query language (SCQL).

— ISO/IEC 7816-8 specifies commands for security operations.

— ISO/IEC 7816-9 specifies commands for card management.

— ISO/IEC 7816-13 specifies commands for application management in a multi-application environment.

In the interindustry class, bit b1 of INS indicates a data field format as follows.

— If bit b1 is set to 0 (even INS code), then no indication is provided.

— If bit b1 is set to 1 (odd INS code), payloads (if any) shall be encoded in BER-TLV (see 8.1).

5.6 Status bytes

SW1-SW2 indicates the processing state. Due to specifications in ISO/IEC 7816-3, any value different from '6XXX' and '9XXX' is invalid; any value '60XX' is also invalid.

The values '61XX', '62XX', '63XX', '64XX', '65XX', '66XX', '68XX', '69XX', '6AXX' and '6CXX' are interindustry. Due to specifications in ISO/IEC 7816-3, the values '67XX', '6BXX', '6DXX', '6EXX', '6FXX' and '9XXX' are proprietary, except the values '6700', '6701', '6702', '6B00', '6D00', '6E00', '6F00' and '9000' that are interindustry.

Figure 2 shows the structural scheme of the values '9000' and '61XX' to '6FXX' for SW1-SW2.

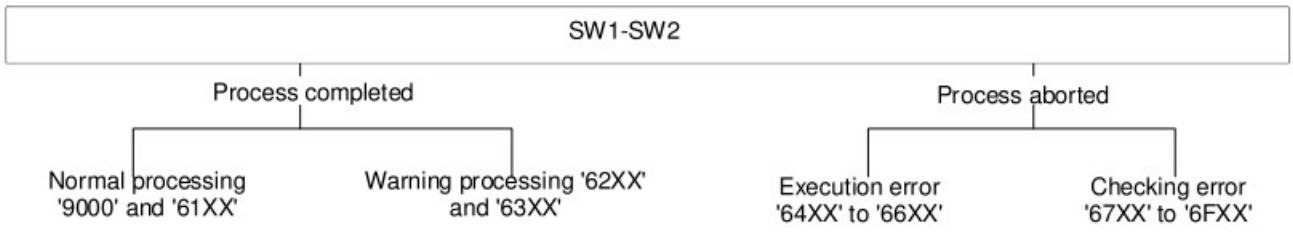


Figure 2 — Structural scheme of values of SW1-SW2

All interindustry values of SW1-SW2 are independent from any transmission protocol. Table 5 lists all the interindustry values of SW1-SW2 and shows their general meaning. ISO/IEC JTC 1/SC 17 reserves for future use any interindustry value of SW1-SW2 not defined in ISO/IEC 7816 [6]. Table 6 lists all the specific interindustry warning and error conditions used in ISO/IEC 7816 [6] at the time of publication.

Table 4.1 — Commands in the alphabetic order

Command name	INS	See
ACTIVATE FILE	'44'	Part 9
ACTIVATE RECORD	'08'	11.3.9
APPEND RECORD	'E2'	11.3.6
APPLICATION MANAGEMENT REQUEST	'40', '41'	Part 13
CHANGE REFERENCE DATA	'24', '25'	11.5.7
COMPARE	'33'	11.6.1
CREATE FILE	'E0'	Part 9
DEACTIVATE FILE	'04'	Part 9
DEACTIVATE RECORD	'06'	11.3.10
DELETE DATA	'EE'	Part 9
DELETE FILE	'E4'	Part 9
DISABLE VERIFICATION REQUIREMENT	'26'	11.5.9
ENABLE VERIFICATION REQUIREMENT	'28'	0
ENVELOPE	'C2', 'C3'	11.7.2
ERASE BINARY	'0E', '0F'	11.2.7
ERASE RECORD (S)	'0C'	11.3.8
EXTERNAL (/MUTUAL) AUTHENTICATE	'82'	11.5.4
GENERAL AUTHENTICATE	'86', '87'	11.5.5
GENERATE ASYMMETRIC KEY PAIR	'46', '47'	Part 8
GET ATTRIBUTE	'34','35'	11.6.2
GET CHALLENGE	'84'	11.5.3
GET DATA/GET NEXT DATA	'CA'/'CC'	11.4.3
GET DATA/GET NEXT DATA	'CB'/'CD'	11.4.4
GET RESPONSE	'C0'	11.7.1
INTERNAL AUTHENTICATE	'88'	11.5.2
LOAD APPLICATION	'EA', 'EB'	Part 13
MANAGE CHANNEL	'70'	11.1.2
MANAGE DATA	'CF'	Part 9
MANAGE SECURITY ENVIRONMENT	'22'	11.5.11
PERFORM BIOMETRIC OPERATION	'2E', '2F'	Part 8
PERFORM SCQL OPERATION	'10'	Part 7
PERFORM SECURITY OPERATION	'2A', '2B'	Part 8
PERFORM TRANSACTION OPERATION	'12'	Part 7
PERFORM USER OPERATION	'14'	Part 7
PUT DATA	'DA', 'DB'	11.4.6
PUT NEXT DATA	'D8', 'D9'	11.4.7
READ BINARY	'B0', 'B1'	11.2.3
READ RECORD (S)	'B2', 'B3'	11.3.3
REMOVE APPLICATION	'EC', 'ED'	Part 13
RESET RETRY COUNTER	'2C', '2D'	11.5.10
SEARCH BINARY	'A0', 'A1'	11.2.6
SEARCH RECORD	'A2'	11.3.7
SELECT	'A4'	11.1.1
SELECT DATA	'A5'	11.4.2
TERMINATE CARD USAGE	'FE'	Part 9
TERMINATE DF	'E6'	Part 9
TERMINATE EF	'E8'	Part 9
UPDATE BINARY	'D6', 'D7'	11.2.5
UPDATE DATA	'DE', 'DF'	11.4.8
UPDATE RECORD	'DC', 'DD'	11.3.5
VERIFY	'20', '21'	11.5.6
WRITE BINARY	'D0', 'D1'	0
WRITE RECORD	'D2'	11.3.4

Table 4.2 — Commands in the numeric order

INS	Command name	See
'04'	DEACTIVATE FILE	Part 9
'06'	DEACTIVATE RECORD	11.3.10
'08'	ACTIVATE RECORD	11.3.9
'0C'	ERASE RECORD (S)	11.3.8
'0E', '0F'	ERASE BINARY	11.2.7
'10'	PERFORM SCQL OPERATION	Part 7
'12'	PERFORM TRANSACTION OPERATION	Part 7
'14'	PERFORM USER OPERATION	Part 7
'20', '21'	VERIFY	11.5.6
'22'	MANAGE SECURITY ENVIRONMENT	11.5.11
'24', '25'	CHANGE REFERENCE DATA	11.5.7
'26'	DISABLE VERIFICATION REQUIREMENT	11.5.9
'28'	ENABLE VERIFICATION REQUIREMENT	0
'2A', '2B'	PERFORM SECURITY OPERATION	Part 8
'2C', '2D'	RESET RETRY COUNTER	11.5.10
'2E', '2F'	PERFORM BIOMETRIC OPERATION	Part 8
'33'	COMPARE	11.6.1
'34','35'	GET ATTRIBUTE	11.6.2
'40', '41'	APPLICATION MANAGEMENT REQUEST	Part 13
'44'	ACTIVATE FILE	Part 9
'46', '47'	GENERATE ASYMMETRIC KEY PAIR	Part 8
'70'	MANAGE CHANNEL	11.1.2
'82'	EXTERNAL (/MUTUAL) AUTHENTICATE	11.5.4
'84'	GET CHALLENGE	11.5.3
'86', '87'	GENERAL AUTHENTICATE	11.5.5
'88'	INTERNAL AUTHENTICATE	11.5.2
'A0', 'A1'	SEARCH BINARY	11.2.6
'A2'	SEARCH RECORD	11.3.7
'A4'	SELECT	11.1.1
'A5'	SELECT DATA	11.4.2
'B0', 'B1'	READ BINARY	11.2.3
'B2', 'B3'	READ RECORD (S)	11.3.3
'C0'	GET RESPONSE	11.7.1
'C2', 'C3'	ENVELOPE	11.7.2
'CA'/'CC'	GET DATA/GET NEXT DATA	11.4.3
'CB'/'CD'	GET DATA/GET NEXT DATA	11.4.4
'CF'	MANAGE DATA	Part 9
'D0', 'D1'	WRITE BINARY	0
'D2'	WRITE RECORD	11.3.4
'D6', 'D7'	UPDATE BINARY	11.2.5
'D8', 'D9'	PUT NEXT DATA	11.4.7
'DA', 'DB'	PUT DATA	11.4.6
'DC', 'DD'	UPDATE RECORD	11.3.5
'DE', 'DF'	UPDATE DATA	11.4.8
'E0'	CREATE FILE	Part 9
'E2'	APPEND RECORD	11.3.6
'E4'	DELETE FILE	Part 9
'E6'	TERMINATE DF	Part 9
'E8'	TERMINATE EF	Part 9
'EA', 'EB'	LOAD APPLICATION	Part 13
'EE'	DELETE DATA	Part 9
'EC', 'ED'	REMOVE APPLICATION	Part 13
'FE'	TERMINATE CARD USAGE	Part 9

— In the interindustry class, any valid INS code not defined in ISO/IEC 7816 [6] is RFU.

Table 5 — General meaning of the interindustry values of SW1-SW2

	SW1-SW2	Meaning
Normal processing	'9000'	No further qualification
	'61XX'	SW2 encodes the number of data bytes still available (see text below)
Warning processing	'62XX'	State of non-volatile memory is unchanged (further qualification in SW2, see Table 6)
	'63XX'	State of non-volatile memory may have changed (further qualification in SW2, see Table 6)
Execution error	'64XX'	State of non-volatile memory is unchanged (further qualification in SW2, see Table 6)
	'65XX'	State of non-volatile memory may have changed (further qualification in SW2, see Table 6)
	'66XX'	Security-related issues (further qualification in SW2 is RFU)
Checking error	'67XX'	Wrong length (further qualification in SW2, see Table 6)
	'68XX'	Functions in CLA not supported (further qualification in SW2, see Table 6)
	'69XX'	Command not allowed (further qualification in SW2, see Table 6)
	'6AXX'	Wrong parameters P1-P2 (further qualification in SW2, see Table 6)
	'6B00'	Wrong parameters P1-P2
	'6CXX'	Wrong L_e field; SW2 encodes the exact number of available data bytes (see text below)
	'6D00'	Instruction code not supported or invalid
	'6E00'	Class not supported
	'6F00'	No precise diagnosis

If SW1 is set to '61', then the process is completed and before issuing any other command, a GET RESPONSE command may be issued with the same CLA and using SW2 (number of data bytes still available) as short L_e field. If the C-RP which returned '61XY' used the extended length fields, the GET RESPONSE may also use an extended L_e field according to the buffer sizes stated in DO'7F66' (see 12.7.1). This feature is used for response chaining (see 5.3.4).

If SW1 is set to '6C', then the process is aborted and before issuing any other command, the same command may be re-issued using SW2 (exact number of available data bytes) as short L_e field.

If the process is aborted with a value of SW1 from '64' to '6F', then the response data field shall be absent.

If SW1 is set to '63' or '65', then the state of the non-volatile memory may have changed. If SW1 is set to '6X' except for '63' and '65', then the state of the non-volatile memory is unchanged.

When command chaining is used for payload fragmentation (see 5.3.3), in response to a command that is not the last command of a chain (see 5.3.3), interindustry warning indications are prohibited (see also ISO/IEC 7816-3), i.e. SW1 shall be set to neither '62' nor '63'.

Table 6 — Specific interindustry warning and error conditions

SW1	SW2	Meaning
'62' (warning)	'00'	No information given
	'02' to '80'	Triggering by the card (see 12.5.1)
	'81'	Part of returned data may be corrupted
	'82'	End of file or record reached before reading N_e bytes, or unsuccessful search.
	'83'	Selected file deactivated
	'84'	File or data control information not formatted according to 7.4
	'85'	Selected file in termination state
	'86'	No input data available from a sensor on the card
	'87'	At least one of the referenced records is deactivated
'63' (warning)	'00'	No information given
	'40'	Unsuccessful comparison (exact meaning depends on the command)
	'81'	File filled up by the last write
	'CX'	Counter from 0 to 15 encoded by 'X' (exact meaning depends on the command)
'64' (error)	'00'	No information given
	'01'	Immediate response required by the card
	'02' to '80'	Triggering by the card (see 12.5.1)
	'81'	Logical channel shared access denied
	'82'	Logical channel opening denied
'65' (error)	'00'	No information given
	'81'	Memory failure
'66' (error)	'00'	No information given, other values are RFU
'67 (error)	'00'	No information given
	'01'	Command APDU format not compliant with this standard (see 5.1)
	'02'	The value of L_c is not the one expected.
'68' (error)	'00'	No information given
	'81'	Logical channel not supported
	'82'	Secure messaging not supported
	'83'	Last command of the chain expected
	'84'	Command chaining not supported
'69' (error)	'00'	No information given
	'81'	Command incompatible with file structure
	'82'	Security status not satisfied
	'83'	Authentication method blocked
	'84'	Reference data not usable
	'85'	Conditions of use not satisfied
	'86'	Command not allowed (no current EF)
	'87'	Expected secure messaging DOs missing
	'88'	Incorrect secure messaging DOs
'6A' (error)	'00'	No information given
	'80'	Incorrect parameters in the command data field
	'81'	Function not supported
	'82'	File or application not found
	'83'	Record not found
	'84'	Not enough memory space in the file
	'85'	N_c inconsistent with TLV structure
	'86'	Incorrect parameters P1-P2
	'87'	N_c inconsistent with parameters P1-P2
	'88'	Referenced data or reference data not found (exact meaning depending on the command)
	'89'	File already exists
	'8A'	DF name already exists
— Any other value of SW2 is RFU.		

6 Data objects

This clause specifies two categories of data objects: SIMPLE-TLV data objects and BER-TLV data objects, the latter abbreviated by DOs in this document.

6.1 SIMPLE-TLV data objects

Each SIMPLE-TLV data object shall consist of two or three consecutive fields: a mandatory tag field, a mandatory length field and a conditional value field. A record (see 11.3.1) may be a SIMPLE-TLV data object.

— The tag field consists of a single byte encoding a tag number from 1 to 254. The values '00' and 'FF' are invalid for tag fields. If a record is a SIMPLE-TLV data object, then the tag may be used as record identifier.

— The length field consists of one or three consecutive bytes.

  — If the first byte is not set to 'FF', then the length field consists of a single byte encoding a number from zero to 254 and denoted N.

  — If the first byte is set to 'FF', then the length field continues on the subsequent two bytes with any value encoding a number from zero to 65 535 and denoted N.

— If N is zero, there is no value field, i.e. the data object is empty. Otherwise (N > 0), the value field consists of N consecutive bytes.

NOTE This standard defines neither tag values nor value fields of simple-tlv data objects. Hence, simple-tlv data objects addressing cannot be used for interchange.

6.2 BER-TLV data objects

Each BER-TLV data object (DO) consists of two or three consecutive fields (see the basic encoding rules of ASN.1 in ISO/IEC 8825-1): a mandatory tag field a mandatory length field and a conditional value field. Any non empty DO is denoted {T-L-V}.

The tag field consists of one or more consecutive bytes. It indicates a class and an encoding and it encodes a tag number. The value '00' is invalid for the first byte of tag fields. ISO/IEC 7816 [6] supports tag fields of one, two and three bytes; longer tag fields are RFU.

The length field encodes a length, i.e. a number denoted N, according to ISO/IEC 8825-1. If N is zero, there is no value field, i.e. the DO is empty, and noted {T-'00'}. Otherwise (N > 0), the value field consists of N consecutive bytes, and the DO is noted {T-L-V}. ISO/IEC 7816 [6]

a) precludes the use of the “indefinite length” (coded '80'), according to the DER encoding rules;

b) recommends to use the shortest possible coding of the length field, according to DER encoding rules (see ISO/IEC 8825-1);

c) uses length fields consisting of one to five bytes, longer length fields are RFU.

NOTE 1 In order to ascertain the length of the length field, a specification may mandate recommendation b), i.e. the use of shortest length field for the coding of a given length.

NOTE 2 ISO/IEC 7816-4 uses '80' with a specific meaning in the value field of an extended header DO (see 8.4.5).

NOTE 3 Annex E provides the detailed coding of tag and length fields.

6.3 Constructed DOs versus primitive DOs

A constructed, non empty DO is denoted {T-L-{T1-L1-V1} … - {Tn-Ln-Vn}}. Its tag T indicates the structure of its value field (see Annex E). This value field is called a template, which may:

— either consist of one DO, called "nested" in the constructed DO.

— or consist of a concatenation of several nested DOs, (n DOs in the example above), without padding (see 8.1.1).

Unless otherwise specified (e.g. wrapper (see 8.4.8) or tagged wrapper (see 8.4.9), ISO/IEC 7816-15, ISO/IEC 24727 [23]), the order of DOs within a template is not defined in this document.

See Annex E for the identification of a primitive or constructed data object by the first byte of its tag. A possible structure of the value field of a primitive data object is to be defined elsewhere.

7 Structures for applications and data

7.1 Available structures

This clause specifies structures for applications and data, as seen at the interface when processing commands in the interindustry class. The actual storage location of data and structural information beyond what is described in this clause is out of scope of ISO/IEC 7816 [6]. The following structures are supported:

— Dedicated file (DF):
The DFs host applications and/or group files and/or store DOs. An application DF is a DF hosting an application. A DF may be the parent of other structures, whose types shall belong to the following set {DF, EF, DO}. These other structures are said to be immediately under the DF.

— Elementary file (EF):
The EFs store data. An EF may be the parent of other structures, whose types shall belong to the following set {DO, Record, DataString}. These other structures are said to be immediately under the EF. Two categories of EFs are specified.

  — An internal EF stores data interpreted by the card, i.e. data used by the card for management and control purposes.

  — A working EF stores data not interpreted by the card, i.e. data used by the outside world exclusively.

— Record:
The records store data. A record may be the parent of other structures, whose types shall belong to the following set {DO}. These other structures are said to be immediately under the record.

— DataString:
DataStrings store data. A DataString is a sequence of bytes in a transparent EF. A DataString may be the parent of other structures, whose types shall belong to the following set {DO}. These other structures are said to be immediately under the DataString.

— Data object (DO):
The DOs store data. A DO may be the parent of other structures, whose types shall belong to the following set {DO}. These other structures are said to be immediately under the DO.

Two types of logical organization are provided.

— Figure 3 illustrates a hierarchy of DFs with its corresponding security architecture (see clause 9). In such a card organization, the DF at the root is called the master file (MF); any DF may be an application DF, with or without its own hierarchy of DFs.

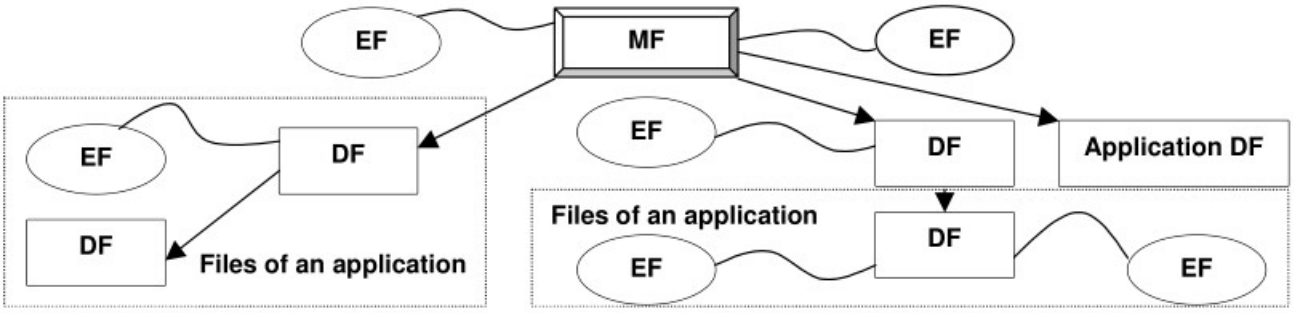


Figure 3 — Example of hierarchy of DFs

— Figure 4 illustrates application DFs in parallel, with no MF seen at the interface, i.e. without any apparent hierarchy of DFs. Such an organization supports independent applications in the card where any application DF may have its own hierarchy of DFs with its corresponding security architecture.

Figure 4 — Example of independent application DFs

7.2 Validity area

7.2.1 Definitions and attributes

The validity area (VA) on a logical channel is the result of all successful selections performed on that logical channel. It generalizes the concept of current file defined in the previous version of ISO/IEC 7816-4. The VA determines the resolution of DO tags and file identifiers. The VA consists of:

— curAppDF: a reference either to a DF or an application DF. It is always set.

— curDF: a reference to a DF, which may be an application DF. It is always set.

— curEF: a reference to an EF belonging to the current DF. It is not always set.

— curFile: a reference to a file, it is always set. Its value shall be identical to curDF if curEF is not set, and identical to curEF if curEF is set.

— curRecord: a reference to a record, belonging to a current EF structured in records. It is not always set.

— curDataString: a reference to a sequence of bytes being part of a transparent current EF. It is not always set.

— curConstructedDO: it shall be set for DO handling. When set, it is a reference to a constructed DO. It is not always set.

— curPrimitiveDO: a reference to a primitive DO whose parent is referenced by curConstructedDO. It is not always set.

— curDO: It shall be set for DO handling. When set, it is a reference to a DO, its value shall be identical to curConstructedDO if curPrimitiveDO is not set, and identical to curPrimitiveDO if curPrimitiveDO is set.

NOTE 1 curFile can be calculated from curDF and curEF and vice versa. Thus either curFile is dispensable or curDF and curEF are dispensable. The reason for redundancy is that some functions are easier to describe with curFile (e.g. ACTIVATE, DEACTIVATE) and other functions are easier to describe with curDF and curEF.

NOTE 2 curDO can be calculated from curConstructedDO and curPrimitiveDO and vice versa. Thus either curDO is dispensable or curConstructedDO and curPrimitiveDO are dispensable. The reason for redundancy is that some functions are easier to describe with curDO and other functions are easier to describe with curConstructedDO and curPrimitiveDO.

NOTE 3 The current SE does not belong to the current VA.

7.2.2 Basic rules for VA handling and use

The following list contains a non exhaustive list of rules describing the use of the VA. More rules are given in clauses containing command descriptions:

a) Enabling a physical interface (see 5.1), which opens the basic logical channel, sets the values of curAppDF and curDF to identical values, i.e. they reference either the MF or the implicitly selected application.

b) Resetting a logical channel resets the VA to the same value set when opening that logical channel. See 11.1.2 for other consequences of resetting.

c) Opening a logical channel sets the values of curAppDF and curDF to identical values (see 11.1.1 and 11.1.2).

d) Selecting an application DF sets curDF and curAppDF so that they reference the selected application DF.

e) Selecting a DF which is not an application DF sets curDF so that it references the selected DF. It sets or confirms curAppDF so that curAppDF references either the nearest application DF (if any) in the ancestors (parent, grandparent…) of the selected DF, or the MF if no application DF exists in the ancestors.

f) Selecting an EF sets curEF so that it references the selected EF, and curDF so that it references the parent of the selected EF (see rule e)). When EF selection occurs as a side-effect of a C-RP using referencing by short EF identifier, curEF may change, while curDF does not change.

g) Selecting a record sets curRecord so that it references the selected record as current record, and curEF so that it references the parent of the selected record (see rule f)).

h) Selecting a DataString within a transparent EF sets curDataString so that it references the selected DataString and curEF so that it references the parent of the selected DataString (see rule f)).

i) Selecting a structure containing DOs sets the curConstructedDO so that it references the selected DO according to the specified structure. If the selected structure is a constructed DO, it sets the curConstructedDO so that it references the selected DO. If the specified structure is a {Application DF, DF, EF, record or DataString}, structure selection sets the curConstructedDO so that it references a virtual root DO'7F70' as current Constructed DO. This DO shall be associated with the last encountered structure supporting DOs in the list above.

j) Selecting a primitive DO sets curPrimitiveDO so that it references the specified DO as current PrimitiveDO, and curConstructedDO so that it references the parent of the selected primitive DO.

k) For DO handling, the current template is the value of the DO referenced by curConstructedDO (see rule i).

An explicit selection may modify elements of the VA beyond the explicit selection due to recursion. Such implicit selections shall have the same outcome as an explicit selection.

EXAMPLE The selection of a constructed DO in a record will set curConstructedDO (rule i) explicitly. It will implicitly set or confirm curRecord (rule g)), curEF (rule f)), curDF (rule e)) and curAppDF (rule d)).

7.3 Structure selection

7.3.1 Structure selection methods

Selecting a structure allows access to its data and to structures below, if any. Structures may be selected implicitly, i.e. automatically (see 7.2.2, rules a), g)) after enabling a physical interface (see 5.1). When a structure cannot be implicitly selected, it shall be selected explicitly, i.e. by at least one of the following four methods.

Selection by DF name — A DF name may reference any DF. It is a string of up to sixteen bytes. Any application identifier (AID, see 12.2.3) may be used as DF name. In order to select unambiguously by DF name, e.g. when selecting by means of application identifiers, each DF name shall be unique within a given card.

Selection by file identifier — A file identifier may reference any file. It consists of two bytes. The value '3F00' is reserved for referencing the MF. The value 'FFFF' is reserved (see 11.4.1.2). The value '3FFF' is reserved (see below and 11.4.1). The value '0000' is reserved (see 11.2.2 and 11.4.1). In order to unambiguously select any file by its identifier, all EFs and DFs immediately under a given DF shall have different file identifiers.

Selection by path — A path may reference any file. It is a concatenation of file identifiers. The path begins with the identifier of a DF (the MF for an absolute path or the current DF for a relative path) 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, then the value '3FFF' (reserved value) can be used at the beginning of the path. The values '3F002F00' and '3F002F01' are reserved (see 12.2.1 and 12.2.2). The path allows an unambiguous selection of any file from the MF or from the current DF (see 12.3).

Selection by short EF identifier — A short EF identifier may reference any EF. It consists of five bits not all equal, i.e. any number from one to thirty. When used as short EF identifier, the number zero, i.e. 00000 in

binary, references the current EF. At MF level, the number thirty, i.e. 11110 in binary, is reserved (see 12.2.1). Short EF identifiers cannot be used in a path or as an EF identifier (e.g. in a SELECT command). All short EF identifiers of EF immediately under a given DF plus all short EF identifiers indicated in FCP DO'A2' associated with this DF shall be unique.

If supported, selection by short EF identifier shall be indicated.

— If the first software function table (see Table 117) is present in the historical bytes (see 12.1.1) or in EF.ATR/INFO (see 12.2.2), then the indication is valid at card level.

— If a short EF identifier DO'88' (see 10.2.3.1 and Table 10) is present in the CPs of an EF, then the indication is valid at EF level.

Selection by tag — A DO may be selected by its sole tag, without further information on the VA, if and only if it belongs to the base template of the current template i.e. the value of the current constructed DO.

Selection by record number — If the EF pointed to by curEF supports records then a record number references a specific record in that EF. A record number is a positive integer.

Selection by offset — If the EF pointed to by curEF is transparent, then an offset references the beginning of a sequence of bytes in that EF. An offset is an integer ≥ 0.

7.3.2 File reference data element and DO

This interindustry DO'51'(see Table 7) references a file. It may have any length.

— An empty DO references the MF.

— If the length is one and if bits b8 to b4 of the data element are not all equal and if bits b3 to b1 are set to 000, then bits b8 to b4 encode a number from one to thirty that is a short EF identifier.

— If the length is two, then the data element is a file identifier.

— If the length is more than two, then the data element is a path.

  — If the length is even and if the first two bytes are set to '3F00', then the path is absolute. The data element is a concatenation of at least two file identifiers starting with the MF identifier.

  — If the length is even and if the first two bytes are not set to '3F00', then the path is relative. The data element is a concatenation of at least two file identifiers starting with the identifier of the current DF.

  — If the length is odd, then the path is qualified. The data element is either an absolute path without '3F00', or a relative path without the identifier of the current DF, followed by a byte to use as P1 in one or more SELECT commands (see 11.1.1 and 12.3).

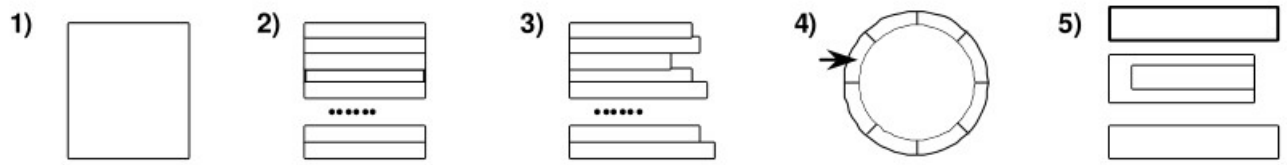
Table 7 — Coding of the file reference DO'51'

Tag	Length	Value
'51'	0	The empty data object references the MF
	1	Short EF identifier (bits b8 to b4 encode a number from one to thirty; bits b3 to b1 are set to 000)
	2	File identifier
	Even, > 2	Absolute path (the first two bytes are set to '3F00')
		Relative path (the first two bytes are not set to '3F00')
	Odd, > 2	Qualified path (the last byte shall be used as P1 in one or more SELECT commands)

7.3.3 General reference data element and DO

This interindustry DO'60' (see Table 86) and DO'7F72' (see Table 37) can reference any structure in a card. The value of DO'60' is used in the command data field of many CR-Ps handling DOs (see 11.4.2). In some cases, those commands temporarily transiently impact the VA; in other cases, they modify the VA.

7.3.4 Data referencing methods in elementary files



1) Transparent structure
2) Linear structure with records of fixed size
3) Linear structure with records of variable size
4) Cyclic structure with records of fixed size (the arrow references the most recently written record)
5) SIMPLE-TLV or BER-TLV structure

Figure 5 — EF structures

Data referencing method is an EF-dependent feature. An EF shall support at least one of the following structures:

— **Transparent structure —** The EF is seen at the interface as a single continuous, numbered sequence of data units accessible by commands for handling data units (see 11.2). Data unit size is an EF-dependent feature.

— **Record structure —** The EF is seen at the interface as a single continuous sequence of individually identifiable records accessible by commands for handling records (see 11.3). Record numbering method, SIMPLE-TLV or BER-TLV structure of the records are EF-dependent features. Three attributes are defined: record size, record organization and record LCS.

  — The size of the records is either fixed, or variable.

  — The organization of the records is either a sequence (linear structure), or a ring (cyclic structure).

  — A record may have a record life cycle. When it does, the record LCS distinguishes at least the following states: ACTIVATED and DEACTIVATED. The coding of a record LCS is out of scope for this part of ISO/IEC 7816 [6]. Within a given EF, either all records have a record life cycle or no record has a record life cycle. The presence of record life cycle is indicated by the CP (see Table 10).

— **SIMPLE-TLV structure** — The EF is seen at the interface as a set of SIMPLE-TLV data objects accessible by commands for handling data objects (see 11.4). This structure precludes BER-TLV structure

— **BER-TLV structure** — The EF is seen at the interface as a set of DOs accessible by commands for handling DOs (see 11.4). This structure precludes SIMPLE-TLV structure.

In DFs or applications, data may be referenced as DOs (see 6.2), the same way as in EFs. If the EF, DF or application supports this referencing, its selection sets the curConstructedDO (see 7.2.2 and 8.2.1).

7.4 File and data control information

7.4.1 File control information retrieval

By definition, the file control information is the byte string available in response to the SELECT command (see 11.1.1); it may be present for any file, i.e. any DF and any EF.

— If the first byte is valued from '00' to 'BF', then the byte string shall be BER-TLV encoded. ISO/IEC JTC 1/SC 17 reserves for future use all the values in the range '00' to 'BF' that are not defined in this document.

— If the first byte is valued from 'C0' to 'FF', then the byte string is not encoded according to this document.

Table 8 shows three interindustry templates:

— The FCP template is a set of CP DOs eligible for files, i.e. logical, structural and security attributes as listed in Table 10 and defined hereafter. Within the FCP template, the context-specific class is reserved for FCPs; tags '85' and 'A5' reference proprietary information.

— The FMD template is a set of file management data, i.e. interindustry DOs such as an application identifier as defined in 12.2.3, an application label as defined in 12.2.4 and an application expiration date as defined in ISO/IEC 7816-6, possibly nested within an application template as defined in 12.2.4. Within the FMD template, tags '53' and '73' reference discretionary data.

— The FCI template is a set of file CPs and file management data.

Table 8 — Interindustry templates for file control information

Tag	Value
'62'	Set of file control parameters (FCP template)
'64'	Set of file management data (FMD template)
'6F'	Set of file control parameters and file management data (FCI template)

The three templates may be retrieved according to selection options of the SELECT command (see Table 62).

— If the FCI option is set, then the FCI tag is optional for introducing the template in the response data field.

— If the CP or FMD option is set, then the corresponding tag is mandatory for introducing the template.

Part of the control information of a DF may additionally be present in an EF under the control of an application and referenced by tag '87' in the file CPs. If present within such an EF, file control information shall be introduced by the appropriate tag, either a CP tag, or a FCI tag.

7.4.2 Data control information retrieval

By definition, the data control information is the byte string available in response to a SELECT command with P1='10' or P1='13' (see 11.1.1) or a SELECT DATA command if bit b3 of P2 is set to 1 (see 11.4.2); it may be present for any DO.

Table 9 — Interindustry templates for data control information

Tag	Value
'62'	Set of data control parameters (CP DOs, possibly including a DO'62')

DO'62' nests CP DOs.

— It may be present in the data control information of any structure (files or applications supporting a BER-TLV structure, constructed DO). It belongs and applies to the current template after structure selection.

— When present within the current base template (see 8.2.2) it applies to, it may be retrieved by GET DATA or GET NEXT DATA.

— When it is present in another template, it may be indirectly referenced by a tagged wrapper (see 8.4.8).

7.4.3 Control parameters

Table 10 lists CP DOs for files and data objects, all in the context-specific class. When a CP is present, the table states whether it occurs at most once (explicit indication), or may be repeated (no indication).

Table 10 — Control parameter data objects

Tag	Length	Value	Applies to
'80'	Var.	Number of data bytes in the file, excluding structural information	Any EF*
'81'	Var.	Number of data bytes in the file or DO, including structural information if any	File* or DO*
'82'	1	File descriptor byte (see 7.4.5 and Table 11)	File*
	2	File descriptor byte and data coding byte (see Table 118)	
	3 or 4	File descriptor byte, data coding byte and maximum record size on one or two bytes	EF* supporting records
	5 or 6	File descriptor byte, data coding byte, maximum record size on two bytes and number of records on one or two bytes (see NOTE)	
'83'	2	File identifier	File*
'84'	up to 16	DF name	DF
'85'	Var.	Proprietary information not encoded in BER-TLV	File
'86'	Var.	Security attribute in proprietary format	File
'87'	2	Identifier of an EF containing an extension of the file control information	DF*
'88'	0 or 1	Short EF identifier (see 7.4.4)	EF*
'8A'	1	Life cycle status (LCS, see 7.4.10 and Table 14)	File* or DO*
'8B'	Var.	Security attribute referencing the expanded format (see 9.3.3 and Table 38)	File*
'8C'	Var.	Security attribute in compact format, SE oriented (see Table 30)	File*
'8D'	2	Identifier of an EF containing security environment templates (see 0)	DF
'8E'	1	Logical channel security attribute (see 9.3.7 and Table 47)	File* or DO*
'8F'	1	Profile indicator (see Table 12)	EF supporting records*
'92'	1	Data descriptor byte (see 7.4.7)	DO* or EF* supporting BER-TLV structure
'96'	Var.	as defined in ISO/IEC 7816-11	see ISO/IEC 7816-11
'97'	Var.	DF list (see 7.4.8)	DF*
'98'	Var.	Instance number (binary coding)	DO*
'99'	Var.	Number of DOs in the current template after file or DO selection (binary coding, see 7.2.2 rule i)).	File* or DO*
'9A'	Var.	Number of EFs within a DF (binary coding)	DF*
'9B'	Var.	EF list (see 7.4.8)	DF*
'9C'	Var.	Security attribute in compact format, SPT oriented (see Table 30). NOTE: It is mainly intended for DO handling.	File* or DO*
'9D'	Var.	Tag of the DO(s) to which DO'62' applies.	DO
'A0'	Var.	Security attribute template for DOs (see 9.3.5)	File* or DO
'A1'	Var.	Security attribute template in proprietary format	File
'A2'	Var.	Template consisting of one or more pairs of DOs: Short EF identifier (DO'88') - File reference (DO'51', L > 2, see 7.3.2)	DF
'A3'	Var.	Interface and LCS dependent security attribute template (see 7.4.12)	File or DO
'A5'	Var.	Proprietary information encoded in BER-TLV	File
'A6'	Var.	as defined in ISO/IEC 7816-11	see ISO/IEC 7816-11
'AB'	Var.	Security attribute template in expanded format (see 9.3.3)	File*
'AC'	Var.	Cryptographic mechanism identifier template (see 9.2)	DF
'AD'	Var.	Security parameters template (see 9.3.6.1)	DO
'AF'	Var.	Template encapsulating one or more DO'06' (OID) relevant to the application.	DF*

* indicates that the DO shall appear at most once under DO'62'.
In the response to a SELECT and under tags '62' and '6F', ISO/IEC JTC 1/SC 17 reserves any other DO of the context-specific class.

7.4.4 Short EF identifier

The following rules apply for the use of DO'88' in the CPs of any EF.

— If the card supports selection by short EF identifiers (see 7.3.1) and if DO'88' is absent, then in the second byte of the file identifier (tag '83'), bits b5 to b1 encode the short EF identifier.

— If DO'88' is present with a length set to zero, then the EF supports no short identifier.

— If DO'88' is present with a length set to one and if bits b8 to b4 of the data element are not all equal and if bits b3 to b1 are set to 000, then bits b8 to b4 encode the short EF identifier (a number from one to thirty).

7.4.5 File descriptor byte

DO'82' may be present in the CPs of any file (see Table 10).

— The first byte of the value is the file descriptor byte (see Table 11).

— If the value consists of two or more bytes, then the second byte is the data coding byte (see

— Table 118). If the card provides data coding bytes in several places, then the indication valid for a given file is in the closest position to that file within the path to that file from:

1) The MF, if present.
2) The DF referred to by curAppDF, if the MF is absent.
3) In the absence of indication in the path, the data unit size is set to the default value (see
4) Table 118).

Table 11 — Coding of the 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	-	1	1	1	0	0	0	**DF**
0	-	Not all set to 1			-	-	-	**EF category**
0	-	0	0	0	-	-	-	— EF for storing data not interpreted by the card (Working EF)
0	-	0	0	1	-	-	-	— EF for storing data interpreted by the card (Internal EF)
0	-	Any other value			-	-	-	— Proprietary categories of EFs
0	-							**EF structure**
0	-	Not all set to 1			0	0	0	— No information given
0	-	Not all set to 1			0	0	1	— Transparent structure
0	-	Not all set to 1			0	1	0	— Linear structure, fixed size, no further information
0	-	Not all set to 1			0	1	1	— Linear structure, fixed size, TLV structure
0	-	Not all set to 1			1	0	0	— Linear structure, variable size, no further information
0	-	Not all set to 1			1	0	1	— Linear structure, variable size, TLV structure
0	-	Not all set to 1			1	1	0	— Cyclic structure, fixed size, no further information
0	-	Not all set to 1			1	1	1	— Cyclic structure, fixed size, TLV structure
0	-	1	1	1	0	0	1	— BER-TLV structure
0	-	1	1	1	0	1	0	— SIMPLE-TLV structure

— Any other value is RFU.
— "Shareable" means that the file supports at least concurrent access on different logical channels.

7.4.6 Profile indicator

The following rules apply for the use of DO'8F' in the file CPs of any EF supporting records:

— If DO'8F' is present, then the value field contains a profile indicator according to Table 12;

— If DO'8F' is absent, then all records in the EF are implicitly in the unchangeable state ACTIVATED.

Table 12 — Coding of the profile indicator

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	x	x	x	x	x	x	x	**Coding of profile indicator defined by ISO/IEC JTC 1/SC 17**
0	–	–	–	–	–	–	x	Record LCS
0	–	–	–	–	–	–	0	All records in this file are in the unchangeable state ACTIVATED
0	–	–	–	–	–	–	1	Each record in this EF has its own record LCS, changeable by the commands ACTIVATE RECORD, DEACTIVATE RECORD and ACTIVATE FILE
1	x	x	x	x	x	x	x	**Proprietary coding of profile indicator**
— Any other value is RFU.								

7.4.7 Data descriptor byte

Referenced by tag '92', this interindustry data element is a CP DO, which contains information relevant to DO handling (see Table 13). The information is valid for all instances (if more than 1) of a DO in the relevant template. If a DO is declared "non shareable", only one instance of this DO is accessible on one logical channel.

Table 13 — Coding of the data descriptor byte

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	x	-	-	-	-	-	-	**Data accessibility**
0	0	-	-	-	-	-	-	— Not shareable DO
0	1	-	-	-	-	-	-	— Shareable DO
0	-	x	x	-	-	-	-	**structuring of the DOs in the template**
0	-	0	0	-	-	-	-	— No information given (default).
0	-	0	1	-	-	-	-	— Linear management (see 11.4.7)
0	-	1	0	-	-	-	-	— Cyclic management of instances (see 11.4.7)
0	-	1	1	-	-	-	-	— RFU
0	-	-		x	-	-	-	**Instances**
0	-	-	-	0	-	-	-	— A tag shall occur at most once in the template.
0	-	-	-	1	-	-	-	— Tags may occur more than once in the template (default).
0	-	-	-	-	x	x	x	**DO properties**
0	-	-	-	-	0	0	0	— No information given
0	-	-	-	-	-	-	1	— Tag list present (Constructed DO) or present in the tag list (Primitive DO)
0	-	-	-	-	-	1	-	— Base template extendable by wrappers (Constructed DO) or part of the template extension (Primitive DO)
0	-	-	-	-	1	1	-	— Automatic resolution of wrappers
— Any other value is RFU. — "Shareable" means that the DO supports at least concurrent access on different logical channels.								

7.4.8 DF and EF list data elements

Referenced by tag '97', the DF list interindustry data element is a file CP which shows the DFs contained in a DF or application DF. It is a concatenation of 2-byte file identifiers.

Referenced by tag '9B', the EF list interindustry data element is a file CP, which contains the identifiers of the elementary files contained in the DF or application DF, and their short EF identifier. The information of each file is coded on three consecutive bytes. The first two bytes contain the file identifier of an EF. The third byte is either the short EF identifier of the EF coded according to 7.4.4 or a byte coded '00' if that elementary file has no short EF identifier.

7.4.9 Instance number data element

Referenced by tag '98', this interindustry data element is a CP DO, which may be present only if there are several instances of the DO within its template. Its value shall be a binary coded positive sequence number, as defined by ISO/IEC 8825-1 for the INTEGER type. The numbering scheme for cyclic management of instances is the same as for the management of record numbers (see 11.3.6). See 11.4.7 for further information of numbering schemes.

7.4.10 Life cycle status

The card, files and other objects, each have a life cycle; the life cycle status (LCS) allows the card and the interface device to identify the different logical security states of the use of the card, files and other objects in the card.

To support flexible management of the life cycle of a file, record or DO as an attribute (see ISO/IEC 7816-9), this clause defines four primary states of the life cycle in the following order.

1) Creation state
2) Initialisation state
3) Operational state: activated or deactivated.
4) Termination state

LCS (1 byte) shall be interpreted according to Table 14.

— The values '00' to '0F' are interindustry.

— The values '10' to 'FF' are proprietary.

The default state shall be Operational state (activated).

Table 14 — Coding of the file or data LCS byte

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	0	0	0	No information given
0	0	0	0	0	0	0	1	Creation state
0	0	0	0	0	0	1	1	Initialisation state
0	0	0	0	0	1	-	1	Operational state (activated)
0	0	0	0	0	1	-	0	Operational state (deactivated)
0	0	0	0	1	1	-	-	Termination state
Not all zero				x	x	x	x	Proprietary
— Any other value is RFU.								

Under tag '8A', an LCS may be present in the CPs of any file or DO (see Table 10). A card LCS may be present in the historical bytes (see 12.1.1.11). Under tag '48', a card LCS may be present in EF.ATR/INFO (see 12.2.2). When it has an MF, the card is in, at least, the creation state.

NOTE Unless otherwise specified, the security attributes are valid for the operational state of files and ACTIVATED state of DOs.

7.4.11 Indirect referencing by short EF identifier using DO'A2'

A DO'88' under DO'A2' in the FCP of a DF indicates that the nested short EF identifier is valid when the DF is current. A command using it shall transiently select the EF referenced by the path which is nested in DO'51' after that DO'88'. Its success has no impact on curDF and curAppDF (see 7.2.2 rule f)). The impact on curEF is described in sections dealing with commands using short EF identifier (see e.g. 11.2.2 or 11.3.2).

7.4.12 Interface and life cycle status dependent security attribute template

DO'A3' may be present in the CPs of any file or DO and in such templates it may occur more than once (see Table 10).

DO'A3' shall contain at most one DO'91' (see 7.4.12.1). If DO'91' is absent then the security attributes contained in DO'A3' apply to all interfaces. If DO'A3' contains a DO'91' it shall be the first DO in the template.

DO'A3' shall contain at most one DO'8A' (see 7.4.12.2). If DO'8A' is absent then the security attributes contained in DO'A3' apply to all life cycle status. If DO'A3' contains a DO'8A' then it shall be

— the first DO in the template if DO'91' is absent, or

— the second DO in the template if DO'91' is present.

DO'A3' may contain any number of DO'8B', DO'8C', DO'9C', DO'A0', DO'AB' in any order having the same meaning and coding as defined by Table 10.

7.4.12.1 Transport type descriptor

Referenced by tag '91' under DO'A3', this data element indicates for which interfaces the security attributes contained in the same DO'A3' apply.

Table 15 — Coding of transport type descriptor (DO'91' under DO'A3')

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	-	-	-	-	-	-	-	0, other values are RFU [a]
-	x	x	-	-	-	-	-	00, other values are RFU
-	-	-	1	-	-	-	-	Communication by contact C6 (see ISO/IEC 7816-3)
-	-	-	-	1	-	-	-	Near-field communication (see ISO/IEC 18092 [22])
-	-	-	-	-	1	-	-	USB interface (see ISO/IEC 7816-12)
-	-	-	-	-	-	1	-	Contactless interface for proximity cards (see ISO/IEC 14443 [18])
-	-	-	-	-	-	-	1	Contact interface (see ISO/IEC 7816-3)

[a] This edition defines the case where one byte is sufficient to reference relevant interfaces. In the future, more than one byte may be necessary.

If in this data element a bit is set to

— 1 then for the corresponding interface the security attributes shall apply.

— 0 then for the corresponding interface the DO'A3' is irrelevant.

7.4.12.2 Indication of life cycle status

Referenced by tag '8A' under DO'A3', this data element indicates for which life cycle status the security attributes contained in the same DO'A3' apply. This data element shall contain one or more bytes and each byte shall be coded and interpreted according to 7.4.10 and Table 14, i.e. each byte encodes a life cycle status and all bytes of the data element build a set of life cycle status. If the life cycle of the file or DO, being given within a DO'8A' in the control parameter template (i.e. the template with tag '62')

— is element of this set then the security attributes shall apply for the file or DO.

— is not element of this set then the DO'A3' is irrelevant for the file or DO.

8 Specific use of DOs and related concepts

8.1 BER-TLV payloads and padding

A BER-TLV-encoded data field or payload is a concatenation of BER-TLV DOs, with possible padding before or after any of those DOs.

There are two standard ways of stating BER-TLV-encoding of a command or response data field or payload:

— CLA indicates secure messaging (see clause 10).

— Bit b1 of INS is set to 1 (odd INS code).

Commands with an even INS code may specify BER-TLV-encoding of data fields and payloads.

Within a command payload, this document may define the order of DOs (see e.g. Table 87). In a response payload, the order of DOs is either not defined, or is inherited from the order of the DOs present in the card.

Because of the coding of BER-TLV length fields (see E.2) a DO with a tag of n bytes cannot have a total length of e.g. (n+1+128) bytes or of (n+2+256) bytes or of (n+3+65536) bytes etc. The situation of invalid total length may also occur if several DOs are to be transmitted and the last DO creates the same problem. In all these cases if receiving a command with N_e set to such an invalid total length the card may send within a response data field containing DO(s) the first N_e bytes requested by the command and conclude with a SW1-SW2 set to '61XY', where 'XY' codes the number of remaining bytes. Afterwards response chaining (see 5.3.4) applies.

8.1.1 Padding conditions

The possibility of padding in a data field depends on the command. The terminology "concatenation of DOs" precludes padding. Padding is allowed:

— when handling data structures through data units; e.g. padding may be present in a READ BINARY response data field due to erasure or modification of DOs within an EF supporting data units.

— when handling data structures through records; e.g. padding may be present in a READ RECORD response data field due to the fixed format of records within an EF supporting records.

— when explicitly allowed by a standard C-RP (CLA<'80').

— when explicitly allowed by a specification of a C-RP (CLA>'7F').

If BER-TLV-encoding is explicitly stated, the data payload(s) shall be BER-TLV encoded, without padding. BER-TLV-encoding shall not apply to the individual data fields (a.k.a. payload fragments) of the commands into which the payload is fragmented, because payload fragmentation is entitled to split DOs. Padding is still precluded in payload fragments.

NOTE Secure messaging may require BER-TLV-encoding of payload fragments (see 10.1).

8.1.2 Padding procedure

When padding is allowed in BER-TLV payloads, bytes set to '00' may be present before, between or after DOs.

When present in the historical bytes (see 12.1.1) or in EF.ATR/INFO (see 12.2.2) or in the control information of any file (see tag '82' in Table 10), the data coding byte (see Table 118) indicates whether the value 'FF' is

— valid for the first byte of long tag fields of the private class, constructed encoding (explicit statement), or

— invalid for the first byte of tag fields (default value), i.e. used for the same purpose (padding) and under the same conditions as the value '00'.

8.2 Current template and data object generations

8.2.1 Current template and current DO

If set, the curConstructedDO references a constructed DO. Its value field, as defined in 7.2.2 rule k), is called the current template.

Any DO within the current template can be directly referenced by its sole tag in a command for interchange which does not provide any information on the VA together with this tag.

If any of those DOs is also constructed, the current template does not include any nested DOs within this constructed DO.

The virtual root DO'7F70' belongs to the 0 generation, and is the only DO in that generation. If curConstructedDO references an n^{th} generation constructed DO'XY', the current template is a sequence of $(n+1)^{th}$ generation DOs nested in DO'XY'. The current template does not include DOs possibly nested in any $(n+1)^{th}$generation DO. Annex F shows examples.

The curDO (see 7.2.1 dash 9) is set by a SELECT command (see 11.1.1) or a SELECT DATA command (see 11.4.2). DO selection also occurs as a side effect of other DO-handling commands, and is described in the relevant clauses, e.g. 11.4.3 and 11.4.4 for the GET DATA command.

In order to support referencing methods defined in this edition of ISO/IEC 7816-4, the card should be able to resolve tag '7F70'.

Immediately after enabling of an interface (see 5.1), thus before any selection, the current template depends on the implicitly selected application or DF (curAppDF in the VA). It may contain specific DOs to support e.g. discovery mechanisms defined in ISO/IEC 24727-2 [24].

Many commands which handle DOs imply one or several internal selections. Those selections are said to be "transient", and set a "transient current file" or a "transient current template", thus a "transient VA". In some cases, a side effect of the command success may validate those internal selections, thus set the transient VA as current VA.

8.2.2 Template extension

The template extension is that part of a template which is accessible only by automatic resolutions of tagged wrappers, whereby extending the base template (see Annex G or examples). All wrapper DO'63' are part of the base template.

If automatic resolution of tagged wrappers is not supported, the template extension is empty. The indirect references are only informative. If needed, the outside world may perform a two-stage procedure:

— selection of the structure where the DO is stored, when this structure is different from the current structure;

— then handling of the DO within the selected structure.

8.2.3 Data object sub-tree

A constructed DO has a tree structure. A sub-tree of this tree structure is obtained by removing nested DOs from the tree. An extended header associated with a constructed DO refers a sub-tree of the DO (see 8.4.6 and 8.4.7).

EXAMPLE A given n^{th} generation DO is denoted {T-L- {T1-L1-{T2-L2-V2}-{T3-L3-V3}}-{T4-L4-V4}-{T5-L5-V5}}. Removing the $(n+2)^{th}$ generation DO T2 and the $(n+1)^{th}$ generation DO T4 yields the sub-tree: {T-L'- {T1-L1'-{T3-L3-V3}}-{T5-L5-V5}}. L and L1 are replaced by L' and L1' because removing a DO from a template impacts its length.

8.2.4 Data object life cycle

Any DO stored in the card may have a DO life cycle. When it does, the LCS shall distinguish at least two states: ACTIVATED and DEACTIVATED. The coding of a DO's LCS and the coding of a file's LCS are analogous (see Table 14). The presence of DO's LCS is indicated in the CP template DO'62' (see Table 10).

8.3 Identification of data elements and data objects

8.3.1 Principles

At the interface between the card and the interface device, a data element is generally presented in the value field of a DO. The data element is said to be "under" the tag of the DO.

If the number of bits representing a data element is not a multiple of eight, one may use the BITSTRING universal DO, as e.g. in ISO/IEC 7816-15. If not, the mapping into a byte or a string of bytes should be specified together with the respective data element. Unless otherwise specified, the appropriate number of bits shall be set to 1 in the last byte starting from bit b1.

For purposes of retrieval and referencing in interchange, a data element shall be associated with the tag of a BER-TLV data object and the data element may be encapsulated in this data object.

A data element may be referenced directly by its associated BER-TLV tag. It may be associated with another data element that sets the context to which it belongs.

One or more command-to-perform data objects may indirectly reference a data element.

The interindustry commands support multiple instances of the same DO within a template, thus within an application and within a card.

A standard or an application may limit or prohibit multiple instances of the same DO within a template, within a structure or within the application.

8.3.2 Tag interpretation in command and response data fields or payloads

DOs of the universal class (first byte from '01' to '3F', see Annex E) always have their general meaning.

The meaning of DOs of the application class (first byte from '40' to '7F', see Annex E) is defined by standards. The default is the ISO/IEC 7816 [6] and the ISO/IEC JTC 1/SC 17 standards (see 8.3.3). The meanings of those DOs are always the same, though their function is defined by the command where they are used.

The way to interpret DOs of the context-specific class (first byte from '80' to 'BF', see Annex E) is defined:

— generally by its nesting within a constructed DO defined by the standard, e.g. DOs nested in DO'62' (see Table 10). This rule is recursive, and applies e.g. to DOs nested in a DO'A0' nested in a DO'62'.

— by CLA: secure messaging indication (see 5.4 and 10.1)

— by INS: commands may define such DOs of the context-specific class for a specific use.

— In other cases, the interpretation of those DOs is defined by the application.

8.3.3 Tag allocation

This document specifies many interindustry DOs, and allocates tags for the BER-TLV-encoding of those DOs. In addition to defining further interindustry DOs, ISO/IEC 7816-6 maintains an exhaustive list of the tuples (name of the interindustry DO, allocated tag) specified in ISO/IEC 7816 [6] at the time of publication. The tags allocated in ISO/IEC JTC 1/SC 17 standards belong:

— either to the application class;

— or to the context-specific class, when the corresponding DOs may be nested in a constructed DO the tag of which has been allocated by the standard.

When a tag is defined by another standard, the national body responsible for the standard shall seek advice of ISO/IEC JTC 1/SC 17/WG 4 for the tag or tags needed, and shall in due time comment in order to add this or those tags in ISO/IEC 7816-6.

Illustrated by Annex A, the subsequent clauses specify tag allocation schemes for identifying interindustry DOs in data fields. When needed, those tag allocation schemes use the interindustry DOs shown in Table 16 for notifying an authority responsible for tag allocation.

Table 16 — Interindustry data objects for tag allocation authority

Tag	Value
'06'	Object identifier (OID), encoding specified in ISO/IEC 8825-1), see examples in Annex A.
'41'	Country code (encoding specified in ISO 3166-1 [3]) and optional national data
'42'	Issuer identification number (encoding and registration specified in ISO/IEC 7812-1 [5]) and optional issuer data
'4F'	Application identifier (AID, encoding specified in 12.2.3)
'5F29'	Interchange profile.

8.3.4 Standard tag allocation scheme

In the application class, ISO/IEC JTC1/SC17 allocates or reserves the use of any application class tag numbers in the range 0-127 or greater than 511 (decimal).

Application class tag numbers in the range 128 to 511 (decimal) may be allocated by any standard or specification referenced by an OID, or by an application referenced by an AID.

In order to identify the tag allocation authority in this range, a constructed DO'78' shall be used. This DO shall nest an empty DO'80' followed by either a DO'06' or a DO'4F'. If DO'78' is present

— in the initial data string (see 12.1.2) or in EF.ATR/INFO (see 12.2.2), then the authority shall be valid for the entire card, except if overruled by another DO'78' in a file or a constructed DO.

— in the management data of an EF or DF, possibly supporting an application (see 7.4), or in the template made current by the selection of any file, then the authority shall be valid within that file or application, except if it is overruled by another DO'78' in a constructed DO.

— in a template, it shall be valid for all DOs in the template, and all objects contained in the constructed DOs, (thus irrespective of their generation), except if it is overruled by another DO'78' in a template.

8.3.5 Compatible tag allocation scheme

These tag allocation schemes use interindustry DOs and further DOs.

These further DOs shall be nested within interindustry DOs referenced by tags '70' to '72', or tags '74' to '77'. In those DOs, the meaning of the application class tags is not defined in ISO/IEC 7816 [6] except for tags '41', '42' and '4F' for identifying tag allocation authorities (see Table 16).

When further DOs use tags of the context-specific class, and belong to the interindustry templates defined above, their meaning is defined by the tag allocation authority.

The use of the context-specific class within interindustry templates with tags '65' (cardholder-related data), '66' (card data), '67' (authentication data) and '6E' (application-related data) is deprecated.

In order to identify a compatible tag allocation scheme and the authority responsible for the scheme, the interindustry DO'78' may be used. Possible occurrence and overruling are the same as in 8.3.4. If present, its value shall contain one of the interindustry DOs shown in Table 16, for identifying a tag allocation authority.

8.3.6 Coexistent tag allocation scheme

In such a scheme, all the interindustry DOs shall be nested within interindustry DOs'7E'. Moreover, tags '79' and '7E' shall not be given another interpretation, as well as tags '62', '64', '6F' (CP, FMD and FCI templates, see 7.4) and '7D' (SM template, see 10.1).

These tag allocation schemes may use tags with an interpretation not defined in ISO/IEC 7816 [6]. In order to identify a coexistent tag allocation scheme and the authority responsible for the scheme, an interindustry DO'79' shall be used. This DO shall nest one of the interindustry DOs shown in Table 16.

— If an authority is valid for the entire card, then DO'79' shall be present in the initial data string (see 12.1.2) or in EF.ATR/INFO (see 12.2.2).

— If an authority is valid within a file or application, then DO'79' shall be present in the EF, DF or application DF management data (see 7.4), or in the current template set by any file selection.

8.3.7 Avoidance of independent tag allocation schemes

An independent tag allocation scheme uses tags with another interpretation than ISO/IEC 7816 [6], but which does not comply with 8.3.6. Such tag allocation schemes do not comply with this document and do not allow interchange.

Besides a consistent use of compatible and coexistent allocation schemes, there are three ways for an application to avoid such a situation, and remain compliant with this document:

— The use of interindustry discretionary DOs'53' to present discretionary DOs, and DOs'73' to nest proprietary DOs in discretionary templates.

— The use of 3-byte tags reserved for this purpose (see 8.3.4).

— The use of DOs of the context-specific class (see 8.3.2).

8.4 Referencing and retrieval of DOs and data elements

8.4.1 General

A tag 'XY' directly references a data element, which is the value of DO'XY'. For retrieval of DOs before selection of an application see 12.4.

Control parameters and file management data may be retrieved in the answer to a SELECT or SELECT DATA command (see 11.1.1, 11.4.2.1 and Table 87).

Once an application is selected, any DO should be retrieved directly or indirectly:

— In the file management data (see 7.4) of the application DF and from specific EFs within the current DF.

— In the current template (see 8.2) after application selection, using GET DATA or GET NEXT DATA commands (see 11.4.3 and 11.4.4). File management data and/or DO'62' (nesting CPs) should be included in this template, to ensure that those particular DOs are relevant to the application.

Element lists, tag lists, header lists, extended headers and extended header lists, are interindustry DOs which reference indirectly data elements, thus reference DOs within any template. Such a data element instructs the card how to interpret a command data field or to construct a response data field. Computing the concatenation of data elements or the DOs from the indirect reference is called resolving (or resolution of) the indirect reference.

The interpretation of a tag list, a header list, an extended header or an extended header list depends on the template in which they are defined. Nesting one of those DOs within a wrapper (which defines this template), allows referencing wherever in a card. An optional tag within the wrapper allows referencing the result of the resolution of the indirect reference by this tag.

The command data field syntax in the SELECT DATA, GET DATA, GET NEXT DATA shows the use of those DOs, which may be close to the syntax of a wrapper.

8.4.2 Element list

Under tag '5F41', this interindustry data element denotes that the information to retrieve is not presented as DOs, but under application control. It shall be used only within the wrapper template. Its structure and the returned information are out of scope of ISO/IEC 7816 [6].

8.4.3 Tag list

Referenced by tag '5C', this interindustry data element is a concatenation of tag fields without delimitation. It references a concatenation of DOs with the respective tags, in the same order as in the tag list.

8.4.4 Header list

Under tag '5D', this interindustry data element is a concatenation of tuples (tag field T, length field L) without delimitation. The byte string is as defined as in the tag list except for possible truncation of the values. When L = '00', no truncation occurs. When L > '00', the value shall be right truncated to L bytes, except when it is already shorter or equal to L bytes. The coding of L is a BER length.

WARNING — Within a header list, all referenced DOs to be truncated should be primitive, because a truncated constructed DO is not a DO. For extraction of parts of a constructed DO, one should use an extended header.

8.4.5 Extended header and extended header list

Under tags '4D', '5F60' or '5F61' (see 8.4.8 for the differences between different tags), this interindustry data element is an extended header list.

An extended header list is

— either an extended header

— or a concatenation of extended headers.

An extended header is a concatenation of tuples (tag field T, length field L) without delimitation. An extended header references information within a target DO. A complete extended header shall be derived by the following procedure:

a) primitive DO not to be referenced:
If the extended header is to be tagged by
   1) '4D', delete the tag, length and value field.
   2) '5F60' or '5F61', delete the value field and replace the length field by '00'.

b) primitive DO to be referenced without truncation:
Delete the value field. If the extended header is to be tagged by
   1) '4D', replace the length field by '00'.
   2) '5F60' or '5F61', replace the length field by '80'.

c) primitive DO to be referenced with truncation:
Delete the value field and replace the length field by a truncation indication (see 8.4.6)

d) constructed DO not to be referenced at all:
Delete the value field and replace the length field by '00'.

e) constructed DO to be referenced entirely:
Delete the value field and replace the length field by '80'.

f) constructed DO where part of the information is to be referenced:
Adjust the value of the length field according to the outcome of applying the procedures above.

As shown in F.2, the explicit referencing of DOs where nothing is referenced is not always necessary, even when several instances of such a DO exist in a given template. Removing useless referencing is entitled by this standard.

8.4.6 Resolving an extended header

To resolve an extended header, one shall build the referenced byte string as follows.

— If a tag indicates a primitive encoding, then the pair of tag field and length field is replaced by data referenced by the tag. If the extended header is tagged by '4D', a length of '00' means that the complete DO/element is included in the byte string. If the extended header is tagged by '5F60' or '5F61', a length of '80' means that the complete DO/element is included in the byte string. A length which is neither '00' in an extended header tagged by '4D', nor '80' in an extended header tagged by '5F60' or '5F61', indicates the maximum number of data bytes to be retrieved and consequently may require truncation as defined in 8.4.4. If the truncation indication indicates more bytes than available in the DO, the behaviour depends on the tagging:

  — Under tag '4D', it is out of the scope of this document.

  — Under tag '5F60' and '5F61' the extended header is invalid. If used in a command APDU, the command shall be rejected by SW1-SW2 = '6985'.

— A tag indicating a constructed encoding followed by a non-zero length, except '80', introduces a subsequent value field that is an extended header list. A tag indicating a constructed encoding followed by a zero length is ignored. A tag indicating a constructed encoding followed by '80' means that the complete constructed DO/complete template is included in the byte string.

— The card shall ignore the elements of the extended header that do not match the target structure.

For security reasons, a card may reject a command with SW1-SW2 = '6985' because of the lack of consistency between the structure used in the command data field and the contents of the card. This may override security attributes.

The byte string consists of either

— the value fields of the primitive DOs, possibly truncated according to the indicated lengths (Case 1), or

— the primitive DOs, possibly truncated according to the indicated lengths, and nested in the respective template, the length of which complies with the BER-TLV rules (Case 2).

— If present, the length '80' shall be replaced by the actual length. The complete constructed DO/complete template is included in the byte string.

The encoding of the referenced byte string, namely, a (possibly constructed) DO(s) or a concatenation of (possibly truncated) data elements, is indicated:

— by an appropriate INS code (odd/even)

— or by an appropriate parameter of the command, e.g., either an appropriate encoding of the data field (either constructed for those containing DOs or primitive for those containing data elements).

— or by tagging the extended header list by another tag than '4D', as shown in 8.4.8 and the PERFORM SECURITY OPERATION command (see ISO/IEC 7816-8).

8.4.7 Resolving an extended header list

The byte string referenced by an extended header list is the concatenation of the byte strings referenced by the extended headers, in the same order as in the extended header list. When parsing an extended header list:

— the resolution of one extended header shall be followed by the resolution of the next one if any.

— the order within the byte string is therefore defined by the order of extended headers in the extended header list, which should match the order within the target (template) in order to avoid the following problem: As in the resolution of an extended header, the card shall ignore the elements (extended headers) of the extended header list that do not match the target (template) structure.

8.4.8 Wrapper

Under tag '63', this interindustry DOs shall nest two or more DOs.

a) The first DO is a mandatory indirect reference. Only one indirect reference shall be given in a wrapper. It is a choice between:

   1) an element list (tag '5F41' or '53');
   2) an element list nesting DOs (tag '73');
   3) a tag list (tag '5C')
   4) a header list (tag '5D');
   5) an extended header list (tag '4D');
   6) an extended header list referencing a byte string with no stated structure (tag '5F60');
   7) an extended header list referencing one DO or a concatenation of DOs (tag '5F61');
   8) a byte string identical to the response data of the last command to perform (tag '80', empty DO), see below.

   possibly followed by a filter DO'7F71' (see 11.4.2.3) in the case of tag list or extended header list. The use of DO'5F60' or the use of DO'5F61' entitle skipping indication of primitive DOs according to 8.4.5.

b) The remainder of the wrapper value field shall be:

   1) either an application identifier DO'4F' (see 12.2.3). The indirect reference shall be valid within the current template after application selection.
   2) or a file reference DO'51' (see 7.3.2 and Table 7). The indirect reference shall be valid within the selected file; if it is empty, it shall be valid in the current template.
   3) or a DO'4F' followed by a non-empty DO'51' referencing a file which shall exist in the application. The indirect reference shall be valid within this file.
   4) or one DO'52' (command to perform). The indirect reference shall be valid within the response payload of this command.
   5) or several DO'52'. The commands to perform shall be processed in the presented order. The indirect reference shall be valid within the response payload of the last command or in the transient VA set by this command.

The resolution of a wrapper:

— starts with the resolution of the second part of the wrapper, which is either a standard structure, or a response APDU. It ends there if the indirect reference is a DO'80'.

— if the indirect reference is not a DO'80', it ends with the resolution of the indirect reference, which is defined in 8.4.3, 8.4.4, 8.4.5.

EXAMPLE The following wrapper DO nests a tag list and one command-to-perform.
{'63'-L-{'5C'-L-(Tag1-Tag2-Tag3)}-{'52'-L-Command APDU}}

8.4.9 Tagged wrapper

Under tag '63', this DO extends a template in a card which supports automatic resolution of wrappers.

The first DO in the value filed of a tagged wrapper DO'63' shall be an empty DO'XY'. The tag 'XY' shall be valid in GET DATA or GET NEXT DATA C-RPs when the extended template is the current template; 'XY' shall respect the current tag allocation scheme. The remainder of the value shall be the value of a wrapper DO (see 8.4.8). The outcome of resolving the indirect reference shall be the value of DO'XY' when addressed within the template it extends (see 8.2.2):

— If the wrapper references a DO'ZT', the value of DO'XY' shall be the value of this DO'ZT' (see Annex G for examples). The wrapper shall reference only one DO, possibly constructed.

— If the wrapper references a byte string, the value of DO'XY' shall be that byte string.

NOTE 1 Neither the notation 'XY' nor 'ZT' precludes the use of tags consisting of more than one byte.

NOTE 2 The automatic resolution of the tagged wrapper creates within the current template a virtual DO. The tag of this virtual DO is taken from the first DO in the tagged wrapper. The length field of this DO codes the length of the content of the resolved indirection. The value of the DO is the content itself (see examples in Annex G).

9 Security architecture

9.1 General

This clause describes security status, security attributes and security mechanisms.

Security status — The security status represents the current state possibly achieved on a logical channel after physical interface enabling (see 5.1) or resetting a logical channel and/or execution on this logical channel a single C-RP or a sequence of C-RPs possibly performing authentication procedures. The security status may also result from completion of a security procedure related to the identification of the involved entities, if any, e.g. by proving knowledge of a password (e.g. using a VERIFY command) or knowledge of a key (e.g. using a GET CHALLENGE command followed by an EXTERNAL AUTHENTICATE command, or using a sequence of GENERAL AUTHENTICATE commands), or by secure messaging (e.g. message authentication). Five security statuses are considered.

— **Global security status —** In a card using a hierarchy of DFs, it may be modified by completion of an MF-related authentication procedure (e.g. entity authentication by a password or a key attached to the MF).

— **Application-specific security status —** It may be modified by completion of an application-related authentication procedure (e.g. entity authentication by a password or a key attached to the application); it may be maintained, recovered or lost by application selection; this modification may be relevant only for the application to which the authentication procedure belongs. If logical channels apply, then the application-specific security status may depend on the logical channel.

— **File-specific security status —** It may be modified by 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; this modification may be relevant only for files within the application to which the authentication procedure belongs. If logical channels apply, then the file-specific security status may depend on the logical channel.

— **DO-specific security status —** It may be modified by completion of a DO-related authentication procedure; it may be maintained, recovered or lost by selections; this modification may be relevant only for DO within the application to which the authentication procedure belongs. If logical channels apply, then the DO-specific security status may depend on the logical channel.

— **Command-specific security status —** It only exists while processing a command using secure messaging and involving authentication; such a command may leave the other security status unchanged.

Security attributes — The security attributes, when they exist, define which actions are allowed, and under which conditions. The security attributes of a structure depends on:

— its category (DF, EF or DO);

— optional CPs in its CP template DO'62' and/or in that of its parent structure(s);

Security attributes may also be associated with commands, DOs and tables & views. In particular, security attributes may

— specify the security status of the card to be in force before accessing data;

— restrict access to data to certain functions (e.g. read only) if the card has a particular status;

— define which security functions shall be performed to obtain a specific security status.

Security mechanisms — This clause considers 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 authenticate has to prove the knowledge of the relevant secret or private key in an authentication procedure (e.g. a GET CHALLENGE command followed by an EXTERNAL AUTHENTICATE command, a sequence of GENERAL AUTHENTICATE commands).

— **Data authentication** — Using internal data, either a secret key or a public key, the card checks redundant data received from the outside world. Alternatively, using secret internal data, either a secret key or a private key, 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, either a secret key or a private key, the card deciphers a cryptogram received in a data field. Alternatively, using internal data, either a secret key or a public key, 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. 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 bytes received from or sent to the outside world. This mechanism may be used for protecting privacy and as a countermeasure against pattern recognition.

The result of an authentication may be logged internally according to application requirements.

9.2 Cryptographic mechanism identifier template

One or more cryptographic mechanism identifier DO'AC' may be present in the CPs of any DF (see Table 10). Each one explicitly indicates the meaning of a cryptographic mechanism reference in the DF and its hierarchy. The template shall consist of two or more DOs.

— The first shall be a cryptographic mechanism reference, DO'80' (see Table 55).

— The second DO shall be an object identifier, DO'06', as defined in ISO/IEC 8825-1. The identified object shall be a cryptographic mechanism specified or registered within a standard, e.g. an ISO standard. Examples of cryptographic mechanisms are encryption algorithms (e.g. ISO/IEC 18033 [21]), message authentication codes (e.g. ISO/IEC 9797 [9]), authentication protocols (e.g. ISO/IEC 9798 [10]), digital signatures (e.g. ISO/IEC 9796 [8] or ISO/IEC 14888 [19]), registered cryptographic algorithms (e.g. ISO/IEC 9979 [11]), and so on.

— If present, one or more subsequent DOs (DO'06' or DO'13') shall either identify a mechanism used by the previous mechanism (i.e. a mode of operation, e.g. ISO/IEC 10116 [13], or a hash-function, e.g. ISO/IEC 10118 [14]), or indicate parameters (tag dependent on the previous mechanism).

DO'13' gives a relative-OID rooted at the nearest preceding DO'06' as defined in ISO/IEC 8825-1.

EXAMPLE (see explanations in Annex A and Annex B)
{'AC' - '0B' - {'80'-'01'-'01'} - {'06'-'06'-'28818C710201'}}
The template associates the local reference '01' to the first encryption algorithm in ISO/IEC 18033-2 [21].

{'AC' - '11' - {'80'-'01'-'02'} - {'06'-'05'-'28CC460502'} - {'06'-'05'-'28CF060303'}}
The first object identifier refers to the second authentication mechanism in ISO/IEC 9798 [10]. The second object identifier refers to the third dedicated hash-function in ISO/IEC 18033-3 [21]. Therefore the template associates the local reference '02' to GQ2 using SHA-1.

9.3 Security attributes

This clause defines the coding of data relevant to the security attributes, and two formats for binding objects and security attributes: a compact format based on bitmaps and an expanded format that extends the compact format by TLV list management. The expanded format may use the codings defined for the compact format under different encapsulations.

9.3.1 Security attributes targets

Referenced by tags '86', '8B', '8C', '8E', '9C', 'A0', 'A1', 'A3', 'AB', security attributes may be present in the CPs of any file or DO (see Table 10). Any object in the card (e.g., command, file, DO, table & view) may be associated with more than one security attribute and/or with a reference contained in a security attribute.

In SCQL environment (see ISO/IEC 7816-7, commands for structured card query language), security attributes can be specified in SCQL operations, e.g. CREATE TABLE and CREATE VIEW commands. If security attributes based on this clause are used, then they shall be conveyed in a DO with tags '8B', '8C' or 'AB' in the security attribute parameters of an SCQL operation.

Security attributes for BER-TLV DOs are defined in 9.3.5. Security attributes for logical channels are defined in 9.3.7.

9.3.2 Compact format

In compact format, an access rule consists of an access mode field followed by one or more security condition bytes. The security attribute contains one or several concatenated access rules.

If several access rules are present in the value field of a DO'8C' or '9C' (see Table 10), they represent an OR condition.

Access mode field — An access mode field consists of either one or more bytes. If the access mode field consists of

— one AMB, then bit b7 to b1 indicates either the absence of a security condition byte when set to 0, or the presence of a security condition byte in the same order (bits b7 to b1) when set to 1. When bit b8 is set to 1, bits b7 to b4 may be used for additional commands, e.g. application-specific commands.

— more bytes, the first byte of the access mode field shall be set to '00'. The second and possibly further bytes (see NOTE below) in the access mode field are AMBs. Each AMB indicates in bit b8 whether or not it is the last AMB. Bit b8 shall be set to 0 in the last AMB. Bit b8 shall be set to 1 in all other AMBs. Bit b6 to b1 indicates either the absence of a security condition byte when set to 0, or the presence of a security condition byte in the same order (bits b6 to b1) when set to 1 and in the same order as the AMBs. When bit b7 is set to 1, bits b6 to b1 may be used for additional commands, e.g. application-specific commands.

NOTE Further AMBs are not defined in this edition of ISO/IEC 7816-4, and may be defined in future editions or other parts of this standard.

Table 17 to Table 29 define access mode bytes respectively for DFs, EFs, DOs and tables & views.

Table 17 — Coding of the sole byte in an access mode field for DFs

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	Bits b7 to b1 according to this table
1	-	-	-	-	-	-	-	Bits b3 to b1 according to this table (bits b7 to b4 proprietary)
0	1	-	-	-	-	-	-	DELETE FILE (self)
0	-	1	-	-	-	-	-	TERMINATE CARD USAGE (MF), TERMINATE DF
0	-	-	1	-	-	-	-	ACTIVATE FILE
0	-	-	-	1	-	-	-	DEACTIVATE FILE
-	-	-	-	-	1	-	-	CREATE FILE (DF creation)
-	-	-	-	-	-	1	-	CREATE FILE (EF creation)
-	-	-	-	-	-	-	1	DELETE FILE (child)

Table 18 — Coding of the sole byte in an access mode field for EFs

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	Bits b7 to b1 according to this table
1	-	-	-	-	-	-	-	Bits b3 to b1 according to this table (bits b7 to b4 proprietary)
0	1	-	-	-	-	-	-	DELETE FILE
0	-	1	-	-	-	-	-	TERMINATE EF
0	-	-	1	-	-	-	-	ACTIVATE FILE, ACTIVATE RECORD(S)
0	-	-	-	1	-	-	-	DEACTIVATE FILE, DEACTIVATE RECORD(S)
-	-	-	-	-	1	-	-	WRITE BINARY, WRITE RECORD, APPEND RECORD
-	-	-	-	-	-	1	-	UPDATE BINARY, UPDATE RECORD, ERASE BINARY, ERASE RECORD (S)
-	-	-	-	-	-	-	1	READ BINARY/RECORD (S), SEARCH BINARY/RECORD, COMPARE BINARY/RECORD

Table 19 — Coding of the sole byte in an access mode field for DOs

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	Bits b7 to b1 according to this table
1	-	-	-	-	-	-	-	Bits b3 to b1 according to this table (bits b7 to b4 proprietary)
0	1	-	-	-	-	-	-	DELETE DATA
0	-	1	-	-	-	-	-	MANAGE DATA Terminate
0	-	-	1	-	-	-	-	MANAGE DATA Activate
0	-	-	-	1	-	-	-	MANAGE DATA Deactivate
-	-	-	-	-	1	-	-	MANAGE SECURITY ENVIRONMENT
-	-	-	-	-	-	1	-	PUT DATA/PUT NEXT DATA/UPDATE DATA
-	-	-	-	-	-	-	1	GET DATA/GET NEXT DATA/COMPARE DATA

Table 20 — Coding of the sole byte in an access mode field for security objects

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	Bits b7 to b1 according to this table
1	-	-	-	-	-	-	-	Bits b3 to b1 according to this table (bits b7 to b4 proprietary)
0	x	-	-	-	-	-	-	0 (any other value is RFU)
0	-	1	-	-	-	-	-	*TERMINATE* [a]
0	-	-	1	-	-	-	-	*ACTIVATE* [a]
0	-	-	-	1	-	-	-	*DEACTIVATE* [a]
-	-	-	-	-	x	-	-	0 (any other value is RFU)
-	-	-	-	-	-	1	-	*PUT/UPDATE* [a]
-	-	-	-	-	-	-	1	*GET* [a]

[a] Description in *italic format* means an operation and not a command.

Table 21 — Coding of the sole byte in an access mode field for tables & views

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	Bits b7 to b1 according to this table
1	-	-	-	-	-	-	-	Bits b3 to b1 according to this table (bits b7 to b4 proprietary)
0	1	-	-	-	-	-	-	CREATE USER, DELETE USER
0	-	1	-	-	-	-	-	GRANT, REVOKE
0	-	-	1	-	-	-	-	CREATE TABLE, CREATE VIEW, CREATE DICTIONARY
0	-	-	-	1	-	-	-	DROP TABLE, DROP VIEW
-	-	-	-	-	1	-	-	INSERT
-	-	-	-	-	-	1	-	UPDATE, DELETE
-	-	-	-	-	-	-	1	FETCH

Table 22 — Coding of the 2nd byte in an access mode field (1st AMB) for DFs

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	Last byte of access mode field
1	-	-	-	-	-	-	-	Another byte follows in this access mode field
-	0	-	-	-	-	-	-	Bits b6 to b1 according to this document
-	1	-	-	-	-	-	-	Bits b6 to b1 proprietary
-	0	1	-	-	-	-	-	TERMINATE CARD USAGE (MF), TERMINATE DF
-	0	-	1	-	-	-	-	ACTIVATE FILE
-	0	-	-	1	-	-	-	DEACTIVATE FILE
-	0	-	-	-	1	-	-	CREATE FILE (DF creation)
-	0	-	-	-	-	1	-	CREATE FILE (EF creation)
-	0	-	-	-	-	-	1	DELETE FILE (child)

Table 23 — Coding of the 3rd byte in an access mode field (2nd AMB) for DFs

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	-	-	-	-	-	-	Same coding and meaning as in Table 22
-	0	1	-	-	-	-	-	DELETE FILE (self)
-	0	-	1	-	-	-	-	APPLICATION MANAGEMENT REQUEST
-	0	-	-	1	-	-	-	REMOVE APPLICATION
-	0	-	-	-	x	x	x	000 (any other value is RFU)

Table 24 — Coding of the 4th byte in an access mode field (3rd AMB) for DFs

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	-	-	-	-	-	-	Same coding and meaning as in Table 22
-	0	x	x	x	x	-	-	0000 (any other value is RFU)
-	0	-	-	-	-	1	-	command creating data objects (see ISO/IEC 7816-9)
-	0	-	-	-	-	-	1	DELETE DATA

Table 25 — Coding of the 2nd byte in an access mode field (1st AMB) for EFs

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	-	-	-	-	-	-	Same coding and meaning as in Table 22
-	0	1	-	-	-	-	-	TERMINATE EF
-	0	-	1	-	-	-	-	ACTIVATE FILE, ACTIVATE RECORD(S)
-	0	-	-	1	-	-	-	DEACTIVATE FILE, DEACTIVATE RECORD(S)
-	0	-	-	-	1	-	-	APPEND RECORD
-	0	-	-	-	-	1	-	UPDATE BINARY, UPDATE RECORD
-	0	-	-	-	-	-	1	READ BINARY/RECORD(S), SEARCH BINARY/RECORD

Table 26 — Coding of the 3rd byte in an access mode field (2nd AMB) for EFs

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	-	-	-	-	-	-	Same coding and meaning as in Table 22
-	0	1	-	-	-	-	-	DELETE EF
-	0	-	x	x	-	-	-	00 (any other value is RFU)
-	0	-	-	-	1	-	-	WRITE BINARY, WRITE RECORD
-	0	-	-	-	-	1	-	ERASE BINARY, ERASE RECORD(S)
-	0	-	-	-	-	-	1	COMPARE BINARY/RECORD

Table 27 — Coding of the 4th byte in an access mode field (3rd AMB) for EFs

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	-	-	-	-	-	-	Same coding and meaning as in Table 22
-	0	x	x	x	x	-	-	0000 (any other value is RFU)
-	0	-	-	-	-	1	-	command creating data objects (see ISO/IEC 7816-9)
-	0	-	-	-	-	-	1	DELETE DATA

Table 28 — Coding of the 2nd byte in an access mode field (1st AMB) for DOs

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	-	-	-	-	-	-	Same coding and meaning as in Table 22
-	0	1	-	-	-	-	-	MANAGE DATA Terminate
-	0	-	1	-	-	-	-	MANAGE DATA Activate
-	0	-	-	1	-	-	-	MANAGE DATA Deactivate
-	0	-	-	-	1	-	-	MANAGE SECURITY ENVIRONMENT
-	0	-	-	-	-	1	-	PUT DATA/UPDATE DATA
-	0	-	-	-	-	-	1	GET DATA/GET NEXT DATA

Table 29 — Coding of the 3rd byte in an access mode field (2nd AMB) for DOs

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	-	-	-	-	-	-	Same coding and meaning as in Table 22
-	0	1	-	-	-	-	-	DELETE DATA
-	0	-	1	-	-	-	-	command creating data objects (see ISO/IEC 7816-9)
-	0	-	-	x	x	-	-	00 (any other value is RFU)
-	0	-	-	-	-	1	-	PUT NEXT DATA
-	0	-	-	-	-	-	1	COMPARE DATA

Security condition byte — Each security condition byte specifies which security mechanisms are necessary to conform to the access rule. Table 30 shows the security condition byte. Bits b8 to b5 indicate the required security conditions. If not all equal, bits b4 to b1 identify:

— either a security environment (see 0, SEID from '01' to '0E').

— or a DO'AD' (see 9.3.6) by its number from '01' to '0E'. The DO'AD' is a CP under the same DO'62' as the security attribute.

Table 30 — Coding of the security condition byte

b8	b7	b6	b5	b4	b3	b2	b1	Meaning	
0	0	0	0	0	0	0	0	Always	
1	1	1	1	1	1	1	1	Never	
-	-	-	-	0	0	0	0	No reference to a security environment nor to a DO'AD'	
-	-	-	-	Not all equal				Under tag '8C' [1] or '9E' [2]	SEID (see 0)
								Under tag '9C' [1] or '9D' [2]	Sequence number of a SPT DO'AD' (see Table 38)
0	-	-	-	-	-	-	-	At least one condition	
1	-	-	-	-	-	-	-	All conditions	
-	1	-	-	-	-	-	-	Secure messaging	
-	-	1	-	-	-	-	-	External authentication	
-	-	-	1	-	-	-	-	User authentication (e.g. password)	

[1] for use in compact format (under tag '62', see Table 10). [2] for use in expanded format (see Table 33).

The mechanisms defined in the security environment, or in the DO'AD', shall be used according to the indications in bits b7 to b5 for command protection and/or external authentication and/or user authentication.

— If bit b8 is set to 1, then all the conditions set in bits b7 to b5 shall be satisfied.

— If bit b8 is set to 0, then at least one of the conditions set in bits b7 to b5 shall be satisfied.

— If bit b7 is set to 1, in a security condition which references a security environment, then the control reference template (see 0) of the security environment, describes whether SM shall apply to the command data field and/or to the response data field (see usage qualifier byte, Table 57). If the security condition indicates a DO'AD' number, the relevant SEID shall be available either in the DO'AD' or in a security attributes extension nested in the DO'AD'.

— If b8 = b7 = b6 = b5 =0, and if b4 b3 b2 b1 indicates a valid DO'AD' number, a boolean expression DO shall be present in the DO'AD' (see 9.3.6.11).

9.3.3 Expanded format

9.3.3.1 General

In expanded format, an access rule consists of an access mode DO followed by one or more security condition DOs. Access control to an object is managed by referencing access rules from the related object. A DO'AB' may be present in the CPs of any file (see Table 10) for such access rules.

Access mode DOs — An access mode DO contains either an access mode field (see Table 17 to Table 29 and Table 41 to Table 44), or a list of command descriptions or a proprietary state machine description; subsequent security condition DOs are relevant for all the indicated commands. Table 31 shows access mode DOs.

Table 31 — Coding of access mode DOs

Tag	Length	Value	Meaning
'80'	Var.	Access mode field	See Table 17 to Table 29 and Table 41 to Table 44
'81' to '8F'	Var.	Command header description	List of (part of) command headers (see Table 32)
'9C'	Var.		Proprietary state machine description

If the tag is from '81' to '8F', then the access mode data element represents a list of possible combinations of values of the four bytes CLA, INS, P1 and P2 in the command header. Depending on bits b4 to b1 of the tag, the list contains only values as described in Table 32. Several groups may appear in order to define a set of commands, e.g. values of INS P1 P2, INS P1 P2, … for tag '87'.

Table 32 — Coding of tags '81' to '8F' for access mode DOs

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	0	0	0	x	x	x	x	**The command description includes**
1	0	0	0	1	-	-	-	— (CLA), i.e. the value of CLA
1	0	0	0	-	1	-	-	— (INS), i.e. the value of INS
1	0	0	0	-	-	1	-	— (P1), i.e. the value of P1
1	0	0	0	-	-	-	1	— (P2), i.e. the value of P2
— The value of CLA shall encode zero as the number for a logical channel with the meaning that the description is independent from logical channels..								

Security condition data objects — According to Table 33, the security condition DOs define the security actions required for accessing an object protected through the particular access mode DO. If used as a security condition, a control reference template (see 0) referenced by tag 'A4' (AT), 'B4' (CCT), 'B6' (DST) or 'B8' (CT) shall contain a usage qualifier DO (see Table 57) indicating the security action.

Table 33 — Coding of security condition DOs

Tag	Length	Value	Meaning
'90'	0	-	Always
'97'	0	-	Never
'9D','9E'	1	Security condition byte	See Table 30
'A4'	Var.	Control reference template	External or user authentication depending on the usage qualifier
'AB'	Var.	Object status condition template	See 9.3.3.2
'B4', 'B6', 'B8'	Var.	Control reference template	SM in command and/or response depending on the usage qualifier
'A0'	Var.	Security condition DOs	At least one security condition shall be fulfilled (OR template)
'A7'	Var.	Security condition DOs	Inversion of the security condition (NOT template)
'AF'	Var.	Security condition DOs	Every security condition shall be fulfilled (AND template)
'BE'	Var.	Security condition template	Boolean. Refers to the values of items present in the card

Several security condition DOs may be attached to the same operation.

— If security condition DOs are nested in an OR template (tag 'A0'), then at least one security condition shall be fulfilled before acting.

— If security condition DOs are not nested in an OR template (tag 'A0') or if they are nested in an AND template (tag 'AF'), then every security condition shall be fulfilled before acting.

— The value of the NOT security condition (tag 'A7') is logical inversion of the value of the security condition it nests.

9.3.3.2 Object status condition template

The template provides the possibility to make a security condition depending to the:

— Current value of an object (EF, DO), referenced in the object status condition template

— Current value of an attribute related to the object (EF, DF, DO, key or password) referenced in the object status condition template

— Current value of OS internal data (e.g. system clock)

— Current value of (parts of) the command message

The template is composed by four different parts:

— A mandatory condition data object that defines the operation to be performed to compare the referenced value with the comparison data given in the template

— A mandatory object locator DO; the object locator identifies unambiguously the data to be used for comparison

— A mandatory comparison data to be compared with the value of the referenced object or the value of an attribute related to the referenced object.

— An optional binary filter to be applied to the data to be compared (logical AND) before comparison.

Table 34 shows an object status condition template.

Table 35 together with Table 36 shows a condition data object.

Table 34 — SC-DO for the description of access conditions depending on object attributes

Tag	Length	Value	Meaning
'AB'	Var.	condition DO'9E' object locator according to Table 37 (tag '7F72') binary filter DO'5F71' (optional) comparison data (tag '53')	Object status condition

Table 35 — Coding of the condition data object

Tag	Length	Value
'9E'	'01'	Rule to verify the object attribute (see Table 36)

Table 36 — Values for coding the conditions

Value	Condition fulfilled if
'01'	<*, the value of the referenced object/attribute is lower than the given
'02'	≤*, the value of the referenced object/attribute is lower than or equal to the given
'03'	=*, the value of the referenced object/attribute is equal to the given comparison data
'04'	≥*, the value of the referenced object/attribute is higher than or equal to the given comparison data
'05'	>*, the value of the referenced object/attribute is higher than the given comparison data
'06'	≠*, the value of the referenced object/attribute is not equal to the given comparison data
	** by equal length of the objects*

The security condition described by the object status condition template is not fulfilled if the referenced object/attribute value does not fulfil the condition defined by the condition DO.

9.3.3.3 Object locator

The object locator provides a reference to an object stored in the card. The following object types may be referenced:

— MF or DF specific key

— MF or DF specific password

— An application specific data object

— An EF or DF

— Special system data managed by the card

— A data element of a command message

The object locator is represented by a template '7F72'. The template is composed by an object reference DO (mandatory) followed by an attribute reference DO (conditional). The type of the object reference DO depends on the object to be referenced (EF, DF, DO, key, password). Table 37 lists the object reference DOs depending on the referenced object type.

A cryptographic reference template (CRT) according to 0 is used to refer to a key or a password. The CRT must contain at least a key reference data object (DO'83' or '84'). Further data objects from Table 51 may be used (e. g. usage qualifier) if required for further qualification of the reference. A mandatory DO (empty) following refers by its tag to the intended key attribute from Table 40.

An object locator according to Table 37 is used to refer either to

— a control parameter of a DF, an EF, or a DO

— (part of) the data content in a transparent EF

— (part of) the data content in a record of a structured EF

— the value field of a DO

If a reference to a DF is given a second data object must follow, indicating a control parameter object from Table 10. If an EF or DO is referenced the second DO is optional. If present it must indicate a control parameter of the EF or DO in the same way as for a DF. If absent the object locator refers to the data content of the EF (transparent EF), the data content of the record inside the EF or the value field of the DO.

To refer to OS internal system data the empty DO'80' shall be used. The following attribute reference DO indicates by its tag the relevant system information. The definition of the relevant tags is out of the scope of this standard.

To refer parts of the command message, the empty DO'81' shall be used. The following mandatory DO indicates which part of the command APDU is addressed. The DO'A1' is used to address either

— (a part of) the data field, defined by an offset and a length

— (a part of) the value field of a data object contained in the data field.

The intended data object in the data field is defined by the tag list. If no tag list is given, the command data field is in the focus scope of the object locator. Using the offset DO and/or the length DO the relevant area of the command data field/the DO value field can be restricted. If the offset DO is missing the offset value zero is implicitly used. If the length DO is missing the object locator refers to the complete range starting from the position indicated by the offset up to the end of the data field of the command APDU/the value field of the data object.

Table 37 — Coding of the object locator DO'7F72'

Object Type	Object reference DO	Attribute reference DO
Key	CRT containing at least a key reference DO ('83' or '84')	Empty DO (mandatory), tag refers to the relevant key attribute (see Table 40)
Password	CRT containing at least a password reference DO'83' and a usage qualifier DO'95'	Empty DO (mandatory), tag refers to the relevant key attribute (see Table 40)
DF	General reference template '60' (see Table 86) indicating a DF	Empty DO (mandatory), tag refers to the relevant control parameter object (see Table 10)
CP of an EF or DO	General reference template '60' (see Table 86) indicating an EF or DO	Empty DO (mandatory), tag refers to the relevant control parameter object (see Table 10)
Record, DataString or data element	General reference template '60' (see Table 86) indicating a record or DataString n an EF or the value filed of a data object (possibly located in an EF)	None
System data	DO'80' (empty) indicating the internal system data area	Data object (empty) that indicates the intended system data element (proprietary)
Command message	DO'81' (empty) indicating the command APDU	One of the following data objects (mandatory): '89': Refers to the command header (empty DO) '96': Refers to the N_e value of the command APDU (empty DO) 'A1': Command data DO; may contain following DOs in the given order: − tag list indicating a DO of the command data field (optional) − an offset DO'54' (optional) − a length DO'02' (optional) An empty DO'A1' refers to the complete data field of the command APDU

9.3.3.4 Binary Filter value (optional)

The binary filter DO'5F71' provides a binary mask to be used in a logical AND operation to the data to be compared before comparison.

9.3.3.5 Comparison data

The comparison data DO'53' provides the binary representation of the value to be compared with the data specified by the object locator.

9.3.4 Access rule references

Access rules in expanded format may be stored in an EF supporting a linear structure with records of variable size. Such an EF is named EF.ARR. One or more access rules may be stored in each record referenced by a record number. Such a record number is named ARR byte. Table 38 illustrates the layout of an EF.ARR.

Table 38 — EF.ARR layout

Record number (ARR byte)	**Record content** (one or more access rules)
1	Access mode DO, one or more security condition DOs, access mode DO, …
2	Access mode DO, one or more security condition DOs, …
…	…
N	Access mode DO, one or more security condition DOs, …

Security attributes DO'8B' referencing expanded format (see Table 38) may be present in the CPs of any file or DO (see Table 10).

— If the length is one, then the value field is an ARR byte that references a record in an implicitly known EF.ARR.

— If the length is three, then the value field is a file identifier followed by an ARR byte; the file identifier references EF.ARR and the ARR byte is the record number in EF.ARR.

— If the length is even and at least four, then the value field is a file identifier followed by one or more pairs of bytes. Each pair consists of an SEID followed by an ARR byte; the SEID identifies the security environment where the access rules referenced by the ARR byte apply.

Table 39 — Security attribute DOs referencing expanded format

Tag	Length	Value
'8B'	1	ARR byte (one byte)
	3	File identifier (two bytes) - ARR byte (one byte)
	Even, > 3	File identifier (two bytes) - SEID (one byte) - ARR byte (one byte) – (SEID byte - ARR byte) - …

The ARR byte of the current SE indicates the access rules valid for the current access to the application DF.

NOTE If no SE is set in a former MANAGE SECURITY ENVIRONMENT command, then the default SE is the current SE.

9.3.5 Security attributes for data objects

The granularity of security-related properties is the DO, identified by its tag within the current template associated with the VA. Any DO may have control parameters seen at the card interface as DO'62' nesting security attributes for that DO. When a DO does not feature control parameters, the security attributes of the parent template will be inherited.

When it belongs to a template extension, the security attributes of a DO shall not be inherited from parent of the tagged wrapper. Its security attributes are those defined where the indirect reference points to.

Under tag 'A0', a security attribute template for DOs may be present in the CPs of any file or DO (see Table 10). Such a template:

— starts with a security attribute DO (tags '86', '8B', '8C', '8E', '9C', 'A0', 'A1', 'AB');

— ends with a possibly empty tag list DO'5C'.

The tag list should contain only tags belonging to the base template to which DO'62' nesting DO'A0' belongs, or tags of DOs expected to be added to this template. When the tag list is empty, the security attribute template is valid for the whole base template to which the DO'62' belongs.

9.3.6 Security parameters template

9.3.6.1 General properties

A security parameters template, DO'AD' under DO'62', provides an extension of security attributes associated with a security object (e.g. a key, password…) or a DO which endorses security handling (see 9.3.6.2). Any DO'AD' which supports this function shall contain either a wrapper or a tagged wrapper DO'63' (see 8.4.9 and 9.3.6.2) giving an indirect reference to the security object or DO. It may give data usage parameters describing the properties and conditions of use of this DO.

A DO'AD' nested in a DO'AD' has the same coding as a DO'AD' nested in a DO'62'.

As any template, the security parameters template may nest a DO'62'. Within this DO, the security attributes for DOs have the meaning defined in 9.3.5. All other security attributes apply to the security object (see Table 20 for the coding of the access mode byte).

EXAMPLE Within a DO'62' under a DO'AD', the following DOs may be both present:
— A DO {A0 06 {8CXYZT}{5C01B3}} has its normal meaning, i.e. the security attributes for access to the DO'B3' which contains DOs specified by an OID.
— A DO {8CXYZT} or {ABXY…} codes the security attributes for access to the DO nesting the security object (see Table 20).

A DO'AD' may also be referenced by any command using this DO or the security object to which the DO'AD' is attached.

If a Boolean expression is to be evaluated as TRUE to allow the execution of a C-RP, a DO defining this Boolean expression shall be present in a DO'AD' (see 9.3.6.11).

9.3.6.2 Extensions of the security attribute

All DOs with tags from 'A0' to 'A5' identify the type of the security object. Each type has a specific Access Mode Byte (seeTable 41 to Table 44). The security attribute for Diffie-Hellman keys is irrelevant, so DO'A5' is always empty.

Any DO with tag from 'A0' to 'A4' shall nest:

— Either a DO'8C' nesting a security attribute in compact format (see 9.3.2), referencing SEs.
— Or a DO'9C' nesting a security attribute in compact format (see 9.3.2), referencing SPTs.
— Or a DO'AB' nesting a security attribute in expanded format (see 9.3.3), where the security condition byte DO'9D' may be used to reference SPTs.

Table 40 — Data usage parameters in a security parameters template (tag 'AD' under tag '62')

Tag	Length	Value		Occurrence
'06'	Var.	OID of a document describing e.g. an algorithm and/or the use of DO'B3'		at most once
'63'	Var.	Wrapper		at most once
'7B'	Var.	security environment DO (see Table 59)		once
'80'	1	Sequence number (mandatory)		once
'82'	Var.	Security object(password/reference data type) number (binary coding)		at most once
'83'	Var.	Security object (key/certificate type) number (binary coding)		at most once
'84'	1	SEID		any
'86'	1	key usage constraints indicator (see 9.3.6.4)		at most once
'8A'	1	LCS byte of the security object		at most once
'90'	Var.	Maximum number of tries of an authentication procedure (binary coding)	volatile	at most once
91			non volatile	
'92'	Var.	Remaining number of tries of an authentication procedure (binary coding)	volatile	at most once
'93'			non volatile	
'94'	Var.	Maximum usage counter (binary coding)	volatile	at most once
'95			non volatile	
'96'	Var.	Remaining usage counter (binary coding)	volatile	at most once
'97			non volatile	
'98'	Var.	Maximum number of signature creations (binary coding)	volatile	at most once
'99			non volatile	
'9A'	Var.	Remaining number of signature creations (binary coding)	volatile	at most once
'9B			non volatile	
'9C'	1	Non-repudiation flag (non endorsed if value = '00')		at most once
'9D'	Var.	Maximum number of key generations (binary coding)		at most once
'9E'	Var.	Remaining number of key generations (binary coding)		at most once
'A0'	Var.	Security attribute extension for authentication objects		at most one choice among those
'A1'	Var.	Security attribute extension for private keys		
'A2'	Var.	Security attribute extension for public keys		
'A3'	Var.	Security attribute extension for certificates		
'A4'	Var.	Security attribute extension for secret keys		
'A5'	0	Security attribute extension for private Diffie-Hellman keys[a]		
'AD'	Var.	Security parameters template		any
'AF'	Var.	Password properties template (see 9.3.6.12)		any
'B3'	Var.	DOs specified by the DO'06'		any
'BE'	Var.	Boolean expression		at most once

— Other tags in the context-specific class are RFU.

— The value of a non volatile counter is not related to physical channel enabling and disabling.

— The value of a volatile counter is impacted by physical channel enabling and disabling.

[a] The use of such a key is not subject to usage conditions, because it belongs to public domain parameters.

When a DO'AD' is referenced in a DO'9C' or a DO'9D', this DO'AD' or a tagged wrapper referencing it shall belong to the same template as the security attribute extension.

The access mode byte in the security attribute extension indicates which functions or commands are supported by the security object. When it is not used in a DO'AD', the access mode byte may also indicate which operations can be performed on the security object.

If no SEID is provided in the security attribute referencing a DO'AD', this extension of the security attribute may provide an SEID if necessary.

Table 41 — Coding of the access mode byte for authentication objects

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	Bits b7 to b1 according to this table
1	-	-	-	-	-	-	-	Bits b3 to b1 according to this table (bits b7 to b4 proprietary)
0	1	-	-	-	-	-	-	ENABLE VERIFICATION REQUIREMENT
0	-	1	-	-	-	-	-	DISABLE VERIFICATION REQUIREMENT
0	-	-	1	-	-	-	-	Proprietary use
0	-	-	-	1	-	-	-	Proprietary use
-	-	-	-	-	1	-	-	RESET RETRY COUNTER
-	-	-	-	-	-	1	-	CHANGE REFERENCE DATA
-	-	-	-	-	-	-	1	VERIFY

Table 42 — Coding of the access mode byte for private asymmetric keys

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	Bits b7 to b1 according to this table
1	-	-	-	-	-	-	-	Bit b5 proprietary, all other bits according to this table
-	1	-	-	-	-	-	-	PSO Compute digital signature
-	-	1	-	-	-	-	-	PSO Decipher
0	-	-	x	-	-	-	-	0 (any other value is RFU)
-	-	-	-	1	-	-	-	GENERAL AUTHENTICATE
-	-	-	-	-	1	-	-	INTERNAL AUTHENTICATE
-	-	-	-	-	-	1	-	GENERATE ASYMMETRIC KEY PAIR
-	-	-	-	-	-	-	1	proprietary when used in the SPT, GET ATTRIBUTE when used elsewhere

Table 43 — Coding of the access mode byte for public asymmetric keys and certificates

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	Bits b7 to b1 according to this table
1	-	-	-	-	-	-	-	Bit b5 proprietary, all other bits according to this table
-	1	-	-	-	-	-	-	PSO Verify digital signature, Verify certificate
-	-	1	-	-	-	-	-	PSO Encipher
0	-	-	x	-	-	-	-	0 (any other value is RFU)
-	-	-	-	1	-	-	-	GENERAL AUTHENTICATE
-	-	-	-	-	1	-	-	EXTERNAL AUTHENTICATE
-	-	-	-	-	-	1	-	GENERATE ASYMMETRIC KEY PAIR
-	-	-	-	-	-	-	1	proprietary when used in the SPT, GET ATTRIBUTE when used elsewhere

Table 44 — Coding of the access mode byte for secret keys

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	-	-	-	-	-	-	-	Bits b7 to b1 according to this table
1	-	-	-	-	-	-	-	Bit b4 proprietary, all other bits according to this table
-	1	-	-	-	-	-	-	PSO encipher, compute cryptographic checksum
-	-	1	-	-	-	-	-	PSO decipher, verify cryptographic checksum
-	-	-	1	-	-	-	-	MUTUAL AUTHENTICATE/GENERAL AUTHENTICATE
0	-	-	-	x	-	-	-	0 (any other value is RFU)
-	-	-	-	-	1	-	-	EXTERNAL AUTHENTICATE/MUTUAL AUTHENTICATE/EXTERNAL AUTHENTICATE for Role[a]
-	-	-	-	-	-	1	-	INTERNAL AUTHENTICATE
-	-	-	-	-	-	-	1	proprietary when used in the SPT, GET ATTRIBUTE when used elsewhere

[a] more information about 'EXTERNAL AUTHENTICATE for Role' can be found in EN 14890-2 [2] clauses 7.3 and 7.4

9.3.6.3 Sequence number

The successive instances of a DO'AD' shall correspond to increasing values of the sequence number. The value of DO'80' is coded in binary, on one byte. The DOs'AD' referenced in a security condition byte (see Table 30), shall have a number in the range '01' to '0E'.

9.3.6.4 Key usage constraints indicator

The data element in DO'86' indicates certain constraints to the usage of a key (see Table 45). Within this clause the following definitions are used:

— Preparation phase: the preparation phase consists of a C-RP or a sequence of C-RPs. The nature of commands sent in the preparation phase is out of scope of this document.

— Usage phase: the usage phase consists of a C-RP or a sequence of C-RPs. The nature of commands sent in the usage phase is out of scope of this document.

Table 45 — Coding of the key usage constraints

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	At most one usage
-	1	-	-	-	-	-	-	Immediate usage
-	-	1	-	-	-	-	-	Details are provided in the OID reference
-	-	-	x	x	x	x	x	00000 (any other value is RFU)

If bit b8 is set to

— 1 this indicates that for each key usage the preparation phase shall be performed.

— 0 this indicates that the key is usable more than once per preparation phase.

If bit b7 is set to

— 1 this indicates that the interface device is not allowed to send a C-RP between the preparation phase and the usage phase on the logical channel used for key usage.

— 0 this indicates that the interface device may send one or several C-RP(s) between the preparation phase and the usage phase, on arbitrary logical channels.

If bit b6 is set to

— 1 then the SPT containing the DO'86' shall contain a DO'06' and the OID reference shall provide details for key usage constraints.

— 0 then the SPT containing the DO'86' may or may not contain a DO'06'.

Bits b5 to b1 of the key usage constraints indicator are RFU.

EXAMPLE 1 For signature keys granting non repudiation the preparation phase typically consists of user authentication (e.g. a VERIFY command) and the usage phase typically contains e.g. zero, one or more PSO HASH commands followed by e.g. a PSO COMPUTE DIGITAL SIGNATURE COMMAND.

EXAMPLE 2 For keys used in authentication procedures proving the authenticity of an external entity the preparation phase typically contains the retrieval of a challenge from the card (e.g. by means of a GET CHALLENGE command) and the usage phase typically contains an authentication procedure (e.g. EXTERNAL AUTHENTICATE command).

9.3.6.5 Number of tries

When the security object is used for authentication, the number of tries is limited to the value of DO'90' or '91' in the data usage parameters nested in DO'AD'. The number of allowed remaining tries is given by the value of DO'92' or '93'. The values of DO'90', '91, '92' and '93' are coded in binary.

9.3.6.6 Number of uses

The number of uses of the security object is limited to the value of DO'94' or '95' in the data usage parameters nested in DO'AD'. The number of allowed remaining uses is given by the value of DO'96' or '97'. The values of DO'94', '95', 96' and '97' are coded in binary.

9.3.6.7 Non-repudiation

When granting non-repudiation, a function can rely on a security object DO with this flag set in DO'9C', i.e. to a value > 0. If this value is '00', the DO ('9C 01 00') cannot endorse non-repudiation.

9.3.6.8 Security object numbering

If needed, security objects (reference data/passwords or keys/certificates) may have a number contained in DO'82' (respectively '83'). The values of DO'82' and '83' are coded in binary, on one or several byte(s). The first byte of the value may be used as object reference in commands handling basic security (see 11.5 and Table 94). The value field may also be used when referencing the security data object by a CRT (see DO'83', '84' in Table 55).

NOTE The constrains from Table 94 limits the range of possible values for the first byte of the value field if that byte is intended to be used as object reference in a command handling basic security.

9.3.6.9 Signature creation

If needed, the number of signature creations with the security object is limited to the value of DO'98' or '99' in the data usage parameters nested in DO'AD'. The number of allowed remaining creations is given by the value of DO'9A' or '9B'. The values of DO'98', '99', '9A' and '9B' are coded in binary

9.3.6.10 Key generation

If needed, the number of key generations with the security object is limited to the value of DO'9D' in the data usage parameters nested in DO'AD'. The number of allowed remaining creations is given by the value of DO'9E'. The values of DO'9D' and '9E' are coded in binary.

9.3.6.11 Boolean expression

The boolean expression DO'BE' under DO'AD' contains:

— A mandatory object identifier DO'06', which defines how to use the DOs which follow to build the Boolean expression.

— One or several DOs of the context-specific class, referencing data or DOs with the same syntax as the general reference DO'60' (see 11.4.2.1) or object locator DO'7F72' (see 9.3.3.3).

NOTE In 9.3.2, the last dash contains a requirement for the presence of a DO'BE'.

9.3.6.12 Password properties template

The password properties template DO'AF' under DO'AD' contains password properties. A DO'06' may be present in the password properties template.

If the password template

— contains an OID in a DO'06' that DO'06' shall be the first DO in the password template. The standard or specification referenced by this OID shall specify further data objects enclosed in the password properties template.

— does not contain a DO'06' then the template is in conformance to ISO/IEC 7816-15 and may contain any DO from Table 46 in any order with the meaning and coding specified in Table 46.

Table 46 — Password properties template (DO'AF' under tag 'AD')

Tag	Length	Description
'81'	1	Verification data history length: the maximum number of verification data values that shall be recorded and maintained. When the verification data is set, it shall be different from all values in the verification data history. The value of verification data history length shall be in the closed interval [0, 8].
'82'	1	transport format for verification data and new reference data — '00', direct: data transferred to the ICC is taken as it is (binary coding) — '01', numeric: each character is encoded according to ISO/IEC 10646 [16] (UTF-8) and shall be element of the closed interval ['30', '39']. Data transferred to the ICC shall padded with 'FF' until maximum length characters are available. — '03', alphanumeric, i.e. digit, upper and lower case ASCII: each character is encoded according to ISO/IEC 10646 [16] (UTF-8) and shall be element of the closed interval ['30', '39'] or ['41', '5A'] or ['61', '7A']. Data transferred to the ICC shall padded with 'FF' until maximum length characters are available. — other values are RFU
'83'	1	Minimum length: minimum number of characters in verification data
'84'	1	Maximum length: maximum number of characters in verification data
- other tags in the context specific class are RFU		

9.3.7 Security attributes for logical channels

A logical channel security attribute DO'8E' (at most one) may be present in the CPs of any file or DO (see Table 10) and in any appropriate security environment (SE, see 0). It shall be interpreted according to Table 47, where:

— "Not shareable" means that at most one logical channel shall be available. The physical technology of the logical channel may be limited.

— "Secured" means that SM keys (see clause 10) shall be available (e.g. established by a previous authentication).

— "User authenticated" means that the user shall be authenticated (e.g. a successful password verification).

Table 47 — Coding of the logical channel security attribute

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	-	-	1	Not shareable
0	0	0	0	0	-	1	-	Secured
0	0	0	0	0	1	-	-	User authenticated
— Any other value is RFU.								

9.4 Security support data elements

This clause specifies a collection of security support data elements with rules governing the way their values are handled. The security support data elements extend and refine the control reference DOs. The card may provide them as generic support to security mechanisms performed by an application. Applications may reference them for secure messaging and for security operations (see ISO/IEC 7816-8). This clause specifies neither some characteristics of the security support data elements, e.g. their lengths, nor the algorithms that alter their value.

Table 48 — Security support DOs

Tag	Value
'7A'	Set of security support DOs with the following tags
'80'	Card session counter
'81'	Card session identifier
'82' to '8E'	File selection counter
'93'	Digital signature counter
'9F2X'	Internal progression value ('X' is a specific index, e.g. an index referencing a counter of file selections)
'9F3Y'	External progression value ('Y' is a specific index, e.g. an index referencing an external time stamp)
— Under tag '7A', ISO/IEC JTC 1/SC 17 reserves any other DO of the context-specific class.	

Principles — The card shall maintain and use the value of security support data elements as follows.

— Update is done with new values either computed by the card, or provided by the outside world, in accordance with the specific rule for a specific type of security support data element.

— Update is performed before any output is produced for the command causing an update. The update is independent of the completion status of the command. If the value is to be used by the application in an operation that causes an update, the update is performed before the value is used.

— Access to application-specific security support data elements is restricted to functions performed by the specific application.

NOTE The actual security achieved in a C-RP ultimately depends on the algorithms and protocols specified by the application; the card only provides support with these data elements and associated usage rules.

Data elements — The card may support C-RP security with data elements called progression values. Increased at specific events throughout the life of the card, these values are different each time the card is activated. Two progression values are specified: a card session counter and a session identifier.

— The card session counter is incremented once during card activation.

— The session identifier is computed from the card session counter and from data provided by the outside world.

Two types of progression values are specified.

— Internal progression values, if so specified for an application, register the number of times specific events are performed. The data element shall be incremented after the event; the card may provide a reset function for these counters which if so specified for an application sets its value to zero. Internal

progression values cannot be controlled by the outside world and are suitable for use as secured in-card approximate representations of real time. Their values can be used in cryptographic computations.

— External progression values, if so specified for an application, shall only be updated by a data value from the outside world. The new value shall be numerically larger than the current value stored in the card.

References — The card may provide access to the value of security support data elements as follows.

— An EF may be present in the MF, e.g. for a card session counter, or in an application DF, e.g. for application-specific progression values.

— Auxiliary DOs, with tags '88', '92' and '93' (see Table 54) may be present in a control reference template. These DOs can be used if the SE supports unambiguous use of these data elements.

— Within the interindustry DO'7A', the context-specific class is reserved for security support DOs as listed in Table 48.

10 Secure messaging

Secure messaging (SM) protects all or part of a C-RP, or a concatenation of consecutive data fields (payload fragmentation, see 5.3), by ensuring two basic security functions: data confidentiality and data authentication. Secure messaging is achieved by applying one or more security mechanisms. Possibly explicitly identified by a cryptographic mechanism identifier template (see 9.2) in the CPs of any DF (see 7.4), each security mechanism involves a cryptographic algorithm, a mode of operation, a key, an argument (input data) and often, initial data.

— The transmission and reception of data fields may be interleaved with the processing 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 cryptographic algorithm with different modes of operation. The hereafter specified padding rules do not preclude such a feature.

10.1 SM fields and SM DOs

10.1.1 SM protection of command payloads

An SM payload is an SM command or response SM field, irrespective of its length. If protection of the payload by secure messaging takes place before possible fragmentation, an oversize SM payload is a particular case of oversize BER-TLV coded payload, to which the rules given in 5.3 apply.

The format of the payload shall be as defined below (see 10.4), except that its length is not limited by the length formats and constraints which mandated chaining (see 5.3).

10.1.2 SM protection of chained commands and responses

If there is a need to protect chained SM command or response data fields individually, fragmentation according to 5.3 shall take place before C-RP protection. Each of the C-RPs shall then be protected (see 10.4) according to the requirements of the security environment.

Fragmentation will yield a sequence of command or response data fields with no ascertained structure. The CLA INS P1 P2 bytes shall comply with the chaining rules used for fragmentation (see 5.3).

10.1.3 SM DOs

Table 49 shows the SM DOs specified in this document, all in the context-specific class. Some SM DOs (SM tags '82', '83', 'B0', 'B1') are recursive, i.e. the plain value field is an SM field.

In each SM field, bit b1 of the last byte of the tag field (tag parity) of each SM DO (context-specific class) indicates whether the SM DO shall be included (bit b1 set to 1, odd tag number) or not (bit b1 set to 0, even tag number) in the computation of a data element for authentication: cryptographic checksum (see 10.2.3.1), or digital signature (see 10.2.3.2). If present, the DOs of the other classes (e.g. interindustry DOs) shall be included in the computation. If such a computation occurs, the data element shall be the value field of an SM DO for authentication (SM tags '8E', '9E') at the end of the SM field.

There are two categories of SM DOs.

— Each basic SM DO (see 10.2) conveys a plain value, or an input or result of a security mechanism.

— Each auxiliary SM DO (see 10.3) conveys a control reference template, or a security environment identifier, or a response descriptor template.

NOTE Basic SM DOs are also used to control security operations (see ISO/IEC 7816-8). Auxiliary SM DOs are also used to manage security environment (see 11.5.11). The global approach to security by secure messaging shares several security-related issues with the security operations, i.e. the atomic approach to security. Annex B illustrates the synergy between the two approaches.

Table 49 — SM data objects

Tag	Value
'80', '81'	Plain value not encoded in BER-TLV
'82', '83'	Cryptogram (plain value encoded in BER-TLV and including SM DOs i.e. an SM field)
'84', '85'	Cryptogram (plain value encoded in BER-TLV, but not including SM DOs)
'86', '87'	Padding-content indicator byte followed by cryptogram (plain value not encoded in BER-TLV)
'89'	Command header (CLA INS P1 P2, four bytes)
'8E'	Cryptographic checksum (at least four bytes)
'90', '91'	Hash-code
'92', '93'	Certificate (data not encoded in BER-TLV)
'94', '95'	Security environment identifier (SEID)
'96', '97'	One or two bytes encoding L_e in the unsecured C-RP (possibly empty)
'99'	Processing status (SW1-SW2, two bytes; possibly empty)
'9A', '9B'	Input data element for the computation of a digital signature (the value field is signed)
'9C', '9D'	Public key
'9E'	Digital signature
'A0', 'A1'	Input template for the computation of a hash-code (the template is hashed)
'A2'	Input template for the verification of a cryptographic checksum (the template is included)
'A4', 'A5'	Control reference template for authentication (AT)
'A6', 'A7'	Control reference template for key agreement (KAT)
'A8'	Input template for the verification of a digital signature (the template is signed)
'AA', 'AB'	Control reference template for hash-code (HT)
'AC', 'AD'	Input template for the computation of a digital signature (the concatenated value fields are signed)
'AE', 'AF'	Input template for the verification of a certificate (the concatenated value fields are certified)
'B0', 'B1'	Plain value encoded in BER-TLV and including SM DOs, i.e. an SM field
'B2', 'B3'	Plain value encoded in BER-TLV, but not including SM DOs
'B4', 'B5'	Control reference template for cryptographic checksum (CCT)
'B6', 'B7'	Control reference template for digital signature (DST)
'B8', 'B9'	Control reference template for confidentiality (CT)
'BA', 'BB'	Response descriptor template
'BC', 'BD'	Input template for the computation of a digital signature (the template is signed)
'BE'	Input template for the verification of a certificate (the template is certified)
— In SM fields, ISO/IEC JTC 1/SC 17 reserves any other DO of the context-specific class. — In order to use those SM DOs out of SM C-RPs, one shall encapsulate them in a DO'7D'.	

10.2 Basic SM DOs

10.2.1 SM DOs for encapsulating plain values

Encapsulation is mandatory for SM fields and for data not encoded in BER-TLV. It is optional for BER-TLV, not including SM, DOs. Table 50 shows SM DOs for encapsulating plain values.

Table 50 — SM DOs for encapsulating plain values

Tag	Value
'B0', 'B1'	Plain value encoded in BER-TLV and including SM DOs (i.e. an SM field)
'B2', 'B3'	Plain value encoded in BER-TLV, but not including SM DOs
'80', '81'	Plain value not encoded in BER-TLV
'89'	Command header (CLA INS P1 P2, four bytes)
'96', '97'	One or two bytes encoding L_e in the unsecured C-RP (possibly empty)
'99'	Processing status (SW1-SW2, two bytes; possibly empty)

10.2.2 SM DOs for confidentiality

Table 51 shows SM DOs for confidentiality. A security mechanism for confidentiality consists of an appropriate cryptographic algorithm in an appropriate mode of operation. In the absence of explicit indication and when no mechanism is implicitly selected for confidentiality, a default mechanism shall apply.

— For computing a cryptogram to be preceded by a padding indication, the default mechanism is a block cipher in “electronic code book” mode, which may involve padding. Padding for confidentiality may have an influence on transmission: the cryptogram (one or more blocks) may be longer than the plain value.

— For computing a cryptogram not to be preceded by a padding indication, the default mechanism is a stream cipher. In this case, the cryptogram is the exclusive-or of the string of data bytes to conceal with a concealing string of the same length. Concealment thus requires no padding and the string of data bytes is recovered by the same operation.

Padding and/or content shall be indicated when the plain value is not encoded in BER-TLV. If padding applies but is not indicated, the rules specified in 10.2.3.1 apply.

Table 52 shows the padding-content indicator byte.

Table 51 — SM DOs for confidentiality

Tag	Value
'82', '83'	Cryptogram (plain value encoded in BER-TLV and including SM DOs, i.e. an SM field)
'84', '85'	Cryptogram (plain value encoded in BER-TLV, but not including SM DOs)
'86', '87'	Padding-content indicator byte (see Table 52) followed by cryptogram (plain value not encoded in BER-TLV)

Table 52 — Padding-content indicator byte

Value	Meaning
'00'	No further indication
'01'	Padding as specified in 10.2.3.1
'02'	No padding
'1X'	One to four secret keys for enciphering information, not keys ('X' is a bitmap with any value from '0' to 'F') '11' indicates the first key (e.g. an “even” control word in a pay TV system) '12' indicates the second key (e.g. an “odd” control word in a pay TV system) '13' indicates the first key followed by the second key (e.g. a pair of control words in a pay TV system)
'2X'	Secret key for enciphering keys, not information ('X' is a reference with any value from '0' to 'F') (e.g. in a pay TV system, either an operational key for enciphering control words, or a management key for enciphering operational keys)
'3X'	Private key of an asymmetric key pair ('X' is a reference with any value from '0' to 'F')
'4X'	Password ('X' is a reference with any value from '0' to 'F')
'80' to '8E'	Proprietary
— Any other value is RFU.	

10.2.3 SM DOs for authentication

Table 53 shows SM DOs for authentication.

Table 53 — SM DOs for authentication

Tag	Value
'8E'	Cryptographic checksum (at least four bytes)
'90', '91'	Hash-code
'92', '93'	Certificate (data not encoded in BER-TLV)
'9C', '9D'	Public key
'9E'	Digital signature
Input DOs (see also ISO/IEC 7816-8)	
'9A', '9B'	Input data element for the computation of a digital signature (the value field is signed)
'A0', 'A1'	Input template for the computation of a hash-code (the template is hashed)
'A2'	Input template for the verification of a cryptographic checksum (the template is included)
'A8'	Input template for the verification of a digital signature (the template is signed)
'AC', 'AD'	Input template for the computation of a digital signature (the concatenated value fields are signed)
'AE', 'AF'	Input template for the verification of a certificate (the concatenated value fields are certified)
'BC', 'BD'	Input template for the computation of a digital signature (the template is signed)
'BE'	Input template for the verification of a certificate (the template is certified)

10.2.3.1 Cryptographic checksum data element

The computation of a cryptographic checksum involves an initial check block, a secret key and either a block cipher algorithm (see ISO/IEC 18033 [21]), or a hash-function (see ISO/IEC 10118 [14]).

The computation method may be part of the system specifications. Alternatively, a cryptographic mechanism identifier template (see 9.2) may identify a standard (e.g. ISO/IEC 9797-1) fixing a computation method.

Unless otherwise specified, the following computation method shall be used. The parameters used in this computation method may be defined in the relevant CRT (see Table 55). This standard does not define a default CRT.

Under the control of the key, the algorithm basically converts a current input block of k bytes (typically 8, 16 or 20) into a current output block of the same size. The computation is performed in the following consecutive stages.

Initial stage — The initial stage shall set either one of the following blocks as the initial check block:

— the null block, i.e. k bytes set 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 converting auxiliary data under the control of the key. If the auxiliary data is less than k bytes, then bits set to 0 head it up to the block size.

Sequential stage(s) — The command header (CLA INS P1 P2) may be encapsulated for protection (SM tag '89'). However, if bits b8 to b6 of CLA are set to 000 and bits b4 and b3 to 11 (see 5.1), then the first data block consists of the command header (CLA INS P1 P2) followed by one byte set to '80' and k–5 bytes set to '00'.

The cryptographic checksum shall include any secure messaging DO having an odd tag number and any DO with the first byte not from '80' to 'BF'. Those DOs shall be included data block by data block in the current check block. The splitting into data blocks shall be performed as follows.

— The blocking shall be continuous at the border between adjacent DOs to include.

— The padding shall apply at the end of each DO to include followed either by a DO not to include, or by no further DO. The padding consists of one mandatory byte set 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.

In this mechanism, the mode of operation is "cipher block chaining" (see ISO/IEC 10116 [13]). 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.

Final stage — The final check block is the last output. The final stage extracts a cryptographic checksum (first m bytes, at least four) from the final check block.

10.2.3.2 Digital signature data element

The digital signature schemes rely on asymmetric cryptographic techniques (see ISO/IEC 9796 [8], ISO/IEC 14888 [19]). The computation implies a hash-function (see ISO/IEC 10118 [14]). The data input consists of the value field of a digital signature input DO, or of the concatenation of the value fields of DOs forming a digital signature input template. It may be determined by the mechanism specified in 10.2.3.1.

10.3 Auxiliary SM DOs

Table 54 shows auxiliary SM DOs.

Table 54 — Auxiliary SM DOs

Tag	Value
'94', '95'	Security environment identifier (SEID)
'A4', 'A5'	Control reference template valid for authentication (AT)
'A6', 'A7'	Control reference template valid for key agreement (KAT)
'AA', 'AB'	Control reference template valid for hash-code (HT)
'B4', 'B5'	Control reference template valid for cryptographic checksum (CCT)
'B6', 'B7'	Control reference template valid for digital signature (DST)
'B8', 'B9'	Control reference template valid for confidentiality (CT)
'BA', 'BB'	Response descriptor template

10.3.1 Control reference templates

Six control reference templates are defined, valid for authentication (AT), key agreement (KAT), hash-code (HT), cryptographic checksum (CCT), digital signature (DST) and confidentiality (CT) using either symmetric or asymmetric cryptographic techniques (CT-sym and CT-asym).

Each security mechanism involves a cryptographic algorithm in a mode of operation and uses a key and, possibly, initial data. Such items are selected either implicitly, i.e. known before issuing the command, or explicitly, i.e. by control reference DOs nested in control reference templates. Within the control reference templates, the context-specific class is reserved for control reference DOs.

In an SM field, the last possible position of a control reference template is just before the first DO to which the referred mechanism applies. For example, the last possible position of a template valid for cryptographic checksum (CCT) is just before the first DO to include in the computation.

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

10.3.2 Control reference DOs in control reference templates

Each control reference template (CRT) is a set of control reference DOs: a cryptographic mechanism reference, a file and key reference, an initial data reference, a usage qualifier and, only in a control reference template for confidentiality, a cryptogram content reference.

— The cryptographic mechanism reference denotes a cryptographic algorithm in a mode of operation. The CPs of any DF (see tag 'AC' in Table 10) may contain cryptographic mechanism identifier templates (see 9.2). Each one indicates the meaning of a cryptographic mechanism reference.

— The file reference (same encoding as in 7.3.2) 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, possibly an application DF. The key reference unambiguously identifies the key to use.

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

Table 55 lists control reference DOs and indicates to which control reference template they are relevant. All the control reference DOs are in the context-specific class.

A CRT may contain interindustry DOs, e.g. certificate holder authorization (tag '5F4C', see 0) in AT, header list or extended header list (tags '5D' and '4D', see 8.4.4 to 8.4.7) in HT or DST. In any control reference template, a key usage template under tag 'A3' may associate a file and key reference with a key usage counter and/or a key retry counter. The key usage template contains a set of key usage DOs (see Table 56).

Table 55 — Control reference DOs in control reference templates

Tag	Value	AT	KAT	HT	CCT	DST	CT-asym	CT-sym
'80'	**Cryptographic mechanism reference**	x	x	x	x	x	x	x
File and security object references								
'81'	— File reference (for encoding, see 7.3.2)	x	x	x	x	x	x	x
'82'	— DF name (see 7.3.1)	x	x	x	x	x	x	x
'83'	— Reference of a secret key (for direct use)	x	x	x	x			x
	— Reference of a public key	x	x	x		x	x	
	— Reference data	x						
'84'	— Reference for computing a session key	x	x		x			x
	— Reference of a private key	x	x			x	x	
'A3'	Key usage template (see text below)	x	x	x	x	x	x	x
Initial data reference: Initial check block								
'85'	— L=0, null block			x	x			x
'86'	— L=0, chaining block			x	x			x
'87'	— L=0, previous initial value block plus one L=k, initial value block			 x	x x			x
Initial data reference: Auxiliary data elements (see also 10.2.3.1)								
'88'	— L=0, previous exchanged challenge plus one L>0, no further indication				x	x	x	x
'89' to '8D'	— L=0, index of a proprietary data element L>0, value of a proprietary data element				x x			x
'90'	— L=0, hash-code provided by the card			x		x		
'91'	— L=0, random number provided by the card L>0, random number		x		x	x x	x x	
'92'	— L=0, time stamp provided by the card L>0, time stamp			x		x x	x x	
'93'	— L=0, previous digital signature counter plus one L>0, digital signature counter			x		x x	x x	x x
'94'	Challenge or data element for deriving a key	x			x			x
'95'	**Usage qualifier byte** (see text below)	x	x		x	x	x	x
'8E'	**Cryptogram content reference** (see text below)						x	x
— In a control reference template, ISO/IEC JTC 1/SC 17 reserves any other DO of the context-specific class.								

Table 56 — Key usage DOs

Tag	Value
'80' to '84'	File and key references as specified in Table 55
'90'	Key usage counter
'91'	Key retry counter
— In this context, ISO/IEC JTC 1/SC 17 reserves any other DO of the context-specific class.	

In any control reference template for authentication (AT), for key agreement (KAT), for cryptographic checksum (CCT), for confidentiality (CT) or for digital signature (DST), a usage qualifier byte (tag '95') may specify the usage of the template either as a security condition (see 9.3.3 and Table 33), or in compliance with the MANAGE SECURITY ENVIRONMENT command (see 11.5.11). Table 57 shows the usage qualifier byte.

Table 57 — Coding of the usage qualifier byte

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	Verification (DST, CCT), Encipherment (CT), External authentication (AT), Key agreement (KAT)
-	1	-	-	-	-	-	-	Computation (DST, CCT), Decipherment (CT), Internal authentication (AT), Key agreement (KAT)
-	-	1	-	-	-	-	-	Secure messaging in response data fields (CCT, CT, DST)
-	-	-	1	-	-	-	-	Secure messaging in command data fields (CCT, CT, DST)
-	-	-	-	1	-	-	-	User authentication, password based (AT)
-	-	-	-	-	1	-	-	User authentication, biometry based (AT)
-	-	-	-	-	-	x	x	00 (any other value is RFU)

In any control reference template for confidentiality (CT), a cryptogram content reference (tag '8E') may specify the content of the cryptogram. The first byte of the value field is mandatory; its name is the cryptogram descriptor byte. Table 58 shows the cryptogram descriptor byte.

Table 58 — Cryptogram descriptor byte

Value	Meaning
'00'	No further indication
'1X'	One to four secret keys for enciphering information, not keys ('X' is a bitmap with any value from '0' to 'F') '11' indicates the first key (e.g. an “even” control word in a pay TV system) '12' indicates the second key (e.g. an “odd” control word in a pay TV system) '13' indicates the first key followed by the second key (e.g. a pair of control words in a pay TV system)
'2X'	Secret key for enciphering keys, not information ('X' is a reference with any value from '0' to 'F') (e.g. in a pay TV system, either an operational key for enciphering control words, or a management key for enciphering operational keys)
'3X'	Private key of an asymmetric key pair ('X' is a reference with any value from '0' to 'F')
'4X'	Password ('X' is a reference with any value from '0' to 'F')
'80' to 'FF'	Proprietary
— Any other value is RFU.	

10.3.3 Security environments

This clause specifies security environments (SE) for referencing cryptographic algorithms, modes of operation, protocols, procedures, keys and any additional data needed for secure messaging and for security operations (see ISO/IEC 7816-8). An SE consists of data elements stored in the card or resulting from some computation, to be processed by the specified algorithms. An SE may contain a mechanism to initialise non-persistent data to be used in the environment, e.g. a session key. An SE may provide directions for handling computation results, e.g. storage in the card. An interindustry SE template (tag '7B') describes an SE.

SE identifier — An SE identifier (SEID) may reference any security environment, e.g. for secure messaging and for storing and restoring by a MANAGE SECURITY ENVIRONMENT command (see 11.5.11).

— Unless otherwise specified by the application, the value '00' denotes an empty environment where no secure messaging and no authentication are defined.

— The value 'FF' denotes that no operation can be performed in this environment.

— Unless otherwise specified by the application, the value '01' is reserved for the default SE, always available. This clause does not specify the content of the default SE; it may be empty.

— The value 'EF' is RFU.

Components — Control reference templates (CRT) may describe various components of an SE. Any relative control reference (files, keys or data) specified with a mechanism in the environment definition shall be resolved with respect to the curDF selected before using the mechanism. Absolute control references (e.g. absolute path) need not be resolved. Within an SE, components may have two aspects: one being valid for SM in command data fields and the other for SM in response data fields.

At any time during card operation, a current SE shall be active, either by default or as a result of commands performed by the card. The current SE contains one or more components among the following components.

— Some components belong to the default SE associated with the curDF.

— Some components are transmitted in commands using secure messaging.

— Some components are transmitted in MANAGE SECURITY ENVIRONMENT commands.

— Some components are invoked by an SEID in a MANAGE SECURITY ENVIRONMENT command.

The current SE is valid until

— disabling a physical interface (see 5.1), or

— resetting a logical channel (see 11.1.2), or

— a change of VA (e.g. a change of curDF), or

— a MANAGE SECURITY ENVIRONMENT command (see 11.5.11) setting or replacing the current SE.

In SM, control reference DOs transmitted in a CRT shall take precedence over any corresponding control reference DO present in the current SE.

Certificate holder authorization — Authentication procedures may use card-verifiable certificates, i.e. templates that can be interpreted and verified by the card by a VERIFY CERTIFICATE operation using a public key (see ISO/IEC 7816-8). In such a certificate, a certificate holder authorization (e.g. a role identifier) may be conveyed in an interindustry DO'5F4C'. If such a data element is used in the security conditions to fulfil for accessing data or functions, then DO'5F4C' shall be present in the control reference template for authentication (AT) describing the authentication procedure.

NOTE In the first edition of ISO/IEC 7816-9, tag '5F4B' references a certificate holder authorization (data element of five or more bytes). In amendment 1 to the first edition of ISO/IEC 7816-6, tag '5F4B' references an integrated circuit manufacturer identifier (1-byte data element). Consequently, tag '5F4B' is deprecated in ISO/IEC 7816 [6].

Access control — The card may store security environments used for access control within EFs (see tag '8D' in Table 10) containing interindustry SE templates (tag '7B'). An application may store DOs'7B' in the VA set by application selection (e.g. 1st generation DOs). Within the interindustry SE template (tag '7B'), the context-specific class is reserved for security environment DOs. As listed in Table 59 for every included SE, the security environment template contains an SEID DO'80', an optional LCS DO'8A', one or more optional cryptographic mechanism identifier template(s) (tag 'AC') and one or more CRTs (tags 'A4', 'A6', 'AA', 'B4', 'B6', 'B8', as SM tags).

Table 59 — Security environment DOs

Tag	Value
'80'	SEID (1byte, mandatory)
'8A'	LCS (1 byte, see 7.4.10 and Table 14), optional
'AC'	Cryptographic mechanism identifier template (see 9.2), optional
'A4', 'A6', 'AA', 'B4', B6', 'B8'	CRTs (see 0)
— Under tag '7B', ISO/IEC JTC 1/SC 17 reserves any other DO of the context-specific class.	

If present in the SE template, the LCS DO'8A' indicates for which LCS the SE is valid. If the SE is used for access control, e.g. to a file, then the LCS of the file and the LCS of the SE have to match. If no LCS DO'8A' is present, then the SE is valid for the activated operational state.

In the SE template, if a CRT carries several DOs with the same tag (e.g. DOs specifying a key reference), then at least one of the DOs has to be fulfilled (OR condition).

SE retrieval — Any CRT in the current SE may be retrieved by a GET DATA command with INS set to 'CA', P1-P2 set to '004D' (extended header, see 8.4.5) and a command data field consisting of an SE template (tag '7B') containing one or more pairs, each one consisting of a CRT tag followed by '80' (see 8.4.1 for the use of a length set to '80' in an extended header). One may also use the GET option of the MANAGE SECURITY ENVIRONMENT command.

10.3.4 Response descriptor template

Each command data field may contain a response descriptor template. If present in the command data field, the response descriptor template shall indicate the SM DOs required in the response data field. Inside the response descriptor template, the security mechanisms are not yet applied; the receiving entity shall apply them for constructing the response data field. The security items (algorithms, modes of operation, keys and initial data) used for processing the command data field may be different from those used for producing the response data field. The following rules apply.

— The card shall fill each empty primitive basic SM DO.

— Each CRT present in the response descriptor template shall be present in the response at the same place with the same control reference DOs for security mechanisms, files and keys.

  — If the response descriptor template provides auxiliary data, then the respective DO shall be empty in the response.

  — If an empty reference DO for auxiliary data is present in the response descriptor template, 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 basic SM DOs.

10.4 SM impact on command-response pairs

Figure 6 illustrates a C-RP.

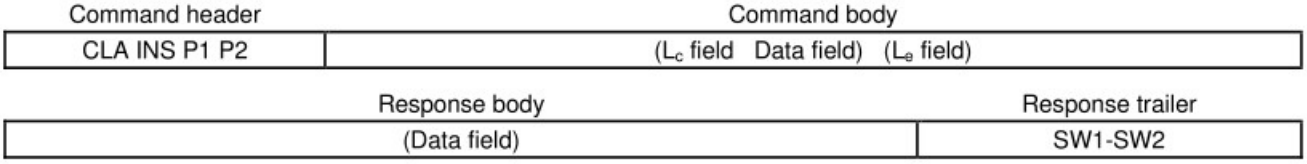


Figure 6 — Command-response pair

The following rules apply for securing a C-RP of the interindustry class (see 5.4.1), i.e. when switching either bit b4 from 0 to 1 in CLA where bits b8, b7 and b6 are set to 000, or bit b6 from 0 to 1 in CLA where bits b8 and b7 are set to 01. The notation CLA* means that secure messaging is indicated in CLA.

— The secured command data field is an SM field; it shall be formed as follows.
  — If a command data field is present ($N_c > 0$), then either a plain value DO (SM tags '80', '81', 'B2', 'B3'), or a DO for confidentiality (SM tags '84', '85', '86', '87') shall convey the N_c bytes.
  — The command header (four bytes) may be encapsulated for protection (SM tag '89').
  — If an L_e field is present, then a new L_e field (containing only bytes set to '00') and an L_e DO (SM tags '96', '97') shall be present. If present, the value of an L_e DO shall provide the L_e for the unprotected response. An empty L_e DO ('9600' or '9700') means the maximum, i.e. 256 or 65 536 depending upon whether the new L_e field is short or extended. Otherwise, if the L_e field for the unprotected response consists of:
    — one byte, this byte shall become the value field of the L_e DO.
    — two bytes, these two bytes shall become the value field of the L_e DO.
    — three bytes, i.e. one byte set to '00' followed by two bytes with any value, the two bytes shall become the value field of the L_e DO.

WARNING — Some implementations encode a three byte L_e field into an L_e DO with a 3-byte value field

— The secured response data field is an SM field; it shall be interpreted as follows.
  — If present, a plain value DO (SM tags '80', '81', 'B2', 'B3') or a DO for confidentiality (SM tags '84', '85', '86', '87') conveys the response data bytes.
  — If present, a processing status DO (SM tag '99') conveys SW1-SW2 encapsulated for protection. The empty processing status DO means SW1-SW2 set to '9000'.

Figure 7 shows the corresponding secured C-RP.

Command header	Command body
CLA* INS P1 P2	(New L_c field - {Secured data field = (T-N_c-Data bytes) - (T-'01' or '02'-L_e)}) - (New L_e field)

Response body	Response trailer
Secured data field = (T-N_r-Data bytes) - (T-'02'-SW1-SW2)	SW1-SW2

Figure 7 — Secured command-response pair

When bit b1 of INS is set to 1 (odd INS code, see 5.5), the unsecured data fields are encoded in BER-TLV and SM tags 'B2', 'B3', '84' and '85' shall be used for their encapsulation; unless the use of tags '80', '81', '86' and '87' is specified at application level.

Otherwise (even INS code, see 5.5), as the format of the data fields to protect is not always apparent, SM tags '80', '81', '86' and '87' are recommended.

— The secured data fields are SM fields; they may contain further or other SM DOs, e.g. a cryptographic checksum (SM tag '8E') or a digital signature (SM tag '9E') at the end.
— The new L_c field encodes the number of bytes in the secured command data field.
— The new L_e field shall be absent when no data field is expected in the secured response data field; otherwise, it shall contain only bytes set to '00'.
— The response trailer indicates the status of the receiving entity after processing the secured command. The following specific error conditions may occur.
  — If SW1-SW2 is set to '6987', then expected secure messaging DOs are missing.

— If SW1-SW2 is set to '6988', then secure messaging DOs are incorrect.

Annex B provides illustrative examples of secure messaging.

11 Commands for interchange

This clause specifies commands for interchange. It shall not be mandatory for all cards complying with this document to support all those commands or all the options of a supported command. When interchange is required, a set of application-independent card services and related commands and options shall be used as specified in clause 12.

11.1 Selection

After enabling a physical interface (see 5.1), the VA on the basic logical channel is as defined in 7.2.2. The historical bytes (see 12.1.1) or the initial data string (see 12.1.2) may indicate an implicitly selected application.

Subclause 11.1.1 covers the selection of files, applications and DOs. Additional DO selection functions are covered in 11.4.2.

NOTE Because it is impossible to use a short EF identifier in a SELECT/SELECT DATA command, the FCP DO'A2' of DFs are irrelevant for a SELECT/SELECT DATA command. Thus indirect file referencing (see 7.4.11) never occurs during the processing of a SELECT/SELECT DATA command.

11.1.1 SELECT command

When completed, the command opens the logical channel (see 5.4.2) numbered in CLA (see 5.4.1), if not yet opened, and sets a current structure within that logical channel. Subsequent commands may implicitly refer to the current structure through that logical channel.

The selected DF (MF, application DF, DF) becomes current in the logical channel (see 7.2.2 rules d) and e)). The previously selected DF, if any, is no longer referred to through that logical channel. After such a selection, an implicit current EF may be referred to through that logical channel.

The selection of an EF sets a pair of current files: the EF and its parent DF (see 7.2.2 rule f)).

Table 60 — SELECT command-response pair

CLA	As defined in 5.4.1
INS	'A4'
P1	See Table 61
P2	See Table 62
L_c field	Absent for encoding $N_c = 0$, present for encoding $N_c > 0$
Data field	Absent or file identifier or path or DF name or tag (according to P1)
L_e field	Absent for encoding $N_e = 0$, present for encoding $N_e > 0$

Data field	Absent or control information (according to P2)
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6283', '6284', '6A80', '6A81', '6A82', '6A86', '6A87', '6A88'

Table 61 — Coding of P1 in the SELECT command

b8	b7	b6	b5	b4	b3	b2	b1	Meaning	Command data field
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	File identifier or absent
0	0	0	0	0	0	0	1	Select child DF	file identifier referencing a DF
0	0	0	0	0	0	1	0	Select EF under the current DF	file identifier referencing an EF
0	0	0	0	0	0	1	1	Select parent DF of the current DF	Absent
0	0	0	0	0	1	**x**	**x**	**Selection by DF name**	
0	0	0	0	0	1	0	0	Select by DF name	e.g. (truncated) application identifier
0	0	0	0	1	0	**x**	**x**	**Selection by path**	
0	0	0	0	1	0	0	0	Select from the MF	Path without the MF identifier
0	0	0	0	1	0	0	1	Select from the current DF	Path without the current DF identifier
0	0	0	1	0	0	**x**	**x**	**Selection of DO**	
0	0	0	1	0	0	0	0	Select a DO in the current template	Tag belonging to the current template
0	0	0	1	0	0	1	1	Select parent DO of the constructed DO setting the current template	Absent

— Any other value is RFU.

— When present in the historical bytes (see 12.1.1) or in EF.ATR/INFO (see 12.2.2), the first software function table (see Table 117) indicates selection methods supported by the card.

Table 62 — Coding of P2 in the SELECT command

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	-	-	x	x	**File or DO occurrence**
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	-	-	**Response data field requirements** (see 7.4.2 and Table 8)
0	0	0	0	0	0	-	-	— Return FCI template, optional use of FCI tag and length
0	0	0	0	0	1	-	-	— Return CP template, mandatory use of CP tag and length
0	0	0	0	1	0	-	-	— Return FMD template, mandatory use of FMD tag and length — Return the tags belonging to the template set by the selection of a constructed DO as a tag list
0	0	0	0	1	1	-	-	— No response data if L_e field absent, or proprietary if L_e field present

— Any other value is RFU.

Unless otherwise specified, the following rules apply to each open logical channel within a hierarchy of DFs.

— If the current EF is changed, or when there is no current EF, then the security status, if any, specific to the previously selected EF shall be revoked.

— If the current DF is a descendant of, or identical to the previously selected DF, then the security status specific to the previously selected DF shall be maintained.

— If the current DF is neither a descendant of, nor identical to the previously selected DF, then the security status specific to the previously selected DF shall be revoked. The security status common to all common ancestors of the previously selected DF and the new current DF shall be maintained.

If P1 is set to '00', then the card knows whether the file to select is the MF, a DF or an EF, either because of a specific encoding of the file identifier, or because of the command processing context. If P2 is also set to '00' and the command data field

— provides a file identifier, then that file identifier shall be unique in the following three environments: the immediate children of the current DF, the parent DF and the immediate children of the parent DF.

— is absent or set to '3F00', then the MF shall be selected.

If P1 is set to '04', then the command data field is a DF name, which may be an application identifier (see 12.2.3), possibly right truncated. If 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 command without data field, then all or a subset of the DFs can be successively selected.

If P1 is set to '10', then the command data field is a tag belonging to the current template. The selected DO becomes the current DO. If the selected DO is constructed, it sets its template as current template. For handling multiple instances of DOs, the use of bits b1 b2 in P2 (see Table 62) is mandatory.

If the L_e field contains only bytes set to '00', then all the bytes corresponding to the selection option should be returned within the limit of 256 for a short L_e field, or 65 536 for an extended L_e field. If the L_e field is absent, the available control information (CPs and/or FMDs) of the selected structure may be recovered through DO handling (see 11.4.3 and 11.4.4).

11.1.2 MANAGE CHANNEL command

When completed, the command opens or closes a logical channel (see 5.4.2) other than the basic one, i.e. a logical channel numbered from one to nineteen (the greater numbers are RFU). The reset function may apply to any logical channel.

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

- If bits b8 and b7 of P1 are set to 00 (i.e. P1 set to '00' because the other six bits are RFU), then MANAGE CHANNEL shall open a logical channel numbered from one to nineteen as follows.
  - If P2 is set to '00', then the L_e field shall be '01' (short format) or '000001' (expanded format), and the response data field shall consist of a single byte for encoding a non-zero channel number assigned by the card from '01' to '13'.
  - If P2 is set from '01' to '13', then it encodes an externally assigned non-zero channel number and the L_e field shall be absent.
- After an open function performed from the basic logical channel (CLA encoding zero as channel number), the MF or a default application DF shall be implicitly selected as the current DF on the new logical channel.
- After an open function performed from a non-basic logical channel (CLA encoding a non-zero channel number), the current DF on the logical channel numbered in CLA shall be selected according to 7.2.2 rule e) on the new logical channel.

The close function explicitly closes a logical channel other than the basic one. The L_e field shall be absent. After closing, the logical channel shall be available for re-use. If bits b8 and b7 of P1 are set to 10 (i.e. P1 set to '80' because the other six bits are RFU), then MANAGE CHANNEL shall close a logical channel numbered from 1 to nineteen as follows.

- If P2 is set to '00', then the logical channel numbered in CLA (a non-zero channel number) shall be closed.
- If P2 is set from '01' to '13', then the logical channel numbered in P2 shall be closed.

WARNING — The close function may be aborted if CLA indicates neither the basic logical channel nor the logical channel numbered in P2. P1-P2 codings in the range '8001' to '8013' will be deprecated in the future.

The reset function explicitly closes and opens again any logical channel defined by CLA. The L_e field shall be absent. After resetting, the VA on the logical channel is defined by 7.2.2. The security status set on this logical channel shall be revoked.

The command which resets the basic logical channel and closes all additional logical channels shall be sent on the basic logical channel.

NOTE At APDU level, this function (P1-P2='4001') is equivalent to enabling a physical interface (see 5.1).

Table 63 — MANAGE CHANNEL command-response pair

CLA	As defined in 5.4.1	
INS	'70'	
P1-P2	'0000'	opens a logical channel to be numbered in the response data field
	'0001' to '0013'	opens the logical channel numbered in P2
	'8000'	closes the logical channel numbered in CLA (other than the basic logical channel)
	'8001' to '8013'	closes the logical channel numbered in P2
	'4000'	resets the logical channel numbered in CLA
	'4001'	resets the basic logical channel and closes all additional logical channels
	other values	RFU
Data field	Absent	
L_e field	Absent for encoding $N_e = 0$, present for encoding $N_e = 1$	

Data field	Absent (P1-P2 not set to '0000'), or '01' to '13' (P1-P2 set to '0000')
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6200', '6482', '6881', '6A81'

11.2 Data unit handling

11.2.1 Data units

Within each EF supporting data units, an offset shall reference each data unit. From zero for the first data unit of the EF, the offset is incremented by one for each subsequent data unit. The offset data element is binary encoded on the minimum number of bytes. Reference to a data unit not contained in the EF is an error.

The card can provide a data coding byte (see

Table 118) in the historical bytes (see 12.1.1), in EF.ATR/INFO (see 12.2.2) and in the control information of any file (see DO'82' in Table 10). The data coding byte fixes a data unit size. If the card provides data coding bytes in several places, then 7.4.5 specifies which data coding byte has to be taken into account.

11.2.2 General

Any command of this group shall be aborted if applied to an EF not supporting data units. It can be performed on an EF only if the security status satisfies the security attributes defined for the function, namely, read, write, update, erase, search or compare.

Each command of this group may use either a short EF identifier or a file identifier. If there is a current EF at the time of issuing the command, then the process may be completed on that EF by just setting all the corresponding bits to 0. If the process is completed, then the identified EF becomes current.

INS P1 P2 — All the commands of this group shall use bit b1 of INS and bit b8 of P1 as follows.

— If bit b1 of INS is set to 0 and bit b8 of P1 to 1, then bits b7 and b6 of P1 are set to 00 (other values are RFU), bits b5 to b1 of P1 encode a short EF identifier and P2 (eight bits) encodes an offset from 0 to 255 in the EF referenced by the command.

— If bit b1 of INS is set to 0 and bit b8 of P1 to 0, then P1-P2 (fifteen bits) encodes an offset in the current EF from 0 to 32 767.

— If bit b1 of INS is set to 1, then P1-P2 shall identify an EF. If the first eleven bits of P1-P2 are set to 0 and if bits b5 to b1 of P2 are not all equal and if the card and/or the EF supports selection by short EF identifier, then bits b5 to b1 of P2 encode a short EF identifier (a number from one to thirty). Otherwise, P1-P2 is a file identifier. P1-P2 set to '0000' identifies the current EF. At least one offset DO'54' shall be present in the command data field. When present in a command or response data field, data shall be encapsulated in a discretionary DO'53' ('73' is deprecated for this use).

In this group of commands:

— SW1-SW2 set to '63CX' indicates a successful change of memory state, but after an internal retry routine; 'X' > '0' encodes the number of retries; 'X' = '0' means that no counter is provided.

— SW1-SW2 set to '6B00' should be used to indicate that the offset points outside of EF.

11.2.3 READ BINARY command

The response data field gives (part of) the content of an EF supporting data units. If the L_e field contains only bytes set to '00', then all the bytes until the end of the file should be read within the limit of 256 for a short L_e field, or 65 536 for an extended L_e field.

Table 64 — READ BINARY command-response pair

CLA	As defined in 5.4.1		
INS	'B0' or 'B1'		
P1-P2	INS = 'B0'	Mandatory offset, possible short EF identifier	See 11.2.2
	INS = 'B1'	EF identifier, or short EF identifier	
L_c field	Absent for encoding $N_c = 0$, present for encoding $N_c > 0$		
Data field	INS = 'B0'	Absent	
	INS = 'B1'	Offset DO	
L_e field	Present for encoding $N_e > 0$		

Data field	INS = 'B0'	Data read
	INS = 'B1'	Discretionary DO for encapsulating data read
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6281', '6282', '6700', '6981', '6982', '6986', '6A81', '6A82', '6B00'	

11.2.4 WRITE BINARY command

The command initiates one of the following operations into an EF according to the file attributes:

1) the logical-OR of the bits already present in the card with the bits given in the command data field (the logical erased state of the bits of the file is zero);

2) the write-once of the bits given in the command data field (the command shall be aborted if the string of data units is not in the logical erased state);

3) the logical-AND of the bits already present in the card with the bits given in the command data field (the logical erased state of the bits of the file is one).

By default, i.e. when a data coding byte is not available (see 11.2.1), operation 1) shall apply in that EF.

Table 65 — WRITE BINARY command-response pair

CLA	As defined in 5.4.1		
INS	'D0' or 'D1'		
P1-P2	INS = 'D0'	Mandatory offset, possible short EF identifier	See 11.2.2
	INS = 'D1'	EF identifier, or short EF identifier	
L_c field	Present for encoding $N_c > 0$		
Data field	INS = 'D0'	String of data units to be written	
	INS = 'D1'	offset DO and discretionary DO for encapsulating the string of data units to be written	
L_e field	Absent for encoding $N_e = 0$		

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '63CX' (see 11.2.2), '6581', '6700', '6981', '6982', '6B00'

11.2.5 UPDATE BINARY command

The command initiates the update of bits already present in an EF with the bits given in the command data field. When the process is completed, each bit of each specified data unit will have the value specified in the command data field.

Table 66 — UPDATE BINARY command-response pair

CLA	As defined in 5.4.1		
INS	'D6' or 'D7'		
P1-P2	INS = 'D6'	Mandatory offset, possible short EF identifier	See 11.2.2
	INS = 'D7'	EF identifier, or short EF identifier	
L_c field	Present for encoding $N_c > 0$		
Data field	INS = 'D6'	String of data units to be updated	
	INS = 'D7'	Offset DO and discretionary DO for encapsulating the string of updating data units	
L_e field	Absent for encoding $N_e = 0$		

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '63CX' (see 11.2.2), '6581', '6700', '6981', '6982', '6B00'

11.2.6 SEARCH BINARY command

The command initiates a search within an EF supporting data units. The response data field gives the offset of a data unit: the byte string at the returned offset within the EF shall have the same value as the search string in the command data field. The response data field is absent either because the L_e field is absent, or because no match is found. If the search string is absent, then the response data field gives the offset of the first data unit in a logically erased state.

Table 67 — SEARCH BINARY command-response pair

CLA	As defined in 5.4.1		
INS	'A0' or 'A1'		
P1-P2	INS = 'A0'	Mandatory offset, possible short EF identifier	See 11.2.2
	INS = 'A1'	EF identifier, or short EF identifier	
L_c field	Absent for encoding $N_c = 0$, present for encoding $N_c > 0$		
Data field	INS = 'A0'	Absent or search string	
	INS = 'A1'	Offset DO and discretionary DO for encapsulating the search string	
L_e field	Absent for encoding $N_e = 0$, present for encoding $N_e > 0$		

Data field	INS = 'A0'	Absent or offset of the first data unit matching the command data field
	INS = 'A1'	Offset DO indicating the first data unit matching the search string
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6282', '6982', '6B00'	

11.2.7 ERASE BINARY command

The command sets (part of) the content of an EF to its logical erased state, sequentially, starting from a given offset.

— If INS = '0E', then, if present, the command data field encodes the offset of the first data unit not to be erased. This offset shall be higher than the one encoded in P1-P2. If the data field is absent, then the command erases up to the end of the file.

— If INS = '0F', then, if present, the command data field shall consist of zero, one or two offset DOs. If there is no offset, then the command erases all the data units in the file. If there is one offset, it indicates the first data unit to be erased; then the command erases up to the end of the file. Two offsets define a sequence of data units: the second offset indicates the first data unit not to be erased; it shall be higher than the first offset.

Table 68 — ERASE BINARY command-response pair

CLA	As defined in 5.4.1		
INS	'0E' or '0F'		
P1-P2	INS = '0E'	Mandatory offset, possible short EF identifier	See 11.2.2
	INS = '0F'	EF identifier, or short EF identifier	
L_c field	Absent for encoding $N_c = 0$, present for encoding $N_c > 0$		
Data field	INS = '0E'	Absent or offset of the first data unit not to be erased	
	INS = '0F'	Absent or one or two offset DOs	
L_e field	Absent for encoding $N_e = 0$		

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '63CX' (see 11.2.2), '6581', '6700', '6981', '6982', '6B00'

11.2.8 COMPARE BINARY function

This function is supported by the COMPARE command (see 11.6.1).

11.3 Record handling

11.3.1 Records

Within each EF supporting records, a record number and/or a record identifier shall reference each record. Reference to a record not contained in the EF is an error.

Referencing by record number — Each record number is unique and sequential.

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

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

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

— Zero shall refer to the current record, i.e. that record referenced by the record pointer.

Referencing by record identifier — Each record identifier is provided by an application. Several records may have the same record identifier, in which case data contained in the records may be used for discriminating between them. If a record is a SIMPLE-TLV data object in a data field, then the record identifier is the first byte of the data object, i.e. the SIMPLE-TLV tag.

Referencing by record identifier shall induce the management of a record pointer. Enabling a physical interface, resetting a logical channel, a SELECT and any command using a valid short EF identifier for accessing an EF may affect the record pointer. Referencing by record number shall not affect the record pointer.

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

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

— Within each EF supporting a 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 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.

— If there is a current record, then 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.

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

— Zero shall refer to the first, last, next or previous record in the numbering sequence, independently from the record identifier.

11.3.2 General

Any command of this group shall be aborted if applied to an EF not supporting records. It can be performed on an EF only if the security status satisfies the security attributes defined for the function, namely, read, write, append, update, search, erase, activate or deactivate.

The records in an EF may support record life cycle. If so, in general deactivated records are not accessible by the commands READ RECORD, WRITE RECORD, UPDATE RECORD, ERASE RECORD, APPEND RECORD and COMPARE (RECORD). If such a command is used, the respective command returns with the status bytes '6287' (at least one of the referenced records is deactivated). Furthermore, deactivated records shall be ignored, when executing a SEARCH RECORD command. Further details and exceptions to the general rules stated above are given below.

Two commands of this group (READ, UPDATE) may use an odd INS code (data fields encoded in BER-TLV) for initiating an action on a part of a given record (partial read, partial update). Then an offset shall reference each byte inside a record: from 0 for the first byte of the record, the offset is incremented by one for each subsequent byte of the record. Reference to a byte not contained in the record is an error. As needed, the offset data element is binary encoded and referenced by tag '54'. When present in a command or response data field, data shall be encapsulated in a discretionary DO'53' ('73' is deprecated for this use).

Each command of this group may use a short EF identifier. If the process is completed, then the identified EF becomes current and the record pointer is reset. If there is a current EF at the time of issuing the command, then the process may be completed without indicating the EF (by just setting the corresponding five bits to 0).

P1 — Each record number or identifier is a number from one to 254, encoded by a value of P1 from '01' to 'FE'. 0 (encoded '00') is reserved for special purposes. 255 (encoded 'FF') is RFU.

NOTE If the number of records exceeds the numbering range ('01' to 'FE') of the record handling command, records can be handled e.g. by using next occurrence option of the record identifier.

P2 — Bits b8 to b4 are a short EF identifier according to Table 69. Bits 3 to 1 depend upon the command.

Table 69 — Coding of the short EF identifier in P2

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	-	-	-	Current EF
Not all equal					-	-	-	Short EF identifier (a number from one to thirty)
1	1	1	1	1	-	-	-	RFU

In this group of commands, SW1-SW2 set to '63CX' indicates a successful change of memory state, but after an internal retry routine; 'X' > '0' encodes the number of retries; 'X' = '0' means that no counter is provided.

11.3.3 READ RECORD (S) command

The response data field gives the (partial) contents of the addressed record(s) (or the beginning part of one record) within an EF.

If any record referenced by P1 and P2 is in record LCS DEACTIVATED, the command is processed with warning '6287' and the response data field shall be empty.

If INS = 'B2' and if the records are SIMPLE-TLV data objects (see 6.1), then Figure 8 illustrates the response data field. The comparison of N_r with the TLV structure indicates whether the unique record (read one record) or the last record (read all records) is incomplete, complete or padded.

NOTE If the records are not data objects, then the read-all-records function results in receiving records without delimitation.

If INS = 'B3', then the command does not support the "read all records" options in P2. It partially reads the record referenced by P1. The command data field shall contain an offset DO'54' indicating the first byte to be read in the record. The response data field shall contain a discretionary DO'53' encapsulating the data read.

Table 70 — READ RECORD (S) command-response pair

CLA	As defined in 5.4.1	
INS	'B2' or 'B3'	
P1	Record number or record identifier ('00' references the current record)	
P2	See Table 71	
L_c field	Absent for encoding $N_c = 0$, present for encoding $N_c > 0$	
Data field	INS = 'B2'	Absent
	INS = 'B3'	offset DO
L_e field	Present for encoding $N_e > 0$	

Data field	INS = 'B2'	Data read
	INS = 'B3'	Discretionary DO for encapsulating the data read
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6281, '6282', '6700', '6981', '6982', '6A81, '6A82', '6A83'	

Table 71 — Coding of P2 in the READ RECORD (S) command

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	x	x	x	-	-	-	Short EF identifier according to Table 69
-	-	-	-	-	0	x	x	**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
-	-	-	-	-	1	x	x	**Record number in P1**
-	-	-	-	-	1	0	0	— Read record P1
-	-	-	-	-	1	0	1	— Read all records from P1 up to the last (for INS = 'B2' only)
-	-	-	-	-	1	1	0	— Read all records from the last up to P1 (for INS = 'B2' only)
					1	1	1	RFU

If the L_e field contains only bytes set to '00', then the command should read completely either the single requested record, or the requested sequence of records, depending on bits 3, 2 and 1 of P2 and within the limit of 256 for a short L_e field, or 65 536 for an extended L_e field.

Case a — Partial read of one record (the L_e field does not contain only bytes set to '00')

T_n (one byte)	L_n (one or three bytes)	First bytes of V_n

<---------- N_r bytes ---------->

Case b — Complete read of one record (the L_e field contains only bytes set to '00')

T_n (one byte)	L_n (one or three bytes)	All the bytes of V_n

Case c — Partial read of a sequence of records (the L_e field does not contain only bytes set to '00')

T_n - L_n - V_n	…	T_{n+m} - L_{n+m} - V_{n+m} (First bytes of the record)

<---------- N_r bytes ---------->

Case d — Read several records up to the file end (the L_e field contains only bytes set to '00')

T_n - L_n - V_n	…	T_{n+m} - L_{n+m} - V_{n+m}

Figure 8 — Response data fields with INS = 'B2' when records are SIMPLE-TLV data objects

11.3.4 WRITE RECORD command

The command initiates one of the following operations into an EF according to the file attributes:

1) the write-once of a record given in the command data field (the command shall be aborted if the record is not in the logical erased state);
2) 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 data field;
3) 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 data field.

By default, i.e. when a data coding byte is not available (see 11.2.1), operation 1) shall apply in that EF.

If the record referenced by P1 and P2 is in the record LCS DEACTIVATED, the command is processed with warning '6287' without changing the record content.

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

If applied to an EF supporting a cyclic structure with records of fixed size, the “previous” option (bits 3, 2 and 1 of P2 set to 011) behaves as APPEND RECORD.

Table 72 — WRITE RECORD command-response pair

CLA	As defined in 5.4.1
INS	'D2'
P1	Record number ('00' references the current record)
P2	See Table 73
L_c field	Present for encoding $N_c > 0$
Data field	Record to be written
L_e field	Absent for encoding $N_e = 0$

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '63CX' (see 11.3.2), '6581', '6700', '6981', '6982', '6986', '6A81', '6A82', '6A83, '6A84', '6A85'

Table 73 — Coding of P2 in the WRITE RECORD command

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	x	x	x	-	-	-	Short EF identifier according to Table 69
-	-	-	-	-	0	x	x	**P1 set to '00'**
-	-	-	-	-	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 in P1**
— Any other value is RFU.								

If the records are SIMPLE-TLV data objects (see 6.1), then Figure 9 illustrates the command data field.

T_n (one byte)	L_n (one or three bytes)	All the bytes of V_n

Figure 9 — APDU data field (one complete record nesting a SIMPLE-TLV data object)

11.3.5 UPDATE RECORD command

The command initiates the update of a specific record with the bytes given in the command data field. When using current record addressing, the command shall set the record pointer on the updated record.

If applied to an EF supporting a linear or cyclic structure with records of fixed size, then the command shall be aborted if the record size after updating would be different from the size of the existing record.

If applied to an EF supporting a linear structure with records of variable size, then the command may be carried out when the record size after updating is different from the size of the existing record.

If applied to an EF supporting a cyclic structure with records of fixed size, the “previous” option (bits 3, 2 and 1 of P2 set to 011) behaves as APPEND RECORD.

If the record referenced by P1 and P2 is in the record LCS DEACTIVATED, the command is processed with warning '6287' without changing the record content.

If INS = 'DC' and if the records are SIMPLE-TLV data objects (see 6.1), then Figure 9 illustrates the command data field.

If INS = 'DD', then the command partially updates the record referenced by P1. The command data field shall contain an offset DO'54' for indicating the first byte to be updated in the record and a DO'53' ('73' is deprecated for this use) for encapsulating the updating data.

Table 74 — UPDATE RECORD command-response pair

CLA	As defined in 5.4.1	
INS	'DC' or 'DD'	
P1	Record number ('00' references the current record)	
P2	See Table 73 (INS = 'DC') or Table 75 (INS = 'DD')	
L_c field	Present for encoding $N_c > 0$	
Data field	INS = 'DC'	Updating data
	INS = 'DD'	Offset DO and discretionary DO for encapsulating the updating data
L_e field	Absent for encoding $N_e = 0$	

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '63CX' (see 11.3.2), '6581', '6700', '6981', '6982', '6986', '6A81', '6A82', '6A83, '6A84', '6A85'

Table 75 — Coding of P2 in the UPDATE RECORD command with INS = 'DD'

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	x	x	x	-	-	-	Short EF identifier according to Table 69
-	-	-	-	-	1	x	x	**Record number in P1**
-	-	-	-	-	1	0	0	— Replace
-	-	-	-	-	1	0	1	— Logical AND
-	-	-	-	-	1	1	0	— Logical OR
-	-	-	-	-	1	1	1	— Logical XOR
— Any other value is RFU.								

11.3.6 APPEND RECORD command

The command initiates either the writing of a new record at the end of a linear structure, or at the beginning of a cyclic structure. When using current record addressing, the command shall set the record pointer on the successfully appended record.

If the command applies to a linear structure full of records, then the command is aborted because there is not enough memory space in the file.

If the command applies to a cyclic structure full of records, then the record with the highest record number is deleted. All other record numbers are incremented by one. The appended record becomes record number one. If the record with the highest record number is in the record LCS DEACTIVATED the command is processed with warning '6287' without changing any record content or record number.

If the records in the EF have record life cycle, the LCS of the appended record shall be set to ACTIVATED unless otherwise specified.

If the records are SIMPLE-TLV data objects (see 6.1), then Figure 9 illustrates the command data field.

Table 76 — APPEND RECORD command-response pair

CLA	As defined in 5.4.1
INS	'E2'
P1	'00' (any other value is invalid)
P2	See Table 73 with bits 3 to 1 set to 000 (any other value is RFU)
L_c field	Present for encoding $N_c > 0$
Data field	Record to be appended
L_e field	Absent for encoding $N_e = 0$

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '63CX' (see 11.3.2), '6287' 6581', '6700', '6981', '6982', '6986', '6A81', '6A82', '6A83, '6A84', '6A85'

11.3.7 SEARCH RECORD command

The command initiates a simple or enhanced or proprietary search on records stored within an EF. The search can be limited to records with a given identifier or to records with a number greater or smaller than a given number. It can be performed in increasing or in decreasing order of record numbers. The search starts either from the first byte of the records (simple search), or from a given offset within the records (enhanced search), or from the first occurrence of a given byte within the records (enhanced search). The response data field gives the numbers of the records matching the search criteria within an EF supporting records. The command shall set the record pointer on the first record matching the search criteria.

In an EF supporting records of variable size with linear structure, the search shall not take into account the records shorter than the search string. In an EF supporting records of fixed size with linear or cyclic structure, if the search string is longer than the records, then the card shall abort the command.

Records with a record LCS set to DEACTIVATED shall be ignored during the search.

Table 77 — SEARCH RECORD command-response pair

CLA	As defined in 5.4.1	
INS	'A2'	
P1	Record number or record identifier ('00' references the current record)	
P2	See Table 78	
L_c field	Present for encoding $N_c > 0$	
Data field	bits 3 and 2 of P2 not set to 11, simple search	Search string
	bits 3, 2 and 1 of P2 set to 110, enhanced search	Search indication (2 bytes, see Table 79) followed by search string
	bits 3, 2 and 1 of P2 set to 111, proprietary search	Proprietary
L_e field	Absent for encoding $N_e = 0$, present for encoding $N_e > 0$	

Data field	Absent or record number(s)
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6282', '6982'
— The response data field is absent either because the L_e field is absent, or because no match is found. — The response data field does not give record identifiers because they may not be unique.	

In an enhanced search (bits 3, 2 and 1 of P2 set to 110), the command data field consists of a search indication on two bytes followed by a search string. Table 79 specifies the first search indication byte. According to the first byte of the search indication, the second byte is either an offset or a value, i.e. the search in the records shall start either from this offset (see 11.3.2) or after the first occurrence of this value.

Table 78 — Coding of P2 in the SEARCH RECORD command

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	x	x	x	-	-	-	Short EF identifier according to Table 69
-	-	-	-	-	0	**x**	**x**	**Simple search with record identifier in P1**
-	-	-	-	-	0	0	0	— Forward from first occurrence
-	-	-	-	-	0	0	1	— Backward from last occurrence
-	-	-	-	-	0	1	0	— Forward from next occurrence
-	-	-	-	-	0	1	1	— Backward from previous occurrence
-	-	-	-	-	1	0	x	**Simple search with record number in P1**
-	-	-	-	-	1	0	0	— Forward from P1
-	-	-	-	-	1	0	1	— Backward from P1
-	-	-	-	-	1	1	0	**Enhanced search**
-	-	-	-	-	1	1	1	**Proprietary search**

Table 79 — Coding of the first byte of the search indication

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	-	-	-	The subsequent byte is an offset (start from that position)
0	0	0	0	1	-	-	-	The subsequent byte is a value (start after the first occurrence)
-	-	-	-	-	0	x	x	**Record identifier in P1**
-	-	-	-	-	0	0	0	— Forward from first occurrence
-	-	-	-	-	0	0	1	— Backward from last occurrence
-	-	-	-	-	0	1	0	— Forward from next occurrence
-	-	-	-	-	0	1	1	— Backward from previous occurrence
-	-	-	-	-	1	x	x	**Record number in P1**
-	-	-	-	-	1	0	0	— Forward from P1
-	-	-	-	-	1	0	1	— Backward from P1
-	-	-	-	-	1	1	0	— Forward from next record
-	-	-	-	-	1	1	1	— Backward from previous record
— Any other value is RFU.								

11.3.8 ERASE RECORD (S) command

The command sets one or more records of an EF to the logical erased state, either the record referenced by P1, or the sequence of records from P1, sequentially, up to the end of the file. Erased records shall not be deleted, and may still be accessible by WRITE RECORD and UPDATE RECORD commands.

If any record referenced by P1 and P2 is in record LCS DEACTIVATED, the command is processed with warning '6287' without changing any record content.

Table 80 — ERASE RECORD (S) command-response pair

CLA	As defined in 5.4.1
INS	'0C'
P1	Record number ('00' references the current record)
P2	See Table 81
L_c field	Absent for encoding $N_c = 0$
Data field	Absent
L_e field	Absent for encoding $N_e = 0$

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6287', '63CX' (see 11.3.2), '6581', '6700', '6981', '6982', '6986', '6A81', '6A82', '6A83, '6A84', '6A85'

Table 81 — Coding of P2 in the ERASE RECORD (S) command

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	x	x	x	-	-	-	Short EF identifier according to Table 69
-	-	-	-	-	1	x	x	**Record number in P1**
-	-	-	-	-	1	0	0	— Erase record P1
-	-	-	-	-	1	0	1	— Erase all records from P1 up to the last
— Any other value is RFU.								

11.3.9 ACTIVATE RECORD (s) command

The command sets the record(s) referenced by P1 and P2 to the record LCS ACTIVATED. The command shall not affect the record pointer.

If the EF referenced by P2 does not support record life cycle, the command shall be aborted with the status bytes '6981'.

If an addressed record is already activated, the command shall return the status bytes '9000'.

For the activation of all records in the record LCS DEACTIVATED, the ACTIVATE FILE command may be used. Independent from the modification of an optionally present file LCS (see 7.4.10) all records will be activated.

Table 82 — ACTIVATE RECORD (s) command-response pair

CLA	As defined in 5.4.1
INS	'08'
P1	Record number ('00' references the current record)
P2	See Table 83
L_c field	Absent
Data field	Absent
L_e field	Absent

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6981', '6982', '6986', '6A82', '6A83'

Table 83 — Coding of P2 in ACTIVATE RECORD(s) or DEACTIVATE RECORD(s) commands

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	x	x	x	–	–	–	Short EF identifier according to Table 69
–	–	–	–	–	1	x	x	Record number in P1
–	–	–	–	–	1	0	0	– Activate or deactivate record P1
–	–	–	–	–	1	0	1	– Activate or deactivate all records from P1 to the last
— Any other value is RFU.								

11.3.10 DEACTIVATE RECORD (s) command

The command sets the record(s) referenced by P1 and P2 to the record LCS DEACTIVATED. The command shall not affect the record pointer.

If the EF referenced by P2 does not support record life cycle, the command shall be aborted with the status bytes '6981'.

If an addressed record is already deactivated, the command shall return the status bytes '9000'.

Table 84 — DEACTIVATE RECORD command-response pair

CLA	As defined in 5.4.1
INS	'06'
P1	Record number ('00' references the current record)
P2	See Table 83
L_c field	Absent
Data field	Absent
L_e field	Absent

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6981', '6982', '6986', '6A82', '6A83'

11.3.11 COMPARE RECORD function

This function is supported by the COMPARE command (see 11.6.1).

11.4 Data object handling

11.4.1 General

In this group of commands, SW1-SW2 set to '63CX' indicates a successful change of memory state, but after an internal retry routine; 'X' > '0' encodes the number of retries; 'X' = '0' means that no counter is provided.

11.4.1.1 Coding of P1-P2 for even INS codes

Table 85 — Coding of P1-P2 for handling data objects with even INS codes

Value of P1-P2	Meaning
'0000'	Dump a file (see 12.4)
	Retrieve a card-originated query from a card or send a reply to the card (see 12.5)
'0001' to '00FE'	BER-TLV tag (one byte) in P2
'00FF'	Special function (see Table 89, Table 91, Table 92 and Table 93)
'0100' to '01FF'	Proprietary
'0200'	RFU
'0201' to '02FE'	SIMPLE-TLV tag in P2
'02FF'	Special function (see 11.4.1.1)
'1F1F' to 'FFFF'	BER-TLV tag (two bytes) in P1-P2
— Invalid BER-TLV tags in the range '01' to 'FE' and '1F1F' to 'FFFF' are RFU. — Any other value is RFU.	

If P1 is set to '00', then P2 from '01' to 'FE' shall be a BER-TLV tag on a single byte.

If P1 is set to '01', then P2 from '00' to 'FF' shall be an identifier for card internal tests and for proprietary services meaningful within a given application.

If P1 is set to '02', then P2 from '01' to 'FE' shall be a SIMPLE-TLV tag. The value '0200' is RFU. The value '02FF' is used either for obtaining all the common SIMPLE-TLV data objects readable in the context or for indicating that the command data field is encoded in SIMPLE-TLV.

If P1-P2 lies from '1F1F' to 'FFFF', then they shall only encode a valid BER-TLV tag on two bytes.

NOTE Many values in the range '1F1F' to 'FFFF' are not valid BER-TLV tags (see Annex E).

11.4.1.2 Coding of P1-P2 for odd INS codes

If bit b1 of INS is set to 1, then, except for the SELECT DATA command, P1-P2 not set to '0000' or 'FFFF' shall identify a file:

— If the first eleven bits of P1-P2 are set to 0 and if bits b5 to b1 of P2 are not all equal and if the card and/or the file supports selection by short EF identifier, then bits b5 to b1 of P2 encode a short EF identifier (a number from one to thirty). Otherwise, P1-P2 is a 2-byte file identifier.

— P1-P2 set to '3FFF' identifies the current DF.

P1-P2 set to '0000' identifies the current EF, unless the command data field provides a file reference DO'51' for identifying a file.

P1-P2 set to 'FFFF' identifies the current template, unless the command data field provides referencing of another template (see arguments 1, 2 and 3 in Table 86).

11.4.1.3 Data fields

If bit b1 of INS is set to 0, if a DO is requested or provided within the current template, then the payload shall contain the value field of the data object, i.e., either the referred data element in the case of a SIMPLE-TLV data object or a primitive BER-TLV data object, or the referred template in the case of a constructed DO.

Irrespective of bit b1 in INS, if a set of DOs is provided or if the content of an EF is requested, then the appropriate data field shall contain the DO(s).

11.4.1.4 Access to the extension of the current template

When a template is extended by tagged wrappers (see 8.4.8), this extension is only valid for GET DATA and GET NEXT DATA commands. All other commands handling DOs by their tag only are limited to the base template. Automatic resolution of tagged wrappers (see 8.4.8) shall not modify the current template.

If the template contains one or several wrapper(s), 8.4.8 describes how to recover DOs expected to be relevant to the template.

11.4.1.5 Execution or rejection conditions

Any command of this group shall be aborted if the parameters which select a DO to be accessed do not match the actual structures within the card, e.g.:

— if applied to a structure (DF or EF) not supporting data objects.

— if the parameters do not match the actual data object structure

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

11.4.2 SELECT DATA command

11.4.2.1 General

Provided that it does not modify the LCS of a DO, the SELECT DATA is always granted.

The functions of this command are

— to set a unique selected DO, i.e. the target of the command, as current DO;

— to set a current template as the value field of the selected DO if this DO is constructed;

— to set reference data to be compared e.g. with comparison data by a subsequent COMPARE command (see 11.6.1);

— to modify the VA (see 7.2.1) if the DO to be selected does not belong to the current VA of a current template. If the command transmits

  — an application identifier DO'4F', and no file reference DO'51', it selects an application DF (see 7.2.2 rule d)).

  — no DO'4F', but a DO'51', it selects a file (see 7.2.2 rule e) or f)).

  — a DO'4F' and a DO'51', it selects an application DF, (see 7.2.2 rule d)), and a file (see 7.2.2 rule e) or f)).

  — a record number (see argument 3 in Table 86) it modifies curRecord (see 7.2.2 rule g)

  — an offset (see argument 3 in Table 86), it modifies curDataString (see 7.2.2 rule h)

Table 86 — Value of the general reference DO'60' (General reference template)

Arg.	Nested DOs		Comment	
1	application identifier DO'4F'		Optional	
2	file reference DO'51' (see Table 7)		Optional, even number of bytes. Valid if the path matches the file structure of the current application, or of the application referenced by a DO'4F'.	
3	Choice between	Record number DO'02' followed by Length DO'02'	Optional. References a record or a virtual DO'7F70' the value of which is a record in an EF	
		Offset DO'54' followed by Length DO'02'	Optional. References a DataString or a virtual DO'7F70' the value of which is a DataString in a transparent EF.	
4	Choice between	tag list DO'5C'	Nests one tag present in the current template.	Argument 4, mandatory in DO referencing. Applies to the current template or to the transient current template set by the previous arguments
		extended header DO'4D' or '5F61' (see 8.4.5 for the rationale of a choice)	The value of the extended header starts with a tag present in the current template	
		masked tag DO'5F8400'	References a DO by partial tag.	
		filter DO'7F71'	References a constructed DO by its content	
5 …	arbitrary number of masked tag DO'5F8400' and/or filter DO'7F71' in any order; the result of applying a mask or filter to a set of DO is a subset; the result of applying all masks and all filters is the intersection of all of those subsets		DO'5F8400' references a DO by partial tag; DO'7F71' references a constructed DO by its content.	Optional if the DO referenced by argument 4 is constructed.

Table 87 — SELECT DATA command-response pair

CLA	As defined in 5.4.1	
INS	'A5'	
P1	< 'F0'	Occurrence number of an instance.
	'F0'	Select the parent of the DO referenced by curConstructedDO
	> 'F0'	RFU
P2	See Table 88	
L_c field	Absent for encoding $N_c = 0$, present for encoding $N_c > 0$	
Data field	May be empty if P1='F0'. In other cases at least argument 4 of a general reference DO'60' (see Table 86). Other arguments are optional. All arguments shall be in the same order as in the general reference template.	
L_e field	Absent for encoding $N_e = 0$, present for encoding $N_e > 0$ if response data are required by P2 (see Table 88)	

Data field	Absent or information according to P2.
SW1-SW2	See Table 5 and Table 6 when relevant, e.g., '6202' to '6280', '6281', '6700', '6981', '6982', '6985', '6A81', '6A88' (DOs not found, i.e., referenced data not found)

Table 88 — Coding of P2 in SELECT DATA command

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	-	-	-	**x**	**x**	**File or DO occurrence in the current template**
0	0	0	-	-	-	0	0	— First or only occurrence of a DO after skipping P1 occurrences
0	0	0	-	-	-	0	1	— Last occurrence of a DO after skipping P1 occurrences
0	0	0	-	-	-	1	0	— Next occurrence of a DO after skipping P1 occurrences
0	0	0	-	-	-	1	1	— Previous occurrence of a DO after skipping P1 occurrences
0	0	0	**x**	**x**	**x**	-	-	**Response data field requirements (see 7.4.2 and Table 8)**
0	0	0	0	0	1	-	-	Return the data control information (DO'62')
0	0	0	0	1	0	-	-	Return the tag list of the current template (DO'5C') (DIR function)
0	0	0	1	0	0	-	-	Return the tag list of the current template (DO'5C') (VIEW function)
0	0	0	1	1	0	-	-	Return the tag list of the current template (DO'5C') (DIR function) followed by the tag list of the current template (DO'5C') (VIEW function)
— Any other value is RFU.								

Except when P1 is set to 'F0', processing this command involves a sequence of searches for a match within transient current templates. The search for the last match will take place:

— in the last transient current constructed DO (highest generation) defined by the arguments, when the last tag references either a primitive DO, or a constructed DO with the options "first" or "last". If a match is found, the selected DO is a child of the last transient constructed DO;

— in the parent DO of the last transient current constructed DO (highest but one generation), when the last tag references a constructed DO with the options "next" or "previous". If a match is found, the selected DO is a sibling of the last transient constructed DO;

In the following examples, "transient current template" denotes the value field of the transient current constructed DO in which the last match will take place:

P1-P2='0000' selects the first occurrence of a DO within the transient current template.
P1-P2='0100' selects the second occurrence of a DO within the transient current template.
P1-P2='0101' selects the penultimate occurrence of a DO within the transient current template.
P1-P2='0001' selects the last occurrence of a DO within the transient current template.
P1-P2='0102' selects either the second next occurrence of a primitive DO within the transient current template or the second next sibling of the current constructed DO.

A failed SELECT DATA C-RP (SW1>'62', see Table 6) does not modify the current VA.

11.4.2.2 Basic target definition

Basic target definition uses arguments 1, 2 and 4. If the target, i.e. the DO to be selected:

— is not contained in the current application, the data field shall include an application identifier DO'4F',

— is not contained in the current file after possible transient application selection, the data field shall include a file reference DO'51' when file referencing applies in the application.

The tag of the target shall be:

— either encapsulated in a tag list DO'5C'

— or defined by an extender header list DO'4D' or '5F61' (see 8.4.5 for the reasons for a choice)

— or defined by a masked tag DO'5F8400'

— or defined by the content of a constructed DO by a filter DO'7F71'

Conditional argument 3 references a virtual DO'7F70' the value of which is either a record in an EF structured in records, or a DataString in a transparent EF. The transient current DO'7F70' shall become the current constructed DO if the mandatory argument does not reference a constructed DO.

11.4.2.3 Referencing by masked tag DO and filter DO

If the command references one constructed DO, setting a current template, a conditional argument 5, masked tag DO'5F8400' or filter DO'7F71' may be present.

If the masked tag DO'5F8400' is present, the command should return a tag list DO'5C' nesting the concatenation of all tags of the template matching the masked tag. A masked tag nests two data elements of the same length, i.e. <mask value> followed by <target tag>. A match with tag <matching tag> happens when:

<mask value> AND <matching tag> = <mask value> AND <target tag>

If the filter DO'7F71' is present, the command shall succeed if and only if the value of the filter DO'7F71' is:

— either a constructed DO belonging to the transient current template; this DO becomes current

— or a sub-tree (see 8.2.3) of a constructed DO belonging to the transient current template; this DO becomes current.

11.4.2.4 GET DATA CONTROL PARAMETERS function

When required by bit b3 set to 1 in P2, the response payload is a DO'62' (see Table 10) relevant to the (transiently or finally) current DO, after possible modification of the data LCS. If DO'62' is not available, CP DOs may be present under a tag reserved for the tag allocation authority (see 8.3.4).

11.4.2.5 DIR function

When required by bit b4 set to 1 in P2, the command should return a tag list DO'5C' nesting the concatenation of all tags of the current template (possibly including a template extension) set by the command.

An actual tag list DO may either be present in the template, or dynamically generated by the implementation. In both cases,

— The tags shall be present irrespective of the security attributes or value of data LCS of the corresponding DO.

— if several instances of the same DO are present, the tag shall be repeated.

— if present, wrapper DOs shall always appear.

If the implementation resolves the indirections, the local tags of the DOs defined in the wrappers may appear in the tag list (which displays the extended template). If not, they shall not (the tag list displays the base template).

11.4.2.6 VIEW function

When required by bit b5 set to 1 in P2, the command should return a tag list DO'5C' nesting the same concatenation of tags as in the DIR function, but, according to its needs, an application may exclude the tags of e.g.:

— DOs not readable under the current security status ;

— DOs not in activated state ;

— tagged wrappers when automatic resolving of those is granted (see 8.4.8 and 11.4.2.6).

11.4.2.7 File-related functions

When argument 4 is absent in the command data field, and argument 3 is present, a successful SELECT DATA command shall

— set curRecord in case argument 3 contains a record number, or

— set the DataString referenced by argument 3 as curDataString in case argument 3 contains an offset DO.

When arguments 3 and 4 are absent in the command data field, arguments 1 and 2 support the selection of an application, of a file in the current application, or file within a given application.

11.4.3 GET DATA/GET NEXT DATA commands - even INS code

Table 89 — GET DATA/GET NEXT DATA command-response pair (even INS codes)

CLA	As defined in 5.4.1	
INS	'CA'	GET DATA
	'CC'	GET NEXT DATA
P1-P2	See Table 85, if the special value '00FF' is used the command obtains all DOs from the current template	
L_c field	Absent for encoding $N_c = 0$	
Data field	Absent	
L_e field	Present for encoding $N_e > 0$,	

Data field	0, 1 or more data bytes according to P1-P2.
SW1-SW2	See Table 5 and Table 6 when relevant, e.g., '6202' to '6280', '6281', '6700', '6981', '6982', '6985', '6A81', '6A88' (data objects not found, i.e., referenced data not found)

11.4.3.1 General

The main function of those commands with even INS code is the retrieval of the value field of a DO belonging to the current template. It may be the content of an EF supporting DOs.

If there are several occurrences of a tag in the current template, this clause does not define which DO is returned by a successful GET DATA because that depends on the definition or the nature or the content of the DO. GET NEXT DATA has a precise behaviour (see 11.4.3.4).

A required DO or data element shall be absent from the response when the security status does not match its security attributes.

11.4.3.2 SELECT function

After a successful GET DATA command of a constructed DO, this DO becomes the current constructed DO.

A successful GET NEXT DATA command shall impact neither the current DO, nor the current template.

11.4.3.3 GET DATA CONTROL PARAMETERS function

When the argument of an even INS GET DATA/GET NEXT DATA is the CP template tag '62', the response payload is a concatenation of CP DOs (see Table 10) attached to the DOs in the same template. If DO'62' is not available, CP DOs may be present under a tag reserved for the tag allocation authority (see 8.4.4).

11.4.3.4 Specific functions of GET NEXT DATA and pointer handling

When there are several occurrences of the same tag in the template, successive GET NEXT DATA commands shall return their values sequentially. The order in which the values are recovered shall be the same as the order of those tags recovered by a DIR or VIEW function (see 11.4.2.5 or 11.4.2.6).

Contrary to GET DATA, a successful GET NEXT DATA command shall not impact the VA. The handling of multiple instances implies that

— a template is seen as an ordered list of DOs at the interface between a card and an interface device. The term "ordered list" implies that this list has a first and a last element. Each list element, except the last, has a next element. Each list element, except the first, has a previous element.

— a pointer is attached to a logical channel. The default value of this pointer is to be unset. The pointer is set by a GET NEXT DATA, PUT DATA, PUT NEXT DATA or UPDATE DATA command (see 11.4.6, 11.4.7 and 11.4.8). This pointer shall be unset if any command different from GET NEXT DATA or PUT NEXT DATA is transmitted on the same logical channel. The transmission of a command on a logical channel should not impact the pointer on another logical channel.

When a sequence of DOs supports cyclic management, a GET DATA or GET NEXT DATA command shall recover first the most recent instance. A further GET NEXT DATA shall recover the remaining most recent instance, etc…

When all data elements or DOs have been transmitted by the card in successive GET NEXT DATA responses, a further GET NEXT DATA shall be rejected by SW1-SW2 = '6A88'.

11.4.4 GET DATA/GET NEXT DATA command - odd INS codes

11.4.4.1 General

The main function of those commands with odd INS code is the retrieval of the value field of one or several DOs according to the arguments of the command. The target selection is very close to the SELECT DATA selection, except that:

— the command may return several DOs, whereas a SELECT DATA selects a single DO.

— Argument 2 of the SELECT DATA may be replaced by a file identifier in P1-P2, functionally equivalent to a file reference DO containing one file identifier. If this option is used, the command data field shall not feature argument 1 (DO'4F').

When a GET DATA/GET NEXT DATA requires one (or several) constructed DO(s), all DOs nested within shall be present in the response, except:

1) when no DO with the required tag is available in the template,

2) when the security status does not match its security attributes,

3) if not explicitly required when the argument of the command is an extended header list

A GET DATA/GET NEXT DATA command may not be rejected if one or several required DOs are unavailable for one of those reasons. It shall be rejected only if no DO is available.

If there are several occurrences of a tag in the current template, this clause does not define which DO is returned by a successful GET DATA because that depends on the definition or the nature or the content of the DO. GET NEXT DATA has a precise behaviour (see 11.4.3.4).

Table 90 — GET DATA/GET NEXT DATA command-response pair (odd INS codes)

CLA	As defined in 5.4.1	
INS	'CB'	GET DATA
	'CD'	GET NEXT DATA
P1-P2	'0000'	Current file, except if the data field references a file.
	'FFFF'	Current template or transient current template possibly set by the data field
	other values	File identifier or short EF identifier (see 11.4.1.2) The data field contains neither an application identifier DO'4F' nor a file reference DO'51'.
L_c field	Present for encoding $N_c > 0$	
Data field	Identical to data field of SELECT DATA, (see Table 86 and Table 87) except that it may reference several DOs	
L_e field	Present for encoding $N_e > 0$	

Data field	0, 1 or more data bytes
SW1-SW2	See Table 5 and Table 6 when relevant, e.g., '6202' to '6280', '6281', '6700', '6981', '6982', '6985', '6A81', '6A88' (data objects not found, i.e., referenced data not found)

11.4.4.2 Mandatory argument

The command data field shall end with either a tag list DO'5C' (see 8.4.3) or an extended header list DO'4D' or DO'5F61' (see 8.4.5), a masked DO'5F8400' or a filter DO'7F71' (see 11.4.2.3).

— In the tag list case, the response data field shall be the concatenation of the DOs referenced in the tag list, in the same order. The order may be different from the order of the DOs in the template. When several DOs within the template have the same tag, all those DOs shall be returned. One or more DOs may be absent for security status reasons. An empty tag list requires all the available DOs.

— In the extended header list case, the response data field shall be the concatenation of the DOs derived from the extended header list according to 8.4.6 and 8.4.7.

— In the masked tag case or in the filter case, the response data field shall be the concatenation of the matching DOs.

11.4.4.3 SELECT function

If a GET DATA command with P1-P2 not equal to 'FFFF' references a file, the side effect of its success shall be to set this file as current and the current template as value field of the root DO'7F70'.

A successful GET DATA with a syntax not defined in the 2nd edition of this document shall not impact the VA, i.e.:

— P1-P2 set to 'FFFF'

— Argument 1 and/or argument 3 present,

— Argument 4 different from DO'4D' or DO'5C'.

11.4.4.4 DIR and VIEW functions

The DIR/VIEW function is supported by an odd INS GET DATA/GET NEXT DATA when its argument is:

— '5C 01 5C' (tag '5C' nested in a tag list DO); the response payload should be a tag list DO'5C' nesting the response payload defined in 11.4.2.5 or 11.4.2.6.

— '5C 01 5D' (tag'5D' nested in a tag list DO) the response payload should be a header list DO'5D' nesting the response payload defined in 11.4.2.5 or 11.4.2.6.

The choice between the DIR and VIEW function is out of scope of this document.

11.4.4.5 GET DATA CONTROL PARAMETERS function

When the argument of an odd INS GET DATA is '5C 01 62' (tag '62' (CP template tag) nested in a tag list DO), the response payload should be a DO'62'.

11.4.4.6 Specific functions of GET NEXT DATA (INS = 'CD')

The properties of GET NEXT DATA (INS='CD') with respect to GET DATA (INS='CB') are identical to those of GET NEXT DATA (INS='CC') with respect to GET DATA (INS='CA', see 11.4.3.4), except that the C-RPs with odd INS codes return DOs, the C-RPs with even INS codes return data elements (values of DOs).

11.4.5 General properties of PUT/PUT NEXT/UPDATE DATA commands

Those commands initiate the modification of the current template contents, by transmitting one or several DO(s), possibly constructed. PUT/PUT NEXT/UPDATE DATA with bit b1 of INS:

— set to 0 shall only operate in the current template.

— set to 1 sets the transient template as the value of the virtual root DO'7F70', selected according to P1-P2 not equal to 'FFFF', before the actual handling of the DO(s). The transient current template shall become the current template if the command succeeds.

If the pointer is set by a successful GET NEXT DATA or PUT NEXT DATA C-RP (see 11.4.3.4) it defines which instance shall be affected by an UPDATE DATA or PUT DATA C-RP.

If the pointer is unset (see 11.4.3.4) curDO (see 7.2.1) defines which DO is affected by an UPDATE DATA or PUT DATA C-RP.

Whereas PUT DATA and UPDATE DATA commands may include several DOs in the command data field, a PUT NEXT DATA command shall feature one and only one DO in the command data field.

11.4.6 PUT DATA command

PUT DATA may have the behaviour and codings defined for UPDATE DATA (resp PUT NEXT DATA). Consistently with the 2nd edition of this document, the definition or the nature or the content of the DOs shall induce the impact on the template content. If INS is set to 'DB' and P1-P2 is set to 'FFFF', the following rules apply:

— If PUT NEXT DATA is supported, a PUT DATA of a DO the tag of which already exists in the template shall replace an existing DO by a new one.

— If UPDATE DATA is supported, a PUT DATA shall ensure that the whole transmitted DO is added to the template. This may result in a duplication of instances within the template.

When the L_c field is absent, the even INS PUT DATA command shall add an empty DO or replace an existing DO by an empty DO of the same tag as indicated by P1-P2.

Table 91 — PUT DATA command-response pair

CLA	As defined in 5.4.1	
INS	'DA' or 'DB'	
P1-P2	INS = 'DA'	See Table 85, if the special value '00FF' is used the command data field contains a concatenation of DOs to be added to or replaced in the current template
	INS = 'DB'	File identifier or short EF identifier (see 11.4.1.2)
L_c field	Present for encoding $N_c > 0$, absent for encoding $N_c = 0$	
Data field	INS = 'DA'	Data bytes according to P1-P2, or absent to delete the value of a DO.
	INS = 'DB'	Concatenation of DOs
L_e field	Absent for encoding $N_e = 0$	

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g., '63CX' (see 11.4.1), '6581', '6700', '6981', '6982', '6985', '6A80', '6A81', '6A84', '6A85'

11.4.7 PUT NEXT DATA command

The command initiates the addition of one DO within the current template. If a DO with the same tag as a DO to be inserted already exists in the template, it shall add this new instance of the DO within the template. If the number of instances has not reached its maximum value, PUT NEXT DATA shall add the transmitted DO to the current template, and set the pointer on the DO which has been put. The outcome of this command depends on the structure of the DO as given by the data descriptor byte (see Table 13). If the structure is of type

- "no information given" and the number of instances has reached its maximum value, PUT NEXT DATA shall be rejected with SW1-SW2='6A84". Otherwise the new DO is inserted at an arbitrary position in the ordered list of DO.
- "linear management" and the number of instances has reached its maximum value, PUT NEXT DATA shall be rejected with SW1-SW2='6A84". Otherwise if the pointer is
  - unset, then the new DO shall be appended after the last element of the list.
  - set, then the new DO shall be inserted such that it becomes the previous list element with respect to the pointed DO.
- "cyclic management" and the pointer is
  - unset, then the new DO shall become the first element of the list.
  - set, then the new DO shall be inserted such that it becomes the next list element with respect to the pointed DO.

Furthermore, if the structure is of type "cyclic management" and after inserting the new DO into the ordered list the number of instances is greater than the maximum value then the DO nearest to the end of the ordered list having the same tag as the inserted one shall be deleted.

NOTE 1 The rules for insertion of a new DO are such that with linear and cyclic management a new DO can be inserted at any position in the ordered list.

NOTE 2 If the pointer is unset then the cyclic management behaves as the APPEND RECORD for EFs with cyclic structure.

NOTE 3 If the structure is of cyclic management and the pointer points to the last element and the number of instances has reached its maximum value then the rules imply that inserting a new DO has no effect to the ordered list.

NOTE 4 If the instances are explicitly numbered, the handling of the instance number (see Table 10) has to be dynamic.

NOTE 5 This version of the standard does not define how the maximum number of instances could be seen at the card's interface.

If several DOs are transmitted, the command shall be aborted (SW1-SW2 = '6A80'), without changing anything in the current template.

Table 92 — PUT NEXT DATA command-response pair

CLA	As defined in 5.4.1	
INS	'D8' or 'D9'	
P1-P2	INS = 'D8'	See Table 85, if the special value '00FF' is used the command data field contains a DO to be added to the current template
	INS = 'D9'	File identifier or short EF identifier (see 11.4.1.2)
L_c field	Present for encoding $N_c > 0$	
Data field	INS = 'D8'	Data bytes according to P1-P2
	INS = 'D9'	One DO
L_e field	Absent for encoding $N_e = 0$	

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g., '63CX' (see 11.4.1), '6581', '6700', '6981', '6982', '6985', '6A80', '6A81', '6A84', '6A85'

11.4.8 UPDATE DATA command

Table 93 — UPDATE DATA command-response pair

CLA	As defined in 5.4.1	
INS	'DE' or 'DF'	
P1-P2	INS = 'DE'	See Table 85, if the special value '00FF' is used the command data field contains a concatenation of DOs to be processed within the current template
	INS = 'DF'	File identifier or short EF identifier (see 11.4.1.2)
L_c field	Present for encoding $N_c > 0$, absent for encoding $N_c = 0$ (see below)	
Data field	INS = 'DE'	Data bytes according to P1-P2, or absent to delete the value of a DO
	INS = 'DF'	One DO (possibly constructed)
L_e field	Absent for encoding $N_e = 0$	

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g., '63CX' (see 11.4.1), '6581', '6700', '6981', '6982', '6985', '6A80', '6A81', '6A84', '6A85'

For each DO indicated in the data field, the command shall perform:

a) either the modification of a DO, within the current template. If several instances of a DO exist, and:

   1) one of them is the current DO, it shall be updated.

   2) if the current DO is not one of the instances, this document does not define which instance of the DO is updated.

b) or the deletion of a DO's value field, with the same conditions as above when several instances exist. Updating a non-empty DO with an empty DO replaces the existing DO by an empty one.

c) or the creation of a DO, within the current template, if no DO with the same tag already exists.

Updating a primitive DO replaces the existing DO by the transmitted DO.

To update a constructed DO (see F.3), all templates present in the transmitted DO shall be processed successively, starting with the lowest generation number. The DOs already present in the template shall be modified, and the DOs not present in the template shall be created in the template. If one or several DOs to be modified are constructed, the procedure shall be repeated at the next generation, and so on.

NOTE This updates DOs within a constructed DO without the need to retransmit the whole constructed DO.

11.4.9 COMPARE DATA function

This function is supported by the COMPARE command (see 11.6.1).

11.5 Basic security handling

11.5.1 General

The security-related procedures, supported by the commands described in this clause, often involve an ordered sequence including those commands, and commands described in ISO/IEC 7816-8. The use of security attribute extensions (see 9.3.6.2) supports the description of such sequences at the interface.

The commands of this group reserve P1-P2 for referencing an algorithm and some related reference data (e.g. a key). If there is a current key and a current algorithm, then the command may implicitly use them.

P1 — Unless otherwise specified, P1 references the algorithm to use: either a cryptographic algorithm or a biometric algorithm (see ISO/IEC 7816-11). P1 set to '00' means that no information is given, i.e. either the reference is known before issuing the command, or the command data field provides it.

P2 — Unless otherwise specified, P2 qualifies reference data according to Table 94. P2 set to '00' means that no information is given, i.e. either the qualifier is known before issuing the command, or the command data field provides it. The qualifier may be for example a password number or a key number or a short EF identifier.

Table 94 — Coding of the reference data qualifier in P2

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	0	0	0	No information given
0	-	-	-	-	-	-	-	Global reference data (e.g. MF specific password or key)
1	-	-	-	-	-	-	-	Specific reference data (e.g. DF specific password or key)
-	x	x	-	-	-	-	-	00 (any other value is RFU)
-	-	-	x	x	x	x	x	Qualifier, i.e. number of the reference data or number of the secret

NOTE A MANAGE SECURITY ENVIRONMENT command may set an algorithm reference and/or a multi-byte reference data qualifier (see Table 51).

In this group of commands, SW1-SW2 set to '6300' or '63CX' indicates that the verification failed, 'X' >= '0' encodes the number of further allowed retries. SW1-SW2 set to '6A88' means “reference data not found”.

11.5.2 INTERNAL AUTHENTICATE command

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

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

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

Any successful authentication may be subject to completion of prior commands (e.g. VERIFY, SELECT) or selections (e.g. the relevant secret).

The card may record the number of times the command is issued, in order to limit the number of further uses of the relevant secret or the algorithm.

NOTE The response data field may include data useful for further security functions (e.g. random number).

Table 95 — INTERNAL AUTHENTICATE command-response pair

CLA	As defined in 5.4.1
INS	'88'
P1-P2	See 11.5.1 and Table 94
L_c field	Present for encoding $N_c > 0$
Data field	Authentication-related data (e.g. challenge)
L_e field	Present for encoding $N_e > 0$

Data field	Authentication-related data (e.g. response to a challenge)
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6300' (see 11.5.1), '63CX' (see 11.5.1), '6581', '6700', '6982', '6983', '6984', '6A81', '6A82', '6A86', '6A88' (see 11.5.1)

11.5.3 GET CHALLENGE command

The command requires the issuing of a challenge (e.g. a random number for a cryptographic authentication or a sentence to prompt for a biometric authentication using voiceprints) for use in a security-related procedure (e.g. EXTERNAL AUTHENTICATE command). The challenge is valid at least for the next command; this clause specifies no further condition.

Table 96 — GET CHALLENGE command-response pair

CLA	As defined in 5.4.1
INS	'84'
P1	See 11.5.1
P2	'00' (any other value is RFU)
L_c field	Absent for encoding $N_c = 0$
Data field	Absent
L_e field	Present for encoding $N_e > 0$

Data field	Challenge
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6700', '6A86' (see 11.5.1)

11.5.4 EXTERNAL AUTHENTICATE command

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

The 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.

Any successful authentication requires the use of the last challenge obtained from the card. The card may record unsuccessful authentications (e.g. to limit the number of further uses of the reference data).

The absence of command data field may be used either to retrieve the number 'X' of further allowed retries (SW1-SW2 set to '63CX'), or to check whether the verification is required or not (SW1-SW2 set to '9000').

MUTUAL AUTHENTICATE function — The MUTUAL AUTHENTICATE function uses the same functionalities as EXTERNAL and INTERNAL AUTHENTICATE commands. It is based upon a previous GET CHALLENGE command and a key, possibly secret, stored in the card. The card and the interface device share authentication-related data, including two challenges: one issued by the card, another one issued by the interface device.

NOTE The command may be used for implementing authentication as specified in parts 2 and 3 of ISO/IEC 9798 [10].

Table 97 — EXTERNAL AUTHENTICATE command-response pair

CLA	As defined in 5.4.1
INS	'82'
P1-P2	See 11.5.1 and Table 94
L_c field	Absent for encoding $N_c = 0$, present for encoding $N_c > 0$
Data field	Absent or authentication-related data (e.g. response to a challenge)
L_e field	Absent for encoding $N_e = 0$

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6300' (see 11.5.1), '63CX' (see 11.5.1), '6581', '6700', '6982', '6983', '6984', '6A81', '6A82', '6A86', '6A88' (see 11.5.1)

Table 98 — Command-response pair for MUTUAL AUTHENTICATE function

CLA	As defined in 5.4.1
INS	'82'
P1-P2	See 11.5.1 and Table 94
L_c field	Present for encoding $N_c > 0$
Data field	Authentication-related data
L_e field	Present for encoding $N_e > 0$

Data field	Authentication-related data
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6300' (see 11.5.1), '63CX' (see 11.5.1), '6581', '6700', '6982', '6983', '6984', '6A81', '6A82', '6A86', '6A88' (see 11.5.1)

11.5.5 GENERAL AUTHENTICATE command

The command refines the EXTERNAL, INTERNAL and MUTUAL AUTHENTICATE functions; namely, either an entity in the outside world authenticates an entity in the card (INTERNAL AUTHENTICATE function), or an entity in the card authenticates an entity in the outside world (EXTERNAL AUTHENTICATE function), or both (MUTUAL AUTHENTICATE function).

While appropriate for authentication mechanisms involving challenge-response pairs, the EXTERNAL and INTERNAL AUTHENTICATE commands preclude authentication mechanisms involving witness-challenge-response triples (see ISO/IEC 9798 [10]) and more generally multi-step authentication protocols. Those require two or more GENERAL AUTHENTICATE C-RPs: such C-RPs may be chained (see 5.3.3).

The function (either INTERNAL, or EXTERNAL, or MUTUAL AUTHENTICATE) can be performed only if the security status satisfies the security attributes for this operation. Any successful authentication may be subject to completion of prior commands (e.g. VERIFY, SELECT) or selections (e.g. the relevant secret). The result (yes or no) of a control performed by the card may conditionally update the security status. The card may record the number of times the function is issued, in order to limit the number of further uses of the relevant secret or the algorithm. The card may record unsuccessful authentications, e.g. to limit the number of further uses of the reference data.

Table 99 — GENERAL AUTHENTICATE command-response pair

CLA	As defined in 5.4.1
INS	'86' or '87'
P1-P2	See 11.5.1 and Table 94
L_c field	Present for encoding $N_c > 0$
Data field	Authentication-related data
L_e field	Absent for encoding $N_e = 0$, present for encoding $N_e > 0$

Data field	Absent (either due to the absence of L_e field, e.g. the last command of an EXTERNAL AUTHENTICATE function, or if the process is aborted), or authentication-related data
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6300' (see 11.5.1), '63CX' (see 11.5.1), '6581', '6700', '6982', '6983', '6984', '6A81', '6A82', '6A86', '6A88' (see 11.5.1)

Table 100 — Dynamic authentication DOs for witness-challenge-response triples

Tag		Value
'7C'		Set of dynamic authentication DOs with the following tags
	'80'	Witness (e.g. one or more positive numbers less than the public modulus in use)
	'81'	Challenge (e.g. one or more numbers, possibly 0 less than the public exponent in use)
	'82'	Response (e.g. one or more positive numbers less than the public modulus in use)
	'83'	Committed challenge (e.g. the hash-code of a large random number including one or more challenges)
	'84'	Authentication code (e.g. the hash-code of one or more data fields and a witness DO)
	'85'	An ephemeral public key for key agreement technique.
	'86'	Enciphered data
	'06'	OID (see text below)
	'A0'	Identification data template
— Under tag '7C', ISO/IEC JTC 1/SC 17 reserves any other DO of the context-specific class, if no OID specifies the context otherwise.		

When present, each data field shall contain an interindustry template referenced by tag '7C'.

The default context of the GENERAL AUTHENTICATE command(s) regarding the cryptographic protocol in use is reserved for witness-challenge-response triples (see C.1). In this case in the dynamic authentication template, the context-specific class is reserved for dynamic authentication DOs as listed in Table 100.

In this case respective context specific DOs shall be sent in the C-RPs, embedded in a template with Tag '7C' (see Table 100).

If the GENERAL AUTHENTICATE COMMAND(s) are used for a multi-step authentication protocol (see C.2), the respective protocol OID or algorithm reference being associated with an OID shall be included in a preceded MANAGE SECURITY ENVIRONMENT command (see 11.5.11) in the CRT AT for Authentication and/or shall be contained in the template with Tag '7C' indicating the protocol specific interpretation of the additionally embedded context specific DOs sent in the C-RPs.

For the default context, the following rules apply within the interindustry template for dynamic authentication.

— If a DO is empty in a template, then it shall be complete in the template in the next data field.

— In the first command data field, the template indicates the dynamic authentication function as follows.

  — A witness request, e.g. an empty witness, denotes an INTERNAL AUTHENTICATE function.

  — A challenge request, e.g. an empty challenge, denotes an EXTERNAL AUTHENTICATE function.

  — The absence of empty DO denotes a MUTUAL AUTHENTICATE function. Then unless the card aborts the process, the template in the response data field shall contain the same DO s as the template in the command data field. The MUTUAL AUTHENTICATE function allows two entities to agree on a session key using a pair of “exponential” data elements referenced by tag '85' (see key agreement techniques in ISO/IEC 11770-3 [17]).

The dynamic authentication may protect data fields exchanged during a session. Both entities maintain a current hash-code, updated by including one command or response data field at a time. The DO'84' conveys an authentication code resulting from updating the current code by including a witness DO'80'. The verifier successively reconstructs a witness and an authentication code: if the reconstructed witness is not 0 and if the two codes are identical, then the authentication is successful.

For the default context, C.1 illustrates GENERAL AUTHENTICATE C-RPs for implementing INTERNAL, EXTERNAL and MUTUAL AUTHENTICATE functions, with extensions to data field authentication and key agreement.

11.5.6 VERIFY command

Table 101 — VERIFY command-response pair

CLA	As defined in 5.4.1	
INS	'20' or '21'	
P1	'00'	Normal operation
	'FF'	set verification status to “not verified”, see last paragraph of this clause
	any other value	RFU
P2	See Table 94	
L_c field	Absent for encoding $N_c = 0$, present for encoding $N_c > 0$	
Data field	INS = '20'	Verification data or absent
	INS = '21'	Verification data DO, and, conditionally, extended header list
L_e field	Absent for encoding $N_e = 0$	

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6286', '6300' (see 11.5.1), '63CX' (see 11.5.1), '6581', '6700', '6982', '6983', '6984', '6A81', '6A82', '6A86', '6A88' (see 11.5.1)

The command initiates the comparison in the card of stored reference data with verification data sent from the interface device (e.g. password) or from a sensor on the card (e.g. fingerprint). The security status may be modified as a result of a comparison. The card may record unsuccessful comparisons (e.g. to limit the number of further uses of the reference data).

If INS = '20', the command data field is normally present for conveying verification data. The absence of command data field is used to check whether the verification is required (SW1-SW2 = '63CX' where 'X' encodes the number of further allowed retries), or not (SW1-SW2 = '9000').

If INS = '21', the command data field shall convey a verification data DO (e.g. tag '5F2E', see ISO/IEC 7816-11), normally not empty. The presence of an empty verification data DO and an extended header list (tag '4D',

see 8.4.5) expresses that the verification data come from a sensor on the card. The extended header list references the verification data DO.

With both INS values, P1='FF' shall only be used with L_c and command data field absent. The command shall set the verification status of the relevant reference data as "not verified".

11.5.7 CHANGE REFERENCE DATA command

The command either replaces reference data stored in the card with new reference data sent from the interface device, or initiates their comparison with verification data sent from the interface device and then conditionally replaces them with new reference data sent from the interface device. It can be performed only if the security status satisfies the security attributes for this command.

Table 102 — CHANGE REFERENCE DATA command-response pair

CLA	As defined in 5.4.1		
INS	'24' or '25'		
P1	'00' or '01' (any other value is RFU)		
P2	See Table 94		
L_c field	Present for encoding $N_c > 0$		
Data field	INS = '24'	P1 = '00'	Verification data followed without delimitation by new reference data
		P1 = '01'	New reference data
	INS = '25'	P1 = '00'	Verification data DO followed by new reference data DO
		P1 = '01'	New reference data DO
L_e field	Absent for encoding $N_e = 0$		

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6300' (see 11.5.1), '63CX' (see 11.5.1), '6581', '6700', '6982', '6983', '6984', '6A81', '6A82', '6A86', '6A88' (see 11.5.1)

11.5.8 ENABLE VERIFICATION REQUIREMENT command

The command switches on the requirement to compare reference data with verification data. It can be performed only if the security status satisfies the security attributes for this command.

Table 103 — ENABLE VERIFICATION REQUIREMENT command-response pair

CLA	As defined in 5.4.1	
INS	'28'	
P1	'00' or '01' (any other value is RFU)	
P2	See Table 94	
L_c field	Absent for encoding $N_c = 0$, present for encoding $N_c > 0$	
Data field	P1 = '00'	Verification data
	P1 = '01'	Absent
L_e field	Absent for encoding $N_e = 0$	

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6300' (see 11.5.1), '63CX' (see 11.5.1), '6581', '6700', '6982', '6983', '6984', '6A81', '6A82', '6A86', '6A88' (see 11.5.1)

11.5.9 DISABLE VERIFICATION REQUIREMENT command

Table 104 — DISABLE VERIFICATION REQUIREMENT command-response pair

CLA	As defined in 5.4.1	
INS	'26'	
P1	'00', '01' or 100xxxxx where xxxxx is a reference data number (any other value is RFU)	
P2	See Table 94	
L_c field	Absent for encoding $N_c = 0$, present for encoding $N_c > 0$	
Data field	P1 = '00' or P1 = 100x xxxx	Verification data
	P1 = '01'	Absent
L_e field	Absent for encoding $N_e = 0$	

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6300' (see 11.5.1), '63CX' (see 11.5.1), '6581', '6700', '6982', '6983', '6984', '6A81', '6A82', '6A86', '6A88' (see 11.5.1)

The command switches off the requirement to compare reference data with verification data, and possibly switches on the requirement to compare other reference data with verification data. It can be performed only if the security status satisfies the security attributes for this command.

11.5.10 RESET RETRY COUNTER command

The command either resets the reference data retry counter to its initial value, or changes reference data on completion of a reset of the reference data retry counter to its initial value. It can be performed only if the security status satisfies the security attributes for this command.

Table 105 — RESET RETRY COUNTER command-response pair

CLA	As defined in 5.4.1		
INS	'2C' or '2D'		
P1	'00', '01', '02' or '03' (any other value is RFU)		
P2	See Table 94		
L_c field	Absent for encoding $N_c = 0$, present for encoding $N_c > 0$		
Data field	INS = '2C'	P1 = '03'	Absent
		P1 = '00'	Resetting code followed without delimitation by new reference data
		P1 = '01'	Resetting code
		P1 = '02'	New reference data
	INS = '2D'	P1 = '03'	Absent
		P1 = '00'	Resetting code DO followed by a new reference data DO
		P1 = '01'	Resetting code DO
		P1 = '02'	New reference data DO
L_e field	Absent for encoding $N_e = 0$		

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6300' (see 11.5.1), '63CX' (see 11.5.1), '6581', '6700', '6982', '6983', '6984', '6A81', '6A82', '6A86', '6A88' (see 11.5.1)

11.5.11 MANAGE SECURITY ENVIRONMENT command

The command prepares secure messaging (see clause 10) and security commands (e.g. EXTERNAL, INTERNAL and GENERAL AUTHENTICATE, see also PERFORM SECURITY OPERATION in ISO/IEC 7816-8). The command supports the following functions:

— SET, i.e. setting or replacing one component of the current SE;

— STORE, i.e. saving the current SE under the SEID given in P2;

— RESTORE, i.e. replacing the current SE by an SE stored in the card and identified by the SEID given in P2;

— ERASE, i.e. erasing an SE stored in the card and identified by the SEID given in P2.

— RESET, i.e. restoring the default SE after DF or application DF selection ;

— GET SE, i.e. retrieving all control reference DOs pertaining to the current SE

— GET CRT, i.e. retrieving one control reference template pertaining to the current SE

KEY DERIVATION function — The usage of a master key concept may require the derivation of a key in the card containing the master key. Table 109 shows the usage of the MANAGE SECURITY ENVIRONMENT command for deriving a key. It is assumed that the master key and the algorithm are implicitly selected in the card (otherwise, the MANAGE SECURITY ENVIRONMENT command can additionally select a key and an algorithm).

NOTE Depending on the algorithm reference, the data for deriving a key from a master key may be part of the input data of the subsequent command (e.g. EXTERNAL AUTHENTICATE). In this case the usage of the MANAGE SECURITY ENVIRONMENT command for deriving the key is not necessary.

Table 106 — MANAGE SECURITY ENVIRONMENT command-response pair

CLA	As defined in 5.4.1	
INS	'22'	
P1	See Table 107	
P2	See Table 108	
L_c field	Absent for encoding $N_c = 0$, present for encoding $N_c > 0$	
Data field	GET, STORE, RESTORE, ERASE, RESET	Absent
	SET	Control reference DO s (SET)
L_e field	SET, STORE, RESTORE, ERASE, RESET	Absent for encoding $N_e = 0$
	GET	Present for encoding $N_e > 0$

Data field	GET SE	Concatenation of control reference DOs
	GET CRT	One control reference template
	other	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6600', '6987', '6988, '6A88' (see 11.5.1)	

Table 107 — Coding of P1 in the MANAGE SECURITY ENVIRONMENT command

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	-	1	-	-	-	-	Secure messaging in command data field
-	-	1	-	-	-	-	-	Secure messaging in response data field
-	1	-	-	-	-	-	-	Computation, decipherment, internal authentication and key agreement
1	-	-	-	-	-	-	-	Verification, encipherment, external authentication and key agreement
-	-	-	-	0	0	0	1	SET
1	1	1	1	0	0	1	0	STORE
1	1	1	1	0	0	1	1	RESTORE
1	1	1	1	0	1	1	1	RESET
1	1	1	1	0	1	0	0	ERASE
0	0	0	0	1	0	0	0	GET CRT
0	0	0	0	0	0	0	0	GET SE
— Any other value is RFU.								

Table 108 — Coding of P2 in the MANAGE SECURITY ENVIRONMENT command

Value	Meaning
'XX'	SEID with from interval ['01' .. 'FE'] without 'EF' (see 0) if P1 indicates STORE, RESTORE or ERASE
'A4', 'A6', 'AA', 'B4', 'B6', 'B8'	Tag of CRT with the meaning from Table 54 present in the command data field if P1 indicates SET or GET CRT
'00'	if P1 indicates GET SE or RESET
— Any other value is RFU	

Table 109 — Command-response pair for KEY DERIVATION function

CLA	As defined in 5.4.1
INS	'22'
P1	'X1' (SET, see Table 107)
P2	CRT tag (e.g. 'A4' if an EXTERNAL AUTHENTICATE follows, or 'B4' if a VERIFY CRYPTOGRAPHIC CHECKSUM follows)
L_c field	Present for encoding $N_c > 0$
Data field	{'94' - L - Data for deriving a key (mandatory)}; SM DOs may be present
L_e field	Absent for encoding $N_e = 0$

Data field	Absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6600', '6987', '6988, '6A88' (reference data not found)

11.6 Miscellaneous

11.6.1 COMPARE command

The command initiates a comparison of comparison data with reference data, which will be either the value of a primitive DO, or the contents of a record, or the contents of a DataString. The reference data are defined according to Table 86. The card shall interpret any individual element relevant for the comparison as a binary coded number. When an interval is provided in the command or response data, the endpoints of that interval shall have the same type as the data to which they are compared.

Table 110 — COMPARE command-response pair

CLA	As defined in 5.4.1		
INS	'33'		
P1	'00'	COMPARE BINARY function,	reference value is located in a transparent EF
	'01'	COMPARE RECORD function,	reference value is located in a structured EF
	'02	COMPARE DATA function,	reference value is located in a DO
	other	RFU	
P2	Operation qualifier (for details see below)		
	'00'	comparison defined by OID	
	'01'	equal	
	'02'	greater than	
	'03'	less than	
	'04'	not equal	
	'05'	element of interval [lowerEndpoint, upperEndpoint]	
	'06'	not element of interval [lowerEndpoint, upperEndpoint]	
	'07'	the comparison data shall belong to the set of finite values defined by the command	
	'08'	the comparison data shall not belong to the set of finite values defined by the command	
	['09' .. '7F']	RFU	
	['80' .. 'FF']	Proprietary	
L_c field	Present for encoding $N_c > 0$		
Data field	DO'06' OID conditional, present if and only if P2='00'		
	Mandatory choice between	General reference template DO'60' as defined in Table 86.	
		DO'78' followed by a DO with a tag '70' to '72' or tags '74' to '77' (see 8.3.5), nesting application defined DOs for referencing the target of the command.	
		Object locator DO'7F72' as defined Table 37	
		Wrapper DO'63' (see 8.4.8).	
	The data field optionally ends with comparison data encapsulated under DO'53' or DO'73'		
L_e field	Absent for encoding $N_e = 0$, or present for encoding $N_e > 0$		

Data field	Absent or present (see below)
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6282', '6340', '6982'

The comparison data shall be:

— either transmitted in the command data field,

— or known by the card.

By definition the reference value is the number coded in the reference data and the comparison value is the number coded in the comparison data.

The function of the command (COMPARE BINARY or COMPARE RECORD or COMPARE DATA) shall be coded in P1 set to the values '00', '01', '02' respectively.

The type of the operation is defined by P2 (operation qualifier):

— P2 = '00': the comparison function is defined by an OID DO'06' in the data field.

— P2 = '01': the comparison is successful if the reference value is equal to the comparison value,,

— P2 = '02': the comparison is successful if the reference value is greater than the comparison value,

— P2 = '03': the comparison is successful if the reference value is less than the comparison value,

— P2 = '04': the comparison is successful if the reference value is not equal to the comparison value,

— P2 = '05': the comparison is successful if the reference value is element of the closed interval defined by the command.

— P2 = '06': the comparison is successful if the reference value is not element of the closed interval defined by the command.

— P2 = '07': the comparison is successful if the reference value is element of the finite set of values defined by the command.

— P2 = '08': the comparison is successful if the reference value is not element of the finite set of values defined by the command.

— P2 values from interval ['09' .. '7F'] are RFU.

— P2 values from interval ['80' .. 'FF'] are proprietary.

The result is indicated by the status word in the command response. SW1-SW2 set to '6340' indicates that the comparison data do not match the reference data.

If the comparison data is given in the command data field, then for P2 equals

— '01' or '02' or '03' or '04' the value shall be given in the value field of DO'53'.

— '05' or '06' the closed interval shall be provided in the value field of DO'73'. In this case the DO'73' shall contain nothing but two DOs'80'. The first DO'80' shall contain the lower endpoint of the interval. The second DO'80' shall contain the upper endpoint of the interval.

— '07 or '08' the set shall be provided in the value field of DO'73'. In this case each value of the set shall be given in the value field of a DO'53'.

When the command is successful, an optional response data field may be present when P2='01 or '04' or '05' or '06'. When present, it shall be the concatenation of two DO'80' defining the endpoints of the closed interval within which one element matches (P2='01' or '05') or within which no element matches (P2='04' or '06') the comparison data.

11.6.2 GET ATTRIBUTE command

The command GET ATTRIBUTE retrieves one object attribute of the referenced security object, if the access conditions for the respective operations are satisfied. The respective security object shall be referenced by a data object locator in the data field of the command. If an EF or a DO is referenced by the command an empty attribute reference is mandatory in the object locator template. The respective attribute may be related to a dedicated security environment or to a dedicated service provided by the referenced security object. The relevant attribute is retrieved as a BER-TLV object in the data field of the response APDU.

Table 111 — GET ATTRIBUTE command-response pair

CLA	As defined in 5.4.1	
INS	'34' or '35'	
P1-P2	'0000'	
L_c field	Present for encoding $N_c > 0$	
Data field	INS = '34'	Object locator template (see Table 37)
	INS = '35	Object locator DO'7F72' (see Table 37)
L_e field	Present for encoding $N_e > 0$	

Data field	Attributes coded in BER-TLV
SW1-SW2	See Table 5 and Table 6 when relevant

11.7 Transmission handling

11.7.1 GET RESPONSE command

The command transmits (part of) response APDUs (see 5.3.4).

Table 112 — GET RESPONSE command-response pair

CLA	As defined in 5.4.1
INS	'C0'
P1-P2	'0000' (any other value is RFU)
L_c field	Absent for encoding $N_c = 0$
Data field	Absent
L_e field	Present for encoding $N_e > 0$

Data field	Absent in any error case, or (part of) a response APDU according to N_e
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '61XX' ('XX' encodes the number of extra bytes still available by a subsequent GET RESPONSE), '6281', '6700', '6A81', '6A82', '6A86', '6CXX'

11.7.2 ENVELOPE command

With INS = 'C2', the command, or a chain of such commands transmits a payload which shall be a command APDU.

With INS= 'C3', the command, or a chain of such commands transmits a payload which shall be a DO. Particular uses are:

— When the DO is DO'52' (command to perform), it has the same functionality as 'C2', except that the response shall be a concatenation of DOs. If the response specified by the nested command is not, it shall be nested in DO'53'.

— When the DO is DO'06' (object identifier), the success of the command (SW1-SW2 = '9000') means that the card is ready to start a procedure defined by the document referenced by this object identifier. This procedure, or script, shall comply with the standard APDU syntax. The end of the procedure shall be defined in the procedure.

EXAMPLE 1 In order to use IP datagrams at the interface, the specification referred to by the OID shall define the encapsulation of those datagrams in command and response payloads.

EXAMPLE 2 In order to use at the interface the command syntax derived from the API defined by ISO/IEC 24727-3 [25], the BER-TLV marshalled requests and confirmations (payloads), also defined in ISO/IEC 24727-3 [25], are encapsulated in the command data fields of ENVELOPE C-RTs with INS= 'C3'.

NOTE Annex B shows the usage of the ENVELOPE command for secure messaging.

Table 113 — ENVELOPE command-response pair

CLA	As defined in 5.4.1	
INS	'C2' or 'C3'	
P1-P2	'0000' (any other value is RFU)	
L_c field	Present for encoding $N_c > 0$	
Data field	INS = 'C2'	(Part of) a command APDU
	INS = 'C3'	DO or DO fragment
L_e field	Absent for encoding $N_e = 0$, present for encoding $N_e > 0$	

Data field	(Part of) a response APDU (INS = 'C2'), or (parts of) DO'53' (INS = 'C3' with DO'52'), or absent
SW1-SW2	See Table 5 and Table 6 when relevant, e.g. '6700'

12 Application-independent card services

The purpose of card services is to provide interchange mechanisms between a card and an interface device knowing nothing about each other except that they both comply with this document. Card services result from any combination of historical bytes (see 12.1.1), the contents of EF.DIR and EF.ATR/INFO (see 12.2.1 and 12.2.2) and sequences of commands. Unless otherwise specified, every command APDU uses CLA set to '00', i.e. no command chaining, 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. An application may use other mechanisms compatible with this document for achieving similar functions. Therefore such solutions may not guarantee interchange.

Interindustry information may also be recovered by the commands sent on additional logical channels than the basic logical channel. Furthermore in such commands any N_e may be used. This statement is related to the paragraph before as well as the information retrieval in 12.1.1.1, 12.1.2, 12.2.1, 12.2.2 and 12.4.

12.1 Card identification

This service allows the interface device to identify the card and to deal with it. The historical bytes (see 12.1.1) provide a generic support to card identification. The card provides information to the outside world on its logical content directly, e.g. through the card service data byte (see Table 116), and/or indirectly, e.g. through the initial access data (see 12.1.1.6) indicating an access to a file implicitly selected immediately after enabling a physical interface (see 5.1). Consequently, the data available at this point, i.e. the initial data string (see 12.1.2), may not be subsequently retrievable.

12.1.1 Historical bytes

12.1.1.1 Purpose and retrieval

The historical bytes (string of up to 15 bytes, as defined in ISO/IEC 7816-3) indicate operating characteristics of the card. When a card answers to reset, the Answer-to-Reset may contain historical bytes.

When the physical interface does not allow a card to answer to reset, e.g. if it is accessed by Universal Serial Bus or by radio frequency, the GET DATA command (see 11.4.3) may retrieve:

— historical bytes as the value of DO'5F52'. The command APDU is: '00CA 5F52 0F'. For practical use, DO'5F52' should belong to the current template after enabling a physical interface (see 5.1).

— the Answer-To-Reset as the value of DO'5F51'. The command APDU is: '00CA 5F51 20'. For practical use, DO'5F51' should belong to the current template after enabling a physical interface (see 5.1).

12.1.1.2 Structure and COMPACT-TLV data objects

The first historical byte is the “category indicator byte”. If the category indicator byte is set to '00' or '8X', then Table 114 summarizes the format of the historical bytes. Any other value indicates a proprietary format.

If the first historical byte is set to:

— '00' the remaining historical bytes shall consist of optional consecutive COMPACT-TLV data objects followed by a mandatory status indicator (see 12.1.1.11).

— '80' the remaining historical bytes shall consist of optional consecutive COMPACT-TLV data objects; the last one may carry a status indicator in COMPACT-TLV format (see 12.1.1.11).

Any interindustry DO consisting of a tag field set to '4X', a length field set to '0Y' and a value field of Y bytes can be converted into a COMPACT-TLV data object consisting of a byte set to 'XY' called “compact header” and a value field of Y bytes.

Any interindustry data element defined hereafter (see 12.1.1.3 to 12.1.1.10) may be present in EF.ATR/INFO (see 12.2.2). If present in EF.ATR/INFO, it shall appear in a DO, i.e. a tag field set to '4X', a length field set to '0Y' and a value field of Y bytes.

Table 114 — Category indicator byte

Value	Meaning
'00'	A status indicator shall be present as the last three historical bytes (see 12.1.1.11)
'80'	A status indicator may be present in a COMPACT-TLV data object (one, two or three bytes, see 12.1.1.11)
'81' to '8F'	RFU
— Any other value indicates a proprietary format.	

12.1.1.3 Country or issuer indicator

Referenced by a compact header set to either '1Y' or '2Y', this interindustry data element is a country or issuer indicator (see also tags '41' and '42' in Table 16). Table 115 shows the country or issuer indicator.

Table 115 — Country or issuer indicator

Compact header	Value
'1Y'	Country code (see ISO 3166-1 [3]) and optional national data
'2Y'	Issuer identification number (see ISO/IEC 7812-1 [5]) and optional issuer data

A country indicator consists of a country code (three quartets with values from '0' to '9', see ISO 3166-1 [3]) followed by subsequent data (at least one quartet). The relevant national standardization body shall choose those subsequent data (odd number of quartets).

An issuer indicator consists of an issuer identification number (see ISO/IEC 7812-1 [5]) possibly followed by subsequent data. The card issuer shall choose those subsequent bytes if any (for encoding, e.g. a Primary Account Number).

NOTE In ISO/IEC 7812-1:1993, an issuer identification number might consist of an odd number of quartets with a value from '0' to '9'. Then it was mapped into a byte string by setting bits b4 to b1 of the last byte to 1111.

12.1.1.4 Application identifier

Referenced by a compact header set to 'FY', this interindustry data element is an application identifier (AID, see 12.2.3, see also tag '4F' in Table 16). If present in the historical bytes or in the initial data string (see 12.1.2), an AID denotes an implicitly selected application (see 12.2.5.1).

12.1.1.5 Card service data

Referenced by a compact header set to '31', this interindustry data element indicates methods available in the card for supporting services described in the present clause 12. Table 116 shows the card service data byte. If present in the historical bytes or in the initial data string (see 12.1.2), the card service data byte indicates whether EF.DIR and/or EF.ATR/INFO (see 12.2.1 and 12.2.2) are present or not, and how to access them. The absence of card service data byte in the historical bytes and in the initial data string indicates that the card supports only the implicit application selection (default value).

12.1.1.6 Initial access data

Referenced by a compact header set to '4Y', this interindustry data element indicates a command APDU assumed to be the first command after enabling a physical interface (see 5.1). The command APDU is specified in 12.1.2.

12.1.1.7 Card issuer's data

Referenced by a compact header set to '5Y', this interindustry data element is not defined in ISO/IEC 7816 [6]. The card issuer defines a length, a structure and a coding.

Table 116 — Coding of the card service data byte

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	-	-	-	-	-	-	**Application selection**
1	-	-	-	-	-	-	-	— by full DF name
-	1	-	-	-	-	-	-	— by partial DF name
-	-	x	x	-	-	-	-	**DOs available**
-	-	1	-	-	-	-	-	— in EF.DIR (see 12.2.1)
-	-	-	1	-	-	-	-	— in EF.ATR/INFO (see 12.2.2)
-	-	-	-	x	x	x	-	**EF.DIR and EF.ATR/INFO access services**
-	-	-	-	1	0	0	-	— by the READ BINARY command (transparent structure)
-	-	-	-	0	0	0	-	— by the READ RECORD (S) command (record structure)
-	-	-	-	0	1	0	-	— by the GET DATA command (BER-TLV structure)
-	-	-	-	any other value			-	RFU
-	-	-	-	-	-	-	0	**Card with MF**
-	-	-	-	-	-	-	1	**Card without MF**

12.1.1.8 Pre-issuing data

Referenced by a compact header set to '6Y', this interindustry data element is not defined in ISO/IEC 7816 [6]. The card manufacturer defines a length, a structure and a coding for a card manufacturer, an integrated circuit name, an integrated circuit manufacturer, a ROM mask version, an operating system version, etc. This interindustry data element may contain an integrated circuit manufacturer identifier (see ISO/IEC 7816-6).

12.1.1.9 Card capabilities

Referenced by a compact header set to '71', '72' or '73', this interindustry data element consists of up to three software function tables. If the length of the data element is

— one byte, then the data element shall consist of the first software function table (see Table 117).

— two bytes, then the data element shall consist of the first software function table as first byte (see Table 117) and the second software function table as second byte (see Table 118).

— three bytes, then the data element shall consist of the first software function table as first byte (see Table 117), the second software function table as second byte (see Table 118) and the third software function table as third byte (see Table 119).

Content of the software function tables:

— The first software function table indicates selection methods supported by the card.

— The second software function table is the “data coding byte”. The data coding byte may also be present as the second byte in the file CP referenced by tag '82' (see Table 10).

— The third software function table indicates the ability to chain commands, to handle extended L_c and L_e fields and to manage logical channels.

Table 117 — Coding of the first software function table (selection methods)

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	x	x	x	-	-	-	**DF selection** (see 7.3)
1	-	-	-	-	-	-	-	— by full DF name
-	1	-	-	-	-	-	-	— by partial DF name
-	-	1	-	-	-	-	-	— by path
-	-	-	1	-	-	-	-	— by file identifier
-	-	-	-	1	-	-	-	Implicit DF selection
-	-	-	-	-	1	-	-	Short EF identifier supported
-	-	-	-	-	-	1	-	Record number supported
-	-	-	-	-	-	-	1	Record identifier supported

Table 118 — Coding of the second software function table (data coding byte)

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	EFs of BER-TLV structure supported
-	x	x	-	-	-	-	-	**Behaviour of write functions**
-	0	0	-	-	-	-	-	— One-time write
-	0	1	-	-	-	-	-	— Proprietary
-	1	0	-	-	-	-	-	— Write OR
-	1	1	-	-	-	-	-	— Write AND
-	-	-	-	x	x	x	x	**Data unit size in quartets** (from one to 32 768 quartets, i.e. 16 384 bytes) (power of 2, e.g. 0001 = 2 quartets = one byte, default value)
-	-	-	x	-	-	-	-	**Value 'FF' for the first byte of BER-TLV tag fields** (see 8.1.1)
-	-	-	0	-	-	-	-	— Invalid (used for padding, default value)
-	-	-	1	-	-	-	-	— Valid (long private tags, constructed encoding)

Table 119 — Coding of the third software function table (command chaining, length fields and logical channels)

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	**Command chaining** (see 5.3.3)
-	1	-	-	-	-	-	-	**Extended L_c and L_e fields** (see 5.1)
-	-	1	-	-	-	-	-	**Extended Length Information in EF.ATR/INFO**
-	-	-	x	x	-	-	-	**Logical channel number assignment** (see 5.4.2 and 11.1)
-	-	-	1		-	-	-	— by the card
-	-	-		1	-	-	-	— by the interface device
-	-	-	0	0	-	-	-	Only basic logical channel available
-	-	-	-	-	y	z	t	**Maximum number of logical channels** (see 5.4.1) — y, z and t not all set to 1 means 4y+2z+t+1, i.e. from one to seven — y = z = t = 1 means eight or more

12.1.1.10 Application Family Identifier

Referenced by a compact header set to '91', this interindustry data element consists of one byte defined in ISO/IEC 14443-3 [18].

12.1.1.11 Status indicator

If the category indicator byte is set to '00', then the last three historical bytes shall be a status indicator, namely a card LCS (1 byte) followed by two processing status bytes denoted SW1-SW2.

If the category indicator byte is set to '80', then an interindustry data element referenced by a compact header set to '81', '82' or '83' may be present as a status indicator on one, two or three bytes (any other length is RFU) at the end of the historical bytes.

— If the length is one, then the data element is a card LCS.

— If the length is two, then the data element is SW1-SW2.

— If the length is three, then the data element is LCS followed by SW1-SW2.

LCS shall be interpreted according to 7.4.10 and Table 14; the value '00' indicates that the status is not reported. SW1-SW2 shall be interpreted according to 5.6, Table 5 and Table 6; the value '0000' indicates that the status is not reported.

12.1.2 Initial data string recovery

Referenced by a compact header set to '4Y' in the historical bytes (see 12.1.1) or by tag '44' in EF.ATR/INFO (see 12.2.2), the interindustry data element called “initial access data” indicates a command APDU.

— If the length is one, then the command APDU is a READ BINARY command (see 11.2.3) as follows: CLA INS P1 P2 set to '00B0 0000' and an L_e field set to the first and only byte of initial access data.

— If the length is two, then the first byte of initial access data indicates the structure (bit b8) and the short EF identifier (bits b5 to b1) of the EF to read, according to Table 120.

  — If bit b8 of the first byte is set to 1, then the command APDU is a READ BINARY command (see 11.2.3) as follows: CLA INS set to '00B0'. If the bits b5 to b1 are set to 00000 then P1 shall be set to '00' otherwise P1 shall be set to the first byte of initial access data, P2 set to '00' and an L_e field set to the second byte of initial access data.

  — If bit b8 of the first byte is set to 0, then the command APDU is a READ RECORD (s) command (see 11.3.3) as follows: CLA INS P1 set to '00B2 01', P2 consisting of bits b8 to b4 set to bits b5 to b1 of the first byte of initial access data (indicating current EF or a short EF identifier) and bits b3 to b1 set to 110, and an L_e field set to the second byte of initial access data.

— If the length is five or more, then the command APDU consists of the Y bytes of initial access data.

The command APDU shall be submitted to the card. If the process is completed, then the response data field is a string of interindustry DOs every application might be interested in, called the “initial data string”.

Table 120 — Coding of the first byte of initial access data when the length is two

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	-	-	-	-	-	-	-	**EF structure**
0	-	-	-	-	-	-	-	Record structure
1	-	-	-	-	-	-	-	Transparent structure
-	x	x	-	-	-	-	-	00 (any other value is RFU)

-	-	-	0	0	0	0	0	Current EF
-	-	-	not all equal					Short EF identifier

12.2 Application identification and selection

This service allows the interface device to know what applications are supported by the card, if any, as well as how to identify and select them.

Two specific EFs provide a generic support to application identification and selection, namely EF.DIR and EF.ATR/INFO. They contain a set of DOs. In these EFs, erased or modified DOs may induce padding before, between and after DOs (see 8.1.1) in the response data fields of READ BINARY/RECORD(s) commands.

12.2.1 EF.DIR

This EF indicates a list of applications supported by the card. It contains a set of application templates (see 12.2.4) and/or application identifier DOs (see 12.2.3), in any order. It determines which commands shall be performed in order to select the indicated applications.

If an MF and an EF.DIR are present, EF.DIR shall have the MF as parent file and its path shall be '3F002F00'. At MF level, the short EF identifier 30, i.e. 11110 in binary, references EF.DIR.

If an MF is not present, EF.DIR may be present and should be addressable by file identifier '2F00' after enabling a physical interface (see 5.1).

NOTE The command (sequence) for retrieving the content depends on the structure of EF.DIR (e.g. see Table 116) and may be known implicitly by an application in the outside world.

If EF.DIR supports DO handling, or if its contents are not stored in a file, a card may provide its contents by the GET DATA command (see 11.4.3) The response data to the command '00CB 2F00 02 5C00 00' is the concatenation of all DOs which are present in EF.DIR, or would be present if EF.DIR does not actually exist.

If the card provides this service, the command should be successful at least immediately after enabling a physical interface (see 5.1).

12.2.2 EF.ATR/INFO

This EF indicates operating characteristics of the card. It contains a set of interindustry DOs which cannot be nested in EF.DIR, either because not relevant to application selection, or because there is no EF.DIR.

NOTE EF.ATR/INFO was called EF.ATR in the previous edition of this document. As some contactless cards do not provide an ATR, this terminology was misleading in the contactless card world. The new terminology EF.ATR/INFO is recommended. It may still be called EF.ATR in contact card standards or specifications, and may now be called EF.INFO in contactless card standards and specifications.

If an MF and an EF.ATR/INFO are present, EF.ATR/INFO shall have the MF as parent file and its path shall be '3F002F01'.

If an MF is not present, an EF.ATR/INFO may be located within an application with its application DF as parent. Its file identifier shall be '2F01', unless defined otherwise by the application.

NOTE The command (sequence) for retrieving the content depends on the structure of EF.ATR/INFO (e.g. see Table 116) and may be known implicitly by an application in the outside world.

If EF.ATR/INFO supports DO handling, or if its contents are not stored in a file, a card may provide its contents by the GET DATA command (see 11.4.3). The response data to the command '00CB 2F01 02 5C00 00' is the concatenation of all DOs which are present in EF.ATR/INFO, or would be present if EF.ATR/INFO does not actually exist.

If the card provides this service, the command should be successful at least immediately after enabling a physical interface (see 5.1).

NOTE Information presented in an ATR at interface enabling may supersede information form EF.ATR/INFO, if present.

12.2.3 Application identifier

Referenced by a compact header set to 'FY' in the historical bytes (see 12.1.1.4), or by tag '4F' in the initial data string (see 12.1.2), in EF.ATR/INFO, in EF.DIR and in the management data of any DF (see 7.4), this interindustry data element identifies an application.

An application identifier (AID) consists of up to sixteen bytes. Bits b8 to b5 of the first byte indicate a category according to Table 121.

Table 121 — Categories of application identifiers

Value	Category	Meaning
'0' to '9'	-	Reserved for backward compatibility with ISO/IEC 7812-1 [5] (see Annex D)
'A'	International	International registration of application providers according to ISO/IEC 7816-5
'B', 'C'	-	RFU
'D'	National	National (ISO 3166-1 [3]) registration of application providers according to ISO/IEC 7816-5
'E'	Standard	Identification of a standard by an object identifier according to ISO/IEC 8825-1
'F'	Proprietary	No registration of application providers

Figure 10 shows an international AID. It consists of a registered application provider identifier (international RID) on five bytes and optionally, a proprietary application identifier extension (PIX) on up to eleven bytes.

— The international RID shall uniquely identify an application provider (see ISO/IEC 7816-5).

  — Bits b8 to b5 of the first byte shall be set to 1010, i.e. the first quartet shall be set to 'A'.

  — Each one of the subsequent nine quartets shall be set from '0' to '9'.

— The extension has a free encoding. It allows the application provider to identify its different applications.

Registered application provider identifier (International RID, five bytes, first byte set to 'AX')	Proprietary application identifier extension (PIX, up to eleven bytes)

Figure 10 — International AID

Figure 11 shows a national AID. It consists of a registered application provider identifier (national RID) on five bytes and optionally, a proprietary application identifier extension (PIX) on up to eleven bytes.

— The national RID shall uniquely identify an application provider (see ISO/IEC 7816-5).

  — Bits b8 to b5 of the first byte shall be set to 1101, i.e. the first quartet shall be set to 'D'.

  — The subsequent three quartets (from '0' to '9') shall form a country code (see ISO 3166-1 [3]).

  — The recommended value of each one of the last six quartets is from '0' to '9'.

— The extension has a free encoding. It allows the application provider to identify its different applications.

Registered application provider identifier (National RID, five bytes, first byte set to 'DX')	Proprietary application identifier extension (PIX, up to eleven bytes)

Figure 11 — National AID

Figure 12 shows a standard AID. It consists of up to sixteen bytes. The first byte shall be set to 1110 1000, i.e. to 'E8'. The values 'E0' to 'E7' and 'E9' to 'EF' are RFU. An object identifier (see ISO/IEC 8825-1) shall follow for identifying a standard specifying an application (see examples in Annex A, e.g. ISO/IEC 7816-11, personal verification through biometric methods, ISO/IEC 7816-15, cryptographic information application). An application identifier extension (specified according to the identified standard) may follow for identifying different implementations.

'E8'	Object identifier (see Annex A)	Application-specific application identifier extension

Figure 12 — Standard AID

Figure 13 shows a proprietary AID. It consists of up to sixteen bytes. Bits b8 to b5 of the first byte shall be set to 1111, i.e. to 'F'. In the proprietary category, as application providers are not registered, different application providers may use the same AID.

Proprietary application identifier (Proprietary AID, up to sixteen bytes, first byte set to 'FX')

Figure 13 — Proprietary AID

CAUTION —An AID being shorter than five bytes conforms to this document. Some implementations may require AIDs to be at least five bytes long. This should be taken into account when assigning an AID.

12.2.4 Application template and related data elements

Referenced by tag '61', this interindustry template may be present in EF.ATR/INFO (see 12.2.2), in EF.DIR (see 12.2.1) and in the management data of any DF (see 7.4).

— Such a template shall contain one and only one application identifier. If several application identifiers are valid names for the same DF, then each one should be present in a different application template.

— Such a template may optionally contain other interindustry DOs relating to the application as listed in Table 122 and defined hereafter.

Table 122 — Interindustry DOs for application identification and selection

Tag	Value	
'4F'	Application identifier	
'50'	Application label	
'51'	File reference	
'52'	Command APDU	
'53', '73'	Discretionary data, discretionary template	
'5F50'	Uniform resource locator (see IETF RFC 1738 [26] and IETF RFC 2396 [27])	
'61'	Set of application-related DOs	
	'79'	under DO'61' this DO indicates a coexistent tag allocation scheme

The following interindustry data elements provide a generic support to application identification and selection.

Application label — Referenced by tag '50', this interindustry data element is not defined in ISO/IEC 7816 [6]. The application provider defines it for use at the man-machine interface, e.g. a trademark to display.

File reference — Referenced by tag '51' (see 7.3.2)

Discretionary data (or template) — Referenced by tag '53' (or '73'), this interindustry data element (or template) consists of relevant data elements (or nests DOs) defined by the application provider.

Uniform resource locator — Referenced by tag '5F50', this interindustry data element is a uniform resource locator (URL) as defined in IETF RFC 1738 [26] and IETF RFC 2396 [27]. It points to part of the software required in the interface device to communicate with the application in the card.

12.2.5 Application selection

The card shall support at least one of the following application selection methods.

1) Implicit application selection

2) Application selection using an application identifier (AID, see 12.2.3) as DF name

3) Application selection using EF.DIR or EF.ATR/INFO

12.2.5.1 Implicit application selection

If an application is implicitly selected as a consequence of enabling a physical interface, then an application identifier should be present in the historical bytes (see 12.1.1) or in the initial data string (see 12.1.2). Such a presence denotes an implicitly selected application. If an application is implicitly selected with no application identifier in the historical bytes and in the initial data string, then an application identifier shall be present in EF.ATR/INFO (see 12.2.2).

12.2.5.2 Application selection using AID as DF name

A multi-application card shall support the SELECT command with P1='04', P2='00' and a data field containing 5 to 16 bytes with the AID of an application that may reside on the card. The command shall complete successfully if the AID of an application the card holds matches the data field.

If the card capabilities (see 12.1.1.9 and Table 117) specify the card supports selection by truncated AID it shall support the SELECT command with P1='04', P2='00' or '02'. In this case the command shall complete successfully if the first part of the AID of any application it holds matches the data field. If multiple AIDs in the card match the input data, the order in which each of these applications is actually selected after completion of the command is implementation-dependent. If the command completes successfully with a partial match of the AID it shall return the full AID of the selected application in the file control or management data (as DO'84' or DO'4F').

The card may support a mechanism for an application to specify a requirement for a full match of the AID in order to be successfully selected. The SELECT command shall fail if the application AID matches a truncated AID as the only selectable application in the card, or ignore the application if other application AIDs match and can be selected.

NOTE The order in which applications in the card are selected by consecutive SELECT commands with P1='02' may be static or dynamic, e.g. based on which application was selected most recently in a previous session.

In a multi-application card an application in the card shall be identified by

— a single AID in the proprietary, national or international category, and/or

— one or more AIDs in the standard category.

If the application is selected by specifying an AID in the standard category, the AID returned by the SELECT command is the AID in the proprietary, national or international category, if such an AID is specified for the application.

12.2.5.3 Application selection using EF.DIR or EF.ATR/INFO

For a multi-application interface device, the use of EF.DIR or EF.ATR/INFO may be more efficient than the previous method.

— If an application identifier DO is not part of an application template and not accompanied by a file reference or command-to-perform DO, then the selection shall use AID as DF name.

— If an application identifier DO is part of an application template together with a file reference DO (see 7.3.2), the value field of which consists of two or more bytes, then the selection by path shall be performed according to 12.3.

— If an application identifier DO is part of an application template together with one or more command-to-perform DOs, then the application selection is done by the indicated command(s). If several, they shall be performed in the order presented in the template.

12.3 Selection by path

This service allows selection of EFs and DFs having file identifiers by using a path, i.e. a file reference DO (see 7.3.2) consisting of three or more bytes.

— When the length is even, the path is either absolute or relative depending on whether the first two bytes are set to '3F00' or not. The last two bytes identify either a DF or an EF.

  — For a path to a DF, the selection should be done by one or more SELECT commands, with CLA INS P1 P2 L_c set to '00A4 0100 02'.

  — For a path to an EF, if the length is four or more, the selection should be done by one or more SELECT commands, with CLA INS P1 P2 L_c set to '00A4 0100 02'. The last and possibly only selection uses the last two bytes of the path (an EF identifier) with CLA INS P1 P2 L_c set to '00A4 0200 02'.

— When the length is odd, the path is qualified. It consists of either an absolute path without '3F00', or a relative path without the identifier of the current DF, followed by a byte to use as P1 in one or more SELECT commands. The value of P1 fixes the selection method.

  — If the value of P1 is '08' or '09', then the card shall support a SELECT command where the qualified path specifies P1, L_c and the data field and where P2 is set to '00'.

  — In the other cases, the card shall support one or more SELECT commands with P1 set to the last byte of the qualified path and P2 L_c set to '0002'. Every file along the path shall be selected sequentially.

12.4 Data retrieval

This service allows the interface device to retrieve interindustry data elements used for interchange, before selecting an application. Interindustry DOs should be retrieved directly or indirectly from the historical bytes (see 12.1.1), the initial data string (see 12.1.2), EF.ATR/INFO (see 12.2.2) and EF.DIR (see 12.2.1), in that order, when present. These interindustry DOs shall be interpreted according to tag allocation schemes (see 8.3). Standards are entitled to recommend or mandate retrieval of interindustry DOs by GET DATA (see 11.4.3).

12.5 Card-originated byte strings

This service allows the card to originate byte strings. For clarity, this clause defines a query as (part of) a card-originated byte string and a reply as (part of) a response byte string sent by an entity in the outside world; for example, a complete set of queries may form a command APDU and a complete set of replies a response APDU, thus allowing communication service from card to interface device and also, from card to card, possibly through a network.

This clause specifies the following three features.

— How the card shall use SW1-SW2 as a trigger indicating that the card wants to issue a query, for which the card possibly expects a reply.

— How the interface device shall use the GET DATA command with even INS code (see 11.4.3) for retrieving a query from the card and the PUT DATA command with even INS code (see 11.4.6) for transmitting a reply, if any, to the card. Such GET DATA and PUT DATA commands shall set P1-P2 to '0000' (see Table 85).

— How queries and replies shall be formatted.

12.5.1 Triggering by the card

SW1-SW2 set to '62XX' with the value of 'XX' from '02' to '80' means that the card has a query of 'XX' bytes that the interface device should retrieve and for which the card possibly expects a reply.

SW1-SW2 set to '64XX' with the value of 'XX' from '02' to '80' means that the card aborted the command; a possible completion of the command is conditioned by the recovery of a query of 'XX' bytes, for which the card possibly expects a reply.

If present in the historical bytes with a value such as above, SW1-SW2 shall be interpreted as above.

If a PUT DATA command (see 12.5.1, dash 2) for transmitting a reply is aborted with SW1-SW2 set to '64XX', then

— with '64XX' from '6402' to '6480', the card wants to send at least one more query of 'XX' bytes;

— with '64XX' set to '6401', the card is expecting an immediate reply.

12.5.2 Queries and replies

For retrieving a query of 'XX' bytes available in the card, the interface device shall send a GET DATA command with INS set to 'CA', P1-P2 set to '0000' and an L_e field set to 'XX'.

— SW1-SW2 set to '62XX' with the value of 'XX' from '02' to '80' means that the interface device should retrieve a further query of 'XX' bytes and concatenate it to the already retrieved query before processing the card-originated byte string in the outside world.

— SW1-SW2 set to '9000' means that the card-originated byte string is complete; it may be processed in the outside world.

For transmitting a reply to the card, the interface device shall send a PUT DATA command with INS set to 'DA', and P1-P2 set to '0000'. If the response byte string is too long for a single command, then several PUT DATA commands shall be chained (see 5.3.3). Each PUT DATA command transmits a reply and the concatenation of the replies is the response byte string.

12.5.3 Formats

The value of the first byte of the card-originated byte string indicates a format as follows.

— If the first byte is set to 'FF', then the subsequent bytes shall encode an initial protocol identifier according to ISO/IEC TR 9577 [7]; the byte strings shall comply with the indicated protocol.

— Otherwise (i.e. when the first byte is not set to 'FF'), the card-originated byte string and the response together shall form a C-RP.

All conditions are relevant to the transmission protocol indicated by the card, except for the proper use of GET DATA command, PUT DATA command and status bytes SW1-SW2. This clause makes no assumption on the need for a response and on the entity responsible for the contents of the possible response.

12.6 General feature management

This service allows the card to inform the external world about existing features on the card in a generic way. A template (DO'7F64') which is possibly extendable in the future by adding additional sub-templates is located in the card, e.g. in the EF.ATR/INFO (see 12.2.2) and/or in the FCI of any application DF. In general the

information may apply to all applications. Values specified in application FCI apply to that application only, possibly superseding values specified in the EF.ATR/INFO. EF.ATR/INFO or an application FCI shall not contain more than one instance of DO'7F64'. The information may be retrieved by reading EF.ATR/INFO or in the response data field of a command selecting an application.

This service supports the retrieval of condensed information about installed features and applications. An interface device may retrieve additional information about installed features and applications as defined by other standards.

12.6.1 On-card services

This service allows the card to inform the external world about existing on-card features. Nested in the feature management template this sub-template consists of a sequence of consecutive octets of bits. Each bit indicates an on-card feature/mechanism. A set bit indicates the existence of this feature on the card. Table 123 shows already predefined on-card services.

12.6.2 Interface services

International standards define a great variety of interfaces which might be active in parallel. The communication mechanisms of each protocol do not offer any possibilities to indicate the existence and the parallel usage of multi interfaces. This sub-template in the feature management template offers the information about existing protocols and the way of communication handling (see Table 123).

Table 123 — Template for Features Management DO'7F64'

Tag	Length	Meaning								
'81'	Var.	Sub-Template Identifier for On-Card Services								
		Feature-List [0..n], expandable								
		b8	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**Meaning of bits in the first byte**
		1	-	-	-	-	-	-	-	Display
		-	1	-	-	-	-	-	-	Biometric Input Sensor
		-	-	x	x	x	x	x	x	000000 (any other value is RFU)
'82'	Var.	Sub-Template Identifier for Interface Services								
		Communication Feature-List [0..n], expandable								
		b8	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**Meaning of bits in the first byte**
		1	-	-	-	-	-	-	-	simultaneous uses of indicated interfaces possible
		0		-	-	-	-	-	-	activation of at most one of the indicated interfaces possible
		-	-	-	-	-	-	-	1	T=0 according to ISO/IEC 7816-3
		-	-	-	-	-	-	1	-	T=1 according to ISO/IEC 7816-3
		-	-	-	-	-	1	-	-	contactless according to ISO/IEC 14443
		-	-	-	-	1	-	-	-	USB according to ISO/IEC 7816-12
		-	-	-	1	-	-	-	-	SWP according to ETSI TS 102 613, TS 102 622
		-	x	x	-	-	-	-	-	00 (any other value is RFU)

12.6.3 Profile services

This standard defines short cut services for applications, e.g. application selection by AID or path (see 12.2.5.2 and 12.3) Many applications, especially supporting legacy versions, have to supply different

hardware chips with a variety of features. Terminals of these applications need to identify the application and the possible feature set of the card in a very fast way.

This identification is provided by the profile service, e.g. in an EF.DIR.

Table 124 — Interindustry DO for application profile identification

Tag	Value
'73'	Application Profile Indicator

12.6.4 Provision of additional information

Detailed information on e.g. peripherals or protocols may be present in the general feature management template. This information shall be provided in a constructed DO having a tag within the context specific class. Tag and content of which may be defined in other standards or specifications.

12.7 APDU management

12.7.1 Extended length information

The presence of extended length information (DO'7F66') is indicated in the third software function table of the card capabilities (see

Table 119).

DO'7F66' may be present in EF.ATR/INFO (see 12.2.2) and/or in the FMD of any Application DF. In the former case, the information applies to all applications. Values specified in the application FMD only apply to that application, possibly superseding values specified in the EF.ATR/INFO.

EF.ATR/INFO shall contain at most one instance of DO'7F66'; an application FMD shall contain at most one instance of DO'7F66'. All DOs nested in DO'7F66' have a variable length.

The command- and response-APDU size limitations are defined by two integers, each nested in a DO'02'. Under tag '7F66':

— The first DO'02' shall contain a positive integer. The number of bytes in a command APDU shall not exceed this number.

— The second DO'02' shall contain a positive integer. If the card does not support response chaining for a particular C-RP then in such a C-RP N_e shall be set such that the number of bytes in a response APDU does not exceed this number.

12.7.2 List of supported INS codes

An INS list DO'5F63' may be present in EF.ATR/INFO (see 12.2.2) and/or in the FMD of any Application DF. In the former case, the information applies to all applications. Values specified in application FMD only apply to that application, possibly superseding values specified in the EF.ATR/INFO.

If DO'5F63' is present then EF.ATR/INFO or any application FMD shall contain at most one instance.

The INS list DO'5F63' nests a concatenation of INS bytes supported by the card or application.

Annex A
(informative)

Examples of object identifiers and tag allocation schemes

A.1 Object identifiers

For ISO standards, the first byte is '28', i.e. 40 in decimal (see ISO/IEC 8825-1). One or more series of bytes follow; bit b8 is set to 0 in the last byte of a series and to 1 in the previous bytes, if there is more than one byte. The concatenation of bits b7 to b1 of the bytes of a series encodes a number. Each number shall be encoded on the fewest possible bytes, that is, the value '80' is invalid for the first byte of a series. The first number is the number of the standard; the second number, if present, is the part in a multi-part standard.

As a first example, {iso(1) standard(0) ic-cards(7816)} references ISO/IEC 7816 [6].

— 7816 is equal to '1E88', i.e. 0001 1110 1000 1000, i.e. two blocks of seven bits: 0111101 0001000.
— After insertion of the appropriate value of bit b8 in each byte, the encoding of the first series is therefore 1011 1101 0000 1000, equal to 'BD08'.

The data element '28 BD08' may be used in AIDs of standard category (see 12.2.3).
AID = 'E8 28 BD08 0B XX … XX' (ISO/IEC 7816-11 specifies the application identifier extension 'XX … XX').
AID = 'E8 28 BD08 0F XX … XX' (ISO/IEC 7816-15 specifies the application identifier extension 'XX … XX').

As a second example, {iso(1) standard(0) e-auth(9798) part(5)} references ISO/IEC 9798 [10]. The first series is obtained as follows.

— 9798 is equal to '2646', i.e. 0010 0110 0100 0110, i.e. two blocks of seven bits: 1001100 1000110.
— After insertion of the appropriate value of bit b8 in each byte, the encoding of the first series is therefore 11001100 01000110, equal to 'CC46'.

The data element '28 CC46 05 02' references the second mechanism in ISO/IEC 9798 [10], i.e. GQ2. Such an identifier may be conveyed in a DO (tag '06', universal class, see ISO/IEC 8825-1).
DO = {'06 05 28 CC 46 05 02'}

As a third example, {iso(1) standard(0) mess(9992) part(2)} references ISO 9992-2 [12]. The first series is obtained as follows.

— 9992 is equal to '2708', i.e. 0010 0111 0000 1000, i.e. two blocks of seven bits: 1001110 0001000.
— After insertion of the appropriate value of bit b8 in each byte, the encoding of the first series is therefore 1100 1110 0000 1000, equal to 'CE08'.

The data element is '28 CE08 02' (the second series is '02'). It may be conveyed in a DO.
DO = {'06 04 28 CE 08 02'}

A.2 Tag allocation schemes

Example of default tag allocation scheme

DO1 = {'59 02 95 02'}
DO2 = {'5F 24 03 97 03 31'}

DO1 (tag '59', card expiration date) encodes February 1995 as card expiration date (see ISO/IEC 7816-6).
DO2 (tag '5F24', application expiration date) encodes March 31st 1997 as application expiration date.

Example of compatible tag allocation scheme

DO1 = {'78 06' {'06 04 28 CE 08 02'}}
DO2 = {'5F 24 03 97 03 31'}
DO3 = {'70 04' {'80 02 XX XX'}}
DO4 = {'67 06' {'5F 29 03 XX XX XX'}}

DO1 (tag '78', compatible tag allocation authority) indicates a compatible tag allocation scheme defined in ISO 9992-2 [12] referenced by its object identifier. If DO1 appears either in the initial data string (see 12.1.2), or in EF.ATR/INFO (see 12.2.2), then the tag allocation authority is valid for the entire card. If DO1 appears in the management data of a DF (see 7.4), then the tag allocation authority is valid within that DF.

DO2 (tag '5F24', application expiration date) encodes March 31st 1997 as application expiration date.

DO3 (tag '70', interindustry template according to the included tag allocation authority) contains a DO, tag '80', defined in ISO 9992-2 [12]; the meaning of tag '70' is also defined in ISO 9992-2 [12].

DO4 (tag '67', authentication data template) contains the interchange profile DO, tag '5F29'.

Another example of compatible tag allocation scheme

DO1 = {'5F 24 03 97 03 31'}
DO2 = {'70 0C' {'06 04 28 CE 08 02'} {'80 04 XX XX XX XX'}}
DO3= {'67 06' {'5F 29 03 XX XX XX'}}

DO1 (tag '5F24', application expiration date) encodes March 31st 1997 as application expiration date.

DO2 (tag '70', interindustry template defined according to the included object identifier) contains a DO, tag '06', which specifies that the subsequent DO, tag '80', is defined in ISO 9992-2 [12]. The meaning of tag '70' is also defined in ISO 9992-2 [12].

DO3 (tag '67', interindustry authentication data template) contains the interchange profile DO, tag '5F29'. Note that it cannot contain DOs defined in ISO 9992-2 [12], because of the choice not to transmit the interindustry DO with tag '78'.

Example of coexistent tag allocation scheme

DO1 = {'79 05' {'06 03 28 XX XX'}}
DO2 = {'7E 06' {'5F 24 03 97 03 31'}}
DO3 = {'70 06' {'XX XX XX XX XX XX'}}

DO1 (tag '79', coexistent tag allocation authority) indicates a coexistent tag allocation scheme defined in a standard referenced by an object identifier starting with '28', therefore an ISO standard. Mandatory in such a scheme, DO1 shall appear either

- in the initial data string (see 12.1.2) or in EF.ATR/INFO (see 12.2.2) if the tag allocation authority is valid for the entire card, or
- in the management data of a DF (see 7.4) if the tag allocation authority is valid within that DF.

DO2 (tag '7E') is an interindustry template for nesting interindustry DOs. Note that the interindustry DO "application expiration date", tag '5F24', is present, encoding March 31st 1997 as application expiration date.

DO3 (tag '70', interindustry template to be interpreted according to the tag allocation authority indicated in template '79') can only be interpreted according to the standard indicated in the object identifier.

Annex B
(informative)

Examples of secure messaging

B.1 Cryptographic checksum

This clause shows the use of secure messaging (see clause 10) and cryptographic checksums (see 10.2.3.1) for each of the four cases of C-RPs defined in ISO/IEC 7816-3.

In the examples, the notation T* means that bit b1 of the last byte of the tag field is set to 1 (an odd tag number), i.e. that the SM DO shall be included in the computation of a data element for authentication.

In the examples, the notation CLA* means the use of secure messaging in the data fields: in CLA (see 5.4.1), either bits b8, b7 and b6 set to 000 and bit b4 set to 1, or bits b8, b7 and b6 set to 011.

In the examples, the notation CLA** means that bits b8, b7 and b6 of CLA are set to 000 and bits b4 and b3 to 11, i.e. that the command header shall be included in the computation of a data element for authentication.

Alternatively the header may be encapsulated in a DO with tag '89', i.e. an SM DO to be included in the computation of a data element for authentication.

Case 1 shows how bit b3 in the first coding of CLA (see Table 2) mandates the protection of the command header by a cryptographic checksum, and how the function is optionally supported when using the second coding of CLA (see Table 3). In Case 1, protection of course applies. The use of bit b3 in the first coding of CLA it is not shown in other cases, to simplify the examples, and because the outcome is always the same: add one block at the beginning of the data covered by the cryptographic checksum of a command APDU.

Case 1 — No command data, no response data

Command header	Command body
CLA INS P1 P2	Absent

Response body	Response trailer
Absent	SW1-SW2

Case 1.a — Status not protected

The assumption is made that the command header is to be protected, as it is the only item to protect in this C-RP.

Command header	Command body	b7 in CLA*
CLA* INS P1 P2	New L_c field – New data field = {T - L - Cryptographic checksum}	0
	New L_c field - New data field= {T* - L - Command header} {T - L - Cryptographic checksum}	1

If the length of the cryptographic checksum is four bytes, then the new L_c field is set to '06' (top line) or '0C' (bottom line).

New data field = One or two DOs = conditional {T* - L - Command header} then {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum:

— first coding of CLA, one block = [CLA** INS P1 P2 Padding]

— second coding of CLA, one block = [{T* - L - Command header} - Padding]

The secured response APDU is as follows.

Response body	Response trailer
Absent	SW1-SW2

Case 1.b — Status protected

The secured command APDU is as follows.

Command header	Command body	b7 in CLA*
CLA* INS P1 P2	New L_c field - New data field = {T - L - Cryptographic checksum} -New L_e field = '00'	1
	New L_c field - New data field = {T - L - Command header} {T - L - Cryptographic checksum} - New L_e field = '00'	0

New data field = One or two DOs =
conditional {T - L - Command header} always ending with {T - L - Cryptographic checksum}

The difference with case 1.a is the New L_e field = '00', because the cryptographic checksum belongs to response data which must be required by the presence of L_e. The data covered by the cryptographic checksum are the same as in case 1.a.

The secured response APDU is as follows.

Response body	Response trailer
New data field = {T* - L - SW1-SW2} - {T - L - Cryptographic checksum})	SW1-SW2

New data field = Two DOs = {T* - L - SW1-SW2} - {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum = One block = [{T* - L - SW1-SW2} – Padding]

Case 2 — No command data, response data

The unsecured C-RP is as follows.

Command header	Command body
CLA INS P1 P2	L_e field

Response body	Response trailer
Data field	SW1-SW2

The secured command APDU is as follows.

Command header	Command body
CLA* INS P1 P2	New L_c field - New data field - New L_e field (one or two bytes set to '00')

New data field = Two DOs = {T* - L - N_e} - {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum = One block = [{T* - L - N_e} - Padding].

NOTE If the original L_e field is '00' or '000000', the value field may be absent, i.e. the respective DO will be '97 00' (see 10.4).

The secured response APDU is as follows.

Response body	Response trailer
New data field	SW1-SW2

New data field = Three DOs =
{T* - L - Plain value} - {T* - L - SW1-SW2} - {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum = one or more blocks, depending on the plain value, e.g.:

— One block = [{T* - L - Plain value} {T* - L - SW1-SW2} - Padding]

— Two blocks = [{T* - L - Plain value, start] - [Plain value, end} - {T* - L - SW1-SW2} - Padding]

Case 3 — Command data, no response data

The unsecured C-RP is as follows.

Command header	Command body
CLA INS P1 P2	L_c field - Data field

Response body	Response trailer
Absent	SW1-SW2

Case 3.a — Status not protected

The secured command APDU is as follows.

Command header	Command body
CLA* INS P1 P2	New L_c field - New data field

New data field or new command payload = Two DOs
= {T* - L - Plain value} - {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum = one or more blocks, depending on the plain value:

1. [{T* - L - Plain value} - Padding]
2. [{T* - L - Plain value}] - [Padding]
3. [{T* - L - Plain value, start] - [Plain value, end} - Padding]

In example 1, with 8-byte blocks, the maximum L in the plain value DO is 5, in which case the block ends with one byte of padding.

In example 2, with 8-byte blocks, L in the plain value DO is 6. The DO fills the block, which must be followed by a full block of padding.

In example 3, with 8-byte blocks, L in the plain value DO is between 7 and 13, to leave at least one byte of padding at the end of the second block.

The secured response APDU is identical to the unsecured response APDU:

Response body	Response trailer
Absent	SW1-SW2

Case 3.b — Status protected

The secured command APDU is as follows.

Command header	Command body
CLA* INS P1 P2	New L_c field - New data field - New L_e field (= '00')

New data field or new command payload = Two DOs
= {T* - L - Plain value} - {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum =one or more blocks, depending on the plain value.

4. [{T* - L - Plain value}- Padding]
5. [{T* - L - Plain value, start}] - [Padding]
6. [{T* - L - Plain value, start] - [Plain value, end} - Padding]

In example 4, with 8-byte blocks, the maximum L in the plain value DO is 5, in which case the block ends with one byte of padding.

In example 5, with 8-byte blocks L in the plain value DO is 6. The DO fills the block, which must be followed by a full block of padding.

In example 6, with 8-byte blocks L in the plain value DO is between 7 and 13, to leave at least one byte of padding at the end of the second block.

The secured response APDU is as follows.

Response body	Response trailer
New data field (= {T* - L - SW1-SW2} - {T - L - Cryptographic checksum})	SW1-SW2

New data field = Two DOs = {T* - L - SW1-SW2} - {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum = One block = [{T* - L - SW1-SW2} - Padding]

Case 4 — Command data, response data

The unsecured C-RP is as follows.

Command header	Command body
CLA INS P1 P2	L_c field - Data field - L_e field

Response body	Response trailer
Data field	SW1-SW2

The secured command APDU is as follows.

Command header	Command body
CLA* INS P1 P2	New L_c field - New data field - New L_e field (one or two bytes set to '00')

New data field or new command payload = Three DOs
= {T* - L - Plain value} - {T* - L - N_e} - {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum are the same as in case 3b.

The secured response APDU is as follows.

Response body	Response trailer
New data field (={T* - L - Plain value} - {T* - L - SW1-SW2} - {T - L – Cryptographic checksum})	SW1-SW2

New data field = Three DOs =
{T* - L - Plain value} - {T* - L - SW1-SW2} - {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum = One or more blocks =
[{T* - L - Plain value} - {T* - L - SW1-SW2} – Padding]

B.2 Cryptograms

The use of cryptograms with and without padding (see 10.2.2) is shown in command and response data fields. Instead of the plain value DOs in the previous examples, DOs for confidentiality shall be used as follows.

1) Case a — Plain value not encoded in BER-TLV

Data field = {T - L - Padding-content indicator byte - Cryptogram}

Plain value conveyed by the cryptogram = One or more blocks =
Plain value not encoded in BER-TLV, possibly padded according to the indicator byte

2) Case b — Plain value encoded in BER-TLV

Data field = {T - L - Cryptogram}

Plain value conveyed by the cryptogram = String of enciphered bytes =
BER-TLV DOs (padding depending on the algorithm and its mode of operation)

B.3 Control references

The use of control references (see 0 and 10.3.2) is shown.

Command data field = {T - L - Control reference template},
where control reference template = {T - L - File reference} - {T - L - Key reference}

B.4 Response descriptor

The use of response descriptor (see 10.3.4) is shown.

Command data field = {T - L - Response descriptor}
where response descriptor = {T (Plain value) - '00' - T (Cryptographic checksum) - '00'}

Response data field = {T - L - Plain value} - {T - L - Cryptographic checksum}

B.5 ENVELOPE command

The use of the ENVELOPE command (see 11.7.2) is shown.

Command data field = {T - L - Padding-content indicator byte - Cryptogram}

Plain value conveyed by the cryptogram =
Command APDU (starting by CLA* INS P1 P2), padding according to the indicator byte

Response data field = {T - L - Padding-content indicator byte - Cryptogram}

Plain value conveyed by the cryptogram =
Response APDU, padding according to the indicator byte

B.6 Synergy between secure messaging and security operations

For the purposes of this clause, the following symbols and abbreviated terms apply.

CC	cryptographic checksum
CG	cryptogram
CLA**	CLA with SM indication (bits b8, b7 and b6 set to 000 and bits b4 and b3 set to 11)
DS	digital signature
MSE	manage security environment
PCI	padding-content indicator byte
PSO	perform security operation
SMC	security module card
USC	user smart card

The example explains how to use a security module card (SMC) that performs security operations for producing a secured command APDU to send to a user card (USC) and for processing the corresponding secured response APDU received thereon from the USC, i.e. for producing and processing data fields in SM format. The example illustrates the synergy between the two approaches: — the atomic approach by security operations (see ISO/IEC 7816-8) and — the global approach by secure messaging (see clause 10).

The example assumes that the USC and the SMC have completed a mutual authentication procedure, based e.g. on card verifiable certificates. The authentication procedure includes a key transport or key agreement mechanism so that after this procedure, two symmetric keys are available in the USC and in the SMC:

1. a symmetric session key for computing cryptographic checksums and
2. a symmetric session key for computing cryptograms.

The authentication procedure initialises one or more counters in the USC and the SMC. The example does not show the maintenance and the use of such counters by the USC and the SMC.

All the C-RPs for the SMC are PSO commands, not using secure messaging, but using SM DOs (and the SM keys set by MSE commands).

All the C-RPs for the USC use secure messaging and the command headers are included in the computation of cryptographic checksums, i.e. CLA is switched to CLA**.

NOTE The cryptographic operations performed by the commands PSO COMPUTE CC, PSO VERIFY CC and PSO ENCIPHER may require an input which has a length being a multiple of the block length of the cryptographic algorithm. In order to fulfill this requirement padding applies. This padding is applied inside the SMC. Furthermore the SMC removes padding bytes before responding to a PSO DECIPHER command. Thus padding at the end of input or output data does not occur in the figures and statements below.

Figure B.1 shows the general principles for producing a secured command APDU.

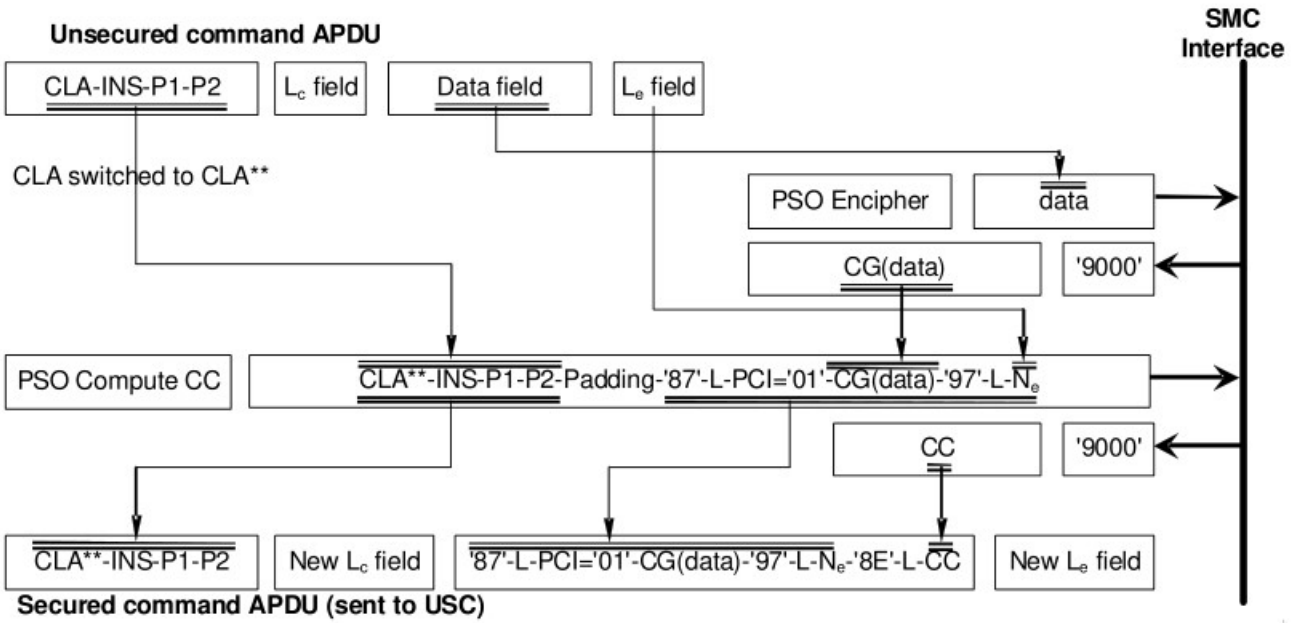


Figure B.1 — Producing a secured command APDU

Figure B.2 shows the general principles for processing a secured response APDU.

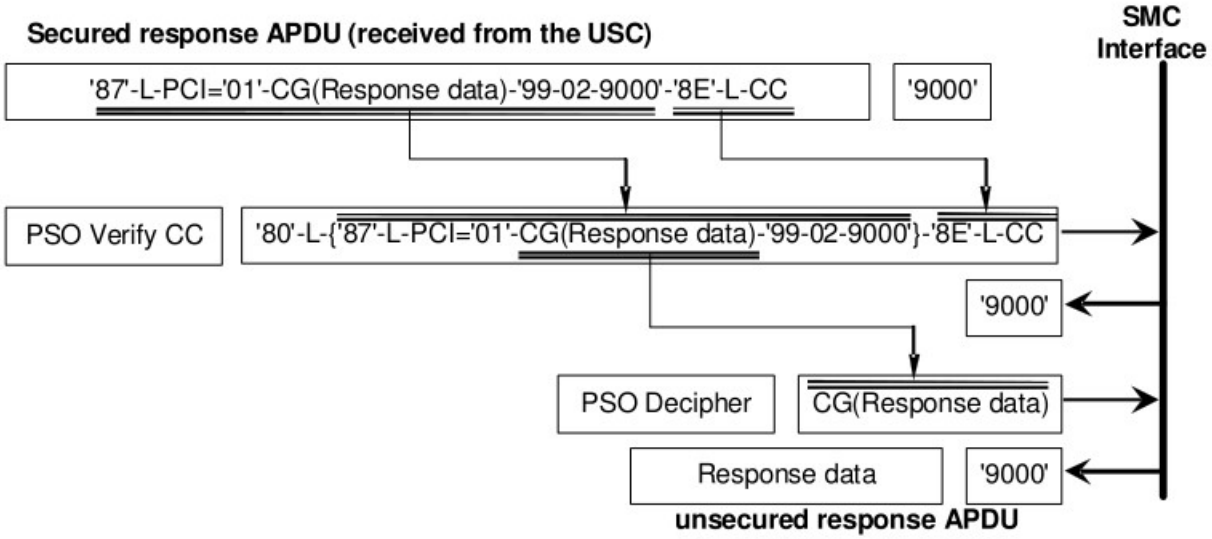


Figure B.2 — Processing a secured response APDU

The following scenario explains the computation of a digital signature (DS) whereby the usage of the private signature key requires the successful presentation of a password. The scenario proceeds in three steps.

Step 1 — Password verification

1.1 Command to SMC: MSE SET <CCT, {'83'-'01'-'81'}>
--- The reference of the session key for computing cryptographic checksums is '81' in the example.
SMC response: OK

1.2 Command to SMC: MSE SET <CT, {'83'-'01'-'82'}>
---The reference of the session key for computing cryptograms is '82' in the example.
SMC response: OK

1.3 Command to SMC: PSO ENCIPHER <Password>
SMC response: <CG (Password)>

1.4 Command to SMC: PSO COMPUTE CC <CLA**-INS-P1-P2 - Padding - {'87'-L-PCI-CG (Password)} - {'97'-L-N_e} >
SMC response: <CC>

--- Now the interface device constructs the secured VERIFY command APDU.

1.5 Command to USC: VERIFY < {'87'-L-PCI='01'-CG (Password)} - {'97'- L-N_e} - {'8E'-'04'-CC}>
USC response: < {'99'-'02'-SW1-SW2} - {'8E'-'04'-CC}>

1.6 Command to SMC: PSO VERIFY CC < {'80'-'04'-('99'-'02'-SW1-SW2)} - {'8E'-'04'-CC}>
SMC response: OK

Step 2 — Hash code computation

2.1 Command to SMC: PSO COMPUTE CC <CLA**-INS-P1-P2 - Padding - {'81'-L-({'90'-L-Intermediate Hash} - {'80'-L-Last block})} - {'97'- L-N_e} >
SMC response: <CC>

2.2 Command to USC: PSO HASH <{'81'-L1 (=4+L2+L3)-({'90'-L2- Intermediate Hash} - {'80'-L3-Last block})} - {'8E'-'04'-CC}>
--- The USC stores the hash code as an internal result for computing the digital signature later on.
USC response: <{'99'-'02'-SW1-SW2} - {'8E'-'04'-CC}>

2.3 Command to SMC: PSO VERIFY CC <{'80'-'04'-({'99'-'02'-SW1-SW2})} - {'8E'-'04'-CC}>
SMC response: OK

Step 3 — Digital signature computation

3.1 Command to SMC: PSO COMPUTE CC <CLA**-INS-P1-P2 - Padding - {'97'-'01'-'00'}>
SMC response: <CC>

3.2 Command to USC: PSO COMPUTE DS <{'97'-'01'-'00'} - {'8E'-'04'-CC}>
USC response: <'81'-L-DS - '8E'-'04'-CC>

3.3 Command to SMC: PSO VERIFY CC <{'80'-L1 (=2+L2)-('81'-L2-DS)} - {'8E'-'04'-CC}>
SMC response: OK

Annex C
(informative)

Examples of AUTHENTICATE functions by GENERAL AUTHENTICATE commands

Two or more GENERAL AUTHENTICATE C-RPs implement an AUTHENTICATE function.

If chaining is used, then CLA is set to 0xx1 xxxx in the first command of the chain up to the penultimate one and to 0xx0 xxxx in the last one: the other six bits shall remain constant within the chain (see 5.3.3).

INS P1 P2 is set to either '86 00 00', or '87 00 00'.

The value of the L_c field depends upon the DOs in the command data field. Depending upon whether a response data field is expected or not, the L_e field is either set to '00', or absent.

C.1 GENERAL AUTHENTICATE using witness-challenge-response triples

C.1.1 Introduction

This annex illustrates data fields of GENERAL AUTHENTICATE commands implementing mechanisms such as specified in ISO/IEC 9798-5 [10], i.e. mechanisms using zero-knowledge techniques.

— A verifier knows a public problem and a claimant knows a secret solution to the public problem.

— As a result of the zero-knowledge protocol, the verifier is convinced that the claimant knows a solution to the public problem. Moreover, the solution remains secret.

NOTE ISO/IEC 9798-5 [10] specifies two GQ techniques.

— Being given a public RSA key where the exponent v is prime such as 257 = 2^8+1, 65537 = 2^{16}+1 or 2^{36}+2^{13}+1, the GQ1 technique allows verifying an RSA signature without taking knowledge of its value, or alternatively, proving knowledge of an RSA signature without revealing its value. As specified by the RSA signature standard in use (e.g. see ISO/IEC 14888-2 [19]), a format mechanism converts the claimant's identification data (a template) into a public number G. The corresponding private number Q is the RSA signature of the identification data. The claimant and the verifier know the public RSA key. The GQ1 protocol proves that the claimant knows the RSA signature of his identification data.

— Being given a public modulus n, product of two prime factors, the GQ2 technique allows verifying the factors without taking knowledge of them, or alternatively, proving knowledge of the factors without revealing them. The mechanism involves a security parameter k > 0 and the first m prime numbers, named the m basic numbers, such that k×m is from 8 to 36. Each public number is the square of a basic number: $G = g^2$. The corresponding private number Q is a modular 2^{k+1}-th root of G. If there is at least one basic number g such that the Jacobi symbol of g with respect to n is –1 and if n is congruent to 1 mod 4, then the GQ2 protocol proves that n is composite and that the claimant knows the factors.

The protocol typically exchanges three numbers, namely a witness, a challenge and a response.

— The claimant works in two steps: as a first step, the claimant privately selects a fresh random number and converts it into a witness according to a “witness formula”; as a second step, having received a challenge, the claimant gets the response to the challenge from the fresh random number and the private number, according to a “response formula”, and then erases the fresh random number.

— The verifier reconstructs a witness from the challenge and the response, according to a “verification formula”.

By definition, a triple consists of three numbers, namely, a witness, a challenge and a response, verifying the verification formula. Any entity may randomly produce triples in “public mode”, from any challenge and response. A judge or an observer cannot distinguish random triples produced in public mode, i.e. by an entity not knowing the secret, and random triples produced in “private mode”, i.e. by an entity knowing the secret.

This part of this part annex illustrates three AUTHENTICATE functions.

— INTERNAL AUTHENTICATE function — A verifier in the outside world authenticates a claimant in the card.
— EXTERNAL AUTHENTICATE function — A verifier in the card authenticates a claimant in the outside world.
— MUTUAL AUTHENTICATE function — Both entities authenticate each other.

C.1.2 INTERNAL AUTHENTICATE function

If the first data field conveys a witness request, namely, either an empty witness ('80 00'), or an empty authentication code ('84 00'), then the function is INTERNAL AUTHENTICATE.

Basic protocol (two C-RPs)

Witness from the card

Command data field	{'7C'-'02'-{'80'-'00'}}
Response data field	{'7C'-L1 (=2+L2)-{'80'-L2-Witness}}

Challenge from the outside world and response from the card

Command data field	{'7C'-L1 (=4+L2)-{'81'-L2-Challenge}-{'82'-'00'}}
Response data field	{'7C'-L1 (=2+L2)-{'82'-L2-Response}}

Committed challenge (two C-RPs)

Witness from the card

Command data field	{'7C'-L1 (=4+L2)-{'83'-L2-Committed Challenge}-{'80'-'00'}}
Response data field	{'7C'-L1 (=2+L2)-{'80'-L2-Witness}}

NOTE The committed challenge ensures that the challenge and the witness are independently selected.

Challenge from the outside world and response from the card

Command data field	{'7C'-L1 (=4+L2)-{'81'-L2-Challenge}-{'82'-'00'}}
Response data field	{'7C'-L1 (=2+L2)-{'82'-L2-Response} if the challenge is correct Absent if the challenge is incorrect

Extension to data field authentication (two C-RPs)

The card has hashed previously exchanged data fields: the result is a current hash-code. The card includes its witness DO for getting an authentication code and transmits it with tag '84'.

Witness from the card

Command data field	{'7C'-'02'-{'84'-'00'}}
Response data field	{'7C'-L1 (=2+L2)-{'84'-L2-Authentication code}}

Challenge from the outside world and response from the card

Command data field	{'7C'-L1 (=4+L2)-{'81'-L2-Challenge}-{'82'-'00'}}
Response data field	{'7C'-L1 (=2+L2)-{'82'-L2-Response}}

C.1.3 EXTERNAL AUTHENTICATE function

If the first data field conveys a challenge request, namely, either an empty challenge ('81 00'), or an empty committed challenge ('83 00'), then the function is EXTERNAL AUTHENTICATE.

Basic protocol (two C-RPs)

	Witness from the outside world and challenge from the card
Command data field	{'7C'-L1 (=4+L2)-{'80'-L2-Witness}-{'81'-'00'}}
Response data field	{'7C'-L1 (=2+L2)-{'81'-L2-Challenge}}
	Response from the outside world and verification by the card
Command data field	{'7C'-L1 (=2+L2)-{'82'-L2-Response}}
Response data field	Absent

Committed challenge (three C-RPs)

	Committed challenge from the card
Command data field	{'7C'-'02'-{'83'-'00'}}
Response data field	{'7C'-L1 (=4+L2)-{'83'-L2-Committed challenge}-{'80'-'00'}}
	Witness from the outside world and challenge from the card
Command data field	{'7C'-L1 (=4+L2)-{'80'-L2-Witness}-{'81'-'00'}}
Response data field	{'7C'-L1 (=2+L2)-{'81'-L2-Challenge}}
	Response from the outside world and verification by the card
Command data field	{'7C'-L1 (=2+L2)-{'82'-L2-Response}}
Response data field	Absent

Extension to data field authentication (two C-RPs)

A claimant has hashed previously exchanged data fields: the result is a current hash-code. It includes its witness DO for getting an authentication code and transmits it with tag '84'.

	Witness from the outside world and challenge from the card
Command data field	{'7C'-L1 (=4+L2)-{'84'-L2-Authentication code}-{'81'-'00'}}
Response data field	{'7C'-L1 (=2+L2)-{'81'-L2-Challenge}}
	Response from the outside world and verification by the card
Command data field	{'7C'-L1 (=2+L2)-{'82'-L2-Response}}
Response data field	Absent

C.1.4 MUTUAL AUTHENTICATE function

If the first data field conveys no empty DO, then the function is MUTUAL AUTHENTICATE; the outside world requests the same DOs in the response data field as in the command data field.

Basic protocol (three C-RPs)

	Witness
Command data field	{'7C'-L1 (=2+L2)-{'81'-L2-Witness}}
Response data field	{'7C'-L1 (=2+L2)-{'81'-L2-Witness}}
	Challenge
Command data field	{'7C'-L1 (=2+L2)-{'81'-L2-Challenge}}
Response data field	{'7C'-L1 (=2+L2)-{'81'-L2-Challenge}}
	Response
Command data field	{'7C'-L1 (=2+L2)-{'82'-L2-Response}}
Response data field	{'7C'-L1 (=2+L2)-{'82'-L2-Response}} if the response is correct Absent if the response is incorrect

Committed challenge (four C-RPs)

	Committed challenge
Command data field	{'7C'-L1 (=2+L2)-{'83'-L2-Committed challenge}}
Response data field	{'7C'-L1 (=2+L2)-{'83'-L2-Committed challenge}}
	Witness
Command data field	{'7C'-L1 (=2+L2)-{'80'-L2-Witness}}
Response data field	{'7C'-L1 (=2+L2)-{'80'-L2-Witness}}
	Challenge
Command data field	{'7C'-L1 (=2+L2)-{'81'-L2-Challenge}}
Response data field	{'7C'-L1 (=2+L2)-{'81'-L2-Challenge}} if the challenge is correct Absent if the challenge is incorrect
	Response
Command data field	{'7C'-L1 (=2+L2)-{'82'-L2-Response}}
Response data field	{'7C'-L1 (=2+L2)-{'82'-L2-Response}} if the response is correct Absent if the response is incorrect

Extension to key agreement (four C-RPs)

A pair of exponential data elements allows the agreement of a session key (see ISO/IEC 11770 [17]).

The first C-RP exchanges dynamic authentication templates nesting an “exponential” data element. In the example, as no message has been previously exchanged during the session, the initial hash-code is a null block. Then the command data field, i.e. the first dynamic authentication template, is included for getting a current hash-code; then the response data field, i.e. the second dynamic authentication template is included for updating the current hash-code; the current hash-code should be the same for both entities. Finally a witness DO (not zero and not transmitted, different for each entity) is included for getting an authentication code (different for each entity).

The second C-RP exchanges dynamic authentication templates nesting authentication codes with tag '84'.

	Exponential
Command data field	{'7C'-L1 (=2+L2)-{'85'-L2-Exponential}}
Response data field	{'7C'-L1 (=2+L2)-{'85'-L2-Exponential}}
	Witness
Command data field	{'7C'-L1 (=2+L2)-{'84'-L2-Authentication code}}
Response data field	{'7C'-L1 (=2+L2)-{'84'-L2-Authentication code}}
	Challenge
Command data field	{'7C'-L1 (=2+L2)-{'81'-L2-Challenge}}
Response data field	{'7C'-L1 (=2+L2)-{'81'-L2-Challenge}}
	Response
Command data field	{'7C'-L1 (=2+L2)-{'82'-L2-Response}}
Response data field	{'7C'-L1 (=2+L2)-{'82'-L2-Response}} if the response is correct Absent if the response is incorrect

C.2 GENERAL AUTHENTICATE for a multi-step authentication protocol

C.2.1 Introduction

This part of the annex describes the usage of the GENERAL AUTHENTICATE command for a multi-step authentication protocol, i.e. the password based authentication scheme PACE v2 as described in EN 14890-1.

Thereby a multi-step authentication protocol contains several authentication operations, at least one or more GENERAL AUTHENTICATE: PERFORM KEY AGREEMENT (PKA) operations which initiate the generation, management and/or use of Diffie-Hellman key pairs and/or key parameters in the card, e.g. to establish secure channel between ICC and external world by means of password authentication only.

According to the specific protocol in use one ore more PKA operations are used to

— Encipher and/or map a nonce

— generate an asymmetric Diffie Hellman (DH) key pair for key agreement and/or

— pre-access an asymmetric DH key pair and/or key parameters previously generated in the card

— store a DH public key from the interface device (IFD) based on the same global parameters in the card,

— process a key agreement

— and provide respective ephemeral keys in the card, e.g. to be used for secure messaging.

The operation(s) are preceded by one or more MANAGE SECURITY ENVIRONMENT operation(s) in order to set the respective object identifier as well as the key generation related parameters (e.g. algorithm reference in the CRT AT according to the protocol res. protocol variant to be used (Extended Access Control (EAC), Simple Password Exponential Key Exchange (SPEKE), PACE v2 according to EN 14890-1 [1]).

One or more general authenticate C-RP(s) implement(s) an authentication operation.

— If chaining is used, see 5.3.3.

— INS P1 P2 is set to either '86 00 00', or '87 00 00'.

— The value of the L_c field depends upon the DOs in the command data field. Depending upon whether a response data field is expected or not, the L_e field is either set to '00', or absent.

C.2.2 Use of GENERAL AUTHENTICATE with even INS byte ('86'): Example Password base Protocol e.g. PACE v2

Step 1:	SET SE Environment (CRT AT)
Command data field	{ '80'- L- Value of object identifier for generic mapping with specific enciphering algorithm (e.g. TDES-3 or AES) and Mapping of Nonce (generic mapping or integrated mapping} { '83'- L- Reference of the password} { '84'- L- Reference of the domain parameters of the private Ephemeral key (session key) }
Response data field	{} – absent
Step 2:	Get Enciphered Nonce
Command data field	'7C' – L=0
Response data field	{{'7C' – L2 – {'80' – L – anywhere nonce)}
Step 3:	Map Nonce (e.g. Diffie-Hellman key exchange (generic mapping) or EC specific mapping to Point (integrated mapping)
Command data field	{{'7C' – L2 – '{81' – L – mapping data)}
Response data field	{{'7C' – L2 – {'82' – L – mapping data)} (generic mapping) or {} – absent (integrated mapping)
Step 4:	Perform Key Agreement (PKA)
Command data field	{{'7C' – L2 – {'83' – L – Ephemeral Public Key)}
Response data field	{{'7C' – L2 – {'84' – L – Ephemeral Public Key)}
Step 5:	Mutual Authentication
Command data field	{{'7C' – L2 – {'85' – L – Authentication token)}
Response data field	{{'7C' – L2 – {'86' – L – Authentication token}

C.2.3 Use of GENERAL AUTHENTICATE with even INS byte ('86'): Extended Access Control (EAC)

The EAC protocol is separated in two parts: Terminal Authentication (TA) allows the authentication of the interface device or remote server, whereas Chip Authentication (CA) proves identity of the card. Two versions are distinguished according EN 14890-1 [1]:

— EACv1 refers to the sequence CA-TA;

— EACv2 refers to the sequence TA-CA.

The Terminal Authentication protocol first imports the public key of the interface device into the card by means of card verifiable certificates. In a second step a challenge-response protocol is performed. The Chip Authentication protocol specifies a Diffie-Hellman key exchange with establishment of a secure channel.

NA.1.1Terminal Authentication protocol

Step 1:	SET SE Environment (CRT DST)	
Command data field	{ '83'– L – Reference of public key	
Response data field	{} – absent	
Step 2:	PERFORM SECURITY OPERATION (VERIFY CERTIFICATE)	
Command data field	{'7F4E' – L – certificate structure, defined in ISO/IEC 7816-8} {'5F37' – L – digital signature}	
Response data field	{} – absent	

The verification of certificates along a certificate chain may continue until the public key of respective certification authority is available in the card that issued the certificate and delivers the public key of the interface device. The EAC protocol family specifies a three level PKI, so that two subsequent certificate verifications become necessary.

Step 3:	SET SE Environment (CRT AT)	
Command data field	{ '80' – L – Value of Object Identifier}	EACv2.0 only
	{ '83' – L – Reference of public key}	
	{ '91' – L – Ephemeral public key of IFD for CA}	EACv2.0 only
	{ '67' – L – auxiliary data}	EACv2.0 only
Response data field	{} – absent	
Step 4:	GET CHALLENGE	
Command data field	{} – absent	
Response data field	{random data}	
Step 5:	EXTERNAL AUTHENTICATE	
Command data field	{response data – digital signature signing random and authentication data}	
Response data field	{} – absent	

NA.1.2Chip Authentication protocol

Step 1[1]:	SET SE Environment (CRT AT)	
Command data field	{ '80' – L – Value of Object Identifier}	
	{ '84' – L – Reference of private key}	
Response data field	{} – absent	
Step 2:	GENERAL AUTHENTICATE with the intention "Perform Key Agreement (PKA)"	
Command data field	{'7C' – L2 – {'80' – L – Ephemeral public key of IFD}}	
Response data field	{'7C' – L2 – {'81' – L – random data} – {'82' – L – Authentication token}}	EAC v2 only

[1] NOTE:

For EACv1 with 3DES only the following command is still in use for step 1. In this case Step2 will be omitted.

Step 1:	SET SE Environment (CRT KAT)	
Command data field	{ '91' – L – Ephemeral public key of IFD }	EACv1 only with TDES
	{ '84' – L – Reference of private key}	EACv1 only with TDES
Response data field	{} – absent	

Annex D
(informative)

Application identifiers using issuer identification numbers

D.1 Background information

In ISO/IEC 7816-5:1994, it was possible to use issuer identification numbers in application identifiers. This annex indicates the format of such AIDs.

D.2 Format

In any AID where bits b8 to b5 of the first byte are set from '0' to '9', the first and possibly only field shall be an issuer identification number according to ISO/IEC 7812-1 [5].

NOTE In ISO/IEC 7812-1:1993, an issuer identification number might consist of an odd number of quartets valued from '0' to '9'. Then it was mapped into a byte string by setting bits b4 to b1 of the last byte to 1111.

If a proprietary application identifier extension is present, then a byte set to 'FF' shall separate the two fields.

Figure D.1 shows an AID using an issuer identification number: it consists of up to sixteen bytes.

Issuer identification number according to ISO/IEC 7812-1 [5] (two or more bytes)	'FF'	Proprietary application identifier extension (PIX)

Figure D.1 — AID using an issuer identification number

Annex E
(informative)

BER Encoding Rules

E.1 BER-TLV tag fields

Bits b8 and b7 of the first byte of the tag field indicate a class (see ISO/IEC 8825-1).

— The value 00 indicates a DO of the universal class.

— The value 01 indicates a DO of the application class.

— The value 10 indicates a DO of the context-specific class.

— The value 11 indicates a DO of the private class.

Bit b6 of the first byte of the tag field indicates an encoding (see E.3).

If bits b5 to b1 of the first byte of the tag field are not all set to 1, then they encode a tag number from zero to thirty and the tag field consists of a single byte. Tag number 0 is precluded in the universal class.

Otherwise (bits b5 to b1 all set to 1), the tag field continues on one or more subsequent bytes.

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

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

— Bits b7 to b1 of the first subsequent byte, followed by bits b7 to b1 of each further subsequent byte, up to and including bits b7 to b1 of the last subsequent byte encode a tag number.

Table E.1 shows the first byte of the tag field. The value '00' is invalid.

Table E.1 — First byte of BER-TLV tag fields in ISO/IEC 7816 [6]

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	-	-	-	-	-	-	Universal class, not defined in ISO/IEC 7816 [6]
0	1	-	-	-	-	-	-	Application class, identification defined in this document
1	0	-	-	-	-	-	-	Context-specific class, defined in ISO/IEC 7816 [6]
1	1	-	-	-	-	-	-	Private class, not defined in ISO/IEC 7816 [6]
-	-	0	-	-	-	-	-	Primitive encoding
-	-	1	-	-	-	-	-	Constructed encoding
-	-	-	Not all set to 1					Tag number from zero to thirty (short tag field, i.e. a single byte)
-	-	-	1	1	1	1	1	Tag number greater than thirty (long tag field, i.e. two or three bytes)

In tag fields of two or more bytes, the values '00' to '1E' and '80' are invalid for the second byte.

— In 2-byte tag fields, the second byte consists of bit b8 set to 0 and bits b7 to b1 encoding a number greater than thirty. The second byte is valued from '1F' to '7F; the tag number is from 31 to 127.

— In 3-byte tag fields, the second byte consists of bit b8 set to 1 and bits b7 to b1 not all set to 0; the third byte consists of bit b8 set to 0 and bits b7 to b1 with any value. The second byte is valued from '81' to 'FF' and the third byte from '00' to '7F'; the tag number is from 128 to 16 383.

E.2 BER-TLV length fields

In short form, the length field consists of a single byte where bit b8 is set to 0 and bits b7 to b1 encode the number of bytes in the value field. One byte can thus encode any number from zero to 127.

NOTE Any number from one to 127 is encoded in the same way in BER-TLV length field as in L_c and L_e fields. The encoding differs for zero, 128 and more. See for example, the coding of DOs in the GET DATA command in 11.4.3.

In long form, the length field consists of two or more bytes. Bit b8 of the first byte is set to 1 and bits b7 to b1 are not all equal, thus encoding the number of subsequent bytes in the length field. Those subsequent bytes encode the number of bytes in the value field.

ISO/IEC 7816 [6] supports length fields of one, two, … up to five bytes (see Table E.2). In ISO/IEC 7816 [6], the values '80' and '85' to 'FF' are invalid for the first byte of length fields.

Table E.2 — BER-TLV length fields in ISO/IEC 7816 [6]

	1st byte	2nd byte	3rd byte	4th byte	5th byte	N
1 byte	'00' to '7F'	-	-	-	-	0 to 127
2 bytes	'81'	'00' to 'FF'	-	-	-	0 to 255
3 bytes	'82'	'0000' to 'FFFF'		-	-	0 to 65 535
4 bytes	'83'	'000000' to 'FFFFFF'			-	0 to 16 777 215
5 bytes	'84'	'00000000' to 'FFFFFFFF'				0 to 4 294 967 295

E.3 BER-TLV value fields

Bit b6 of the first byte of the tag field indicates an encoding of the value field:

— The value 0 indicates a primitive encoding of the DO, i.e. no information is provided on the coding of the value field, even if the value field is encoded in BER-TLV.

— The value 1 indicates a constructed encoding of the DO, i.e. the value field, if present, is encoded in BER-TLV. A non-empty value field is one DO, or a concatenation of DOs without padding.

Annex F
(informative)

BER-TLV data object handling

F.1 Generations and templates in a constructed DO

Root DO (virtual) 0 Generation DO		1st Generation DOs		2nd Generation DOs		3rd Generation DOs		4th Generation DOs	
'7F70' L	*Value =*								
		T1 L1	Value 1 =						
				T2 L2	V2				
				T3 L3	Value 3 =				
						T4 L4	V4		
						T6 L6	V6		
						T5 L5	Value 5 =		
								T8 L8	V8
								T8 L8'	V8'
								T8 V8"	V8"
				T2 L2'	V2'				
				T14 L14	V14				
				T9 L9	Value 9 =				
						T4 L4'	V4'		
						T6 L6"	V6"		
		T6 L6'	V6'						

Figure F.1 — Generations, DOs and templates in a constructed DO

Each line of Figure F.1 contains one and only one DO. A DO uses two successive cells on the same line, the leftmost cell contains the DO header (Tag, Length), and the rightmost cell contains its value.

The leftmost vertical bold line shows the current template after file selection, or after selection of the virtual DO'7F70'. The template contains DO'T1' and 'T6', which appear immediately on the right of a bold solid line. No Tn Ln tuple appears immediately on the right of a bold dotted line; this dotted line shows that the template is not finished.

The DOs belonging to a line with this dotted border **belong to the structure (file, record or DataString)**, **but do not belong to the** template, because they are nested in other constructed DOs, i.e. do not belong to the 1st generation.

After selection of DO'T1', the current template is shown by a bold line, starting on the immediate right of the cell holding "Value 1". This cell is framed by a bold border to indicate that this value is a template, i.e. a concatenation of the DOs nested within the constructed DO'T1', which make up the current template after selection of DO'T1'. This template features five 2nd generation DOs, DO'T2', 'T3', 'T2', 'T14', 'T9'. The template also uses a dotted bold line. The DOs belonging to a line with this dotted border **belong to DO'T1'**, **but do not belong to the current template after DO'T1' selection**, because they are nested in other constructed DOs, i.e. do not belong to 2nd generation.

There are two instances of DO'T2', with identical tags, but possibly different lengths and values. This is allowed, but implies that handling of those DOs shall use DO ordering within this template. As the two

instances belong to the same template, the consistent use of tagging for referencing ensures that they have the same meaning.

After selection of DO'T3', the current template is shown by a bold line. This template features three 3rd generation DOs, DO'T4', 'T6', 'T5'.

After selection of DO'T9', the current template features two 3rd generation DOs, DO'T4', 'T6'. Though in the same generation as the current template after selection of DO'T3', it is a different template. The 3rd generation shows two instances of DO'T6', but they belong to different templates and as such may have different meanings, except if further information may be derived from e.g. the class of tag 'T6'.

F.2 Referencing by an extended header

All examples shown in this sub clause would result in the same byte strings with the two leftmost columns deleted from all figures. The title of this sub clause would then be "Referencing by an extended header list". This emphasizes that the target DO and the first header of an extended header must match, i.e. have the same tag, in order to reference a non-empty byte string.

'A1 37'	Value 1 =						
		'82 03'	'21 22 23'				
		'A3 1C'	Value 3 =				
				'84 04'	'40 41 42 43'		
				'86 02'	'61 62'		
				'A5 10'	Value 5 =		
						'88 03'	'81 82 83'
						'88 03'	'84 85 86'
						'88 04'	'87 88 89 8A'
		'82 04'	'24 25 26 27'				
		'8E 03'	'E0 E1 E2'				
		'A3 07'	Value 3' =				
				'84 01'	'46'		
				'86 02'	'63 64'		

Figure F.2 — Target DO of all extended headers in this sub clause

'A1 0C'	Value 1 =						
		'A3 06'	Value 3 =				
				'A5 04'	Value 5 =		
						'88 00'	
						'88 02'	
		'8E 00'					
		'A3 80'					

Figure F.3 — Extended header data element 1, including no skipped match on any primitive DO.

'A1 1B'	Value 1 =						
		'A3 0B'	Value 3 =				
				'A5 09'	Value 5 =		
						'88 03'	**'81 82 83'**
						'88 02'	**'84 85'**
		'8E 03'	**'E0 E1 E2'**				
		'A3 07'	Value 3' =				
				'84 01'	**'46'**		
				'86 02'	**'63 64'**		

Figure F.4 — Matches found with extended header 1 in a sub-tree of DO'A1'

Figure F.4 shows that only two first instances of DO'88' were matched. If the extended header is under tag '4D', one cannot include the second instance without including the first. Truncation of the values of primitive DOs have been performed. The referenced byte string is:

— Either in Data Object format:
{88 03 81 82 83} {88 02 84 85} {8E 03 E0 E1 E2} {A3 07 {84 01 46} {86 02 63 64}}.

— Or in data element format: '81 82 83 84 85 E0 E1 E2' {84 01 46} {86 02 63 64}

Figure F.5 shows how to include only the second instance of DO'A3':

'A1 04'	Value 1 =						
		'8E 00'					
		'A3 80'					

Figure F.5 — Extended header data element 2, implicitly excluding a match on a constructed DO

'A1 0E'	Value 1 =						
		'8E 03'	**'E0 E1 E2'**				
		'A3 07'	Value 3 =				
				'84 01'	**'46'**		
				'86 02'	**'63 64'**		

Figure F.6 — Matches found with extended header 2 in a sub-tree of DO'A1'

The first instance of DO'A3' is not referenced, because the first DO to be referenced was DO'8E', which excludes referencing any DO before it, according to 8.4.7. Figure F.7 shows an extended header which will have the same outcome as the extended header of Figure F.5.

'A1 06'	Value 1 =						
		'A3 00'					
		'8E 00'					
		'A3 80'					

Figure F.7 — Redundant coding of extended header data element 2

In the case shown by Figure F.5, Figure F.6 and Figure F.7, the referenced byte string is:

— Either in Data Object format: {8E 03 E0 E1 E2} {A3 07 {84 01 46} {86 02 63 64}}.

— Or in data element format: 'E0 E1 E2' {84 01 46} {86 02 63 64}

'A1 04'	Value 1 =						
		'A3 00'					
		'A3 80'					

Figure F.8 — Extended header data element 3, including a skipped match on a constructed DO

'A1 09'	Value 1 =						
		'A3 07'	Value 3' =				
				'84 01'	**'46'**		
				'86 02'	**'63 64'**		

Figure F.9 — Matches found with extended header 3 in a sub-tree of DO'A1'

In the case shown by Figure F.8 and Figure F.9, the indication that a match with the first instance of DO 'A3' must be skipped becomes necessary, because no match may occur between matches with the two instances of DO 'A3'. The referenced byte string is:

— Either in Data Object format: {A3 07 {84 01 46} {86 02 63 64}}.

— Or in data element format: {84 01 46} {86 02 63 64}

'A1 08'	Value 1 =						
		'A3 06'	Value 3 =				
				'A5 04'	Value 5 =		
						'88 00'	
						'88 80'	

Figure F.10 — Extended header data element 4, including a skipped match on a primitive DO

'A1 09'	Value 1 =						
		'A3 07'	Value 3 =				
				'A5 05'	Value 5 =		
						'88 03'	'84 85 86'

Figure F.11 — Matches found with extended header 4 in a sub-tree of DO'A1'.

Figure F.8 shows that only the second instance of DO'88' was matched. The syntax of extended header 4 is not possible under tag '4D'. The referenced byte string is :

— Either in Data Object format: {88 03 84 85 86}. The extended header is tagged by '5F 61'

— Or in data element format: '84 85 86'. The extended header is tagged by '5F 60'

F.3 Use of the UPDATE DATA command

This example makes the assumption that one and only one DO 'B1' is present in the VA. If more than one instance of DO'B1' are present, updating one of them can be performed:

— Either by selection of an instance by a SELECT DATA C-RP, then update the DOs in the next generation.

— Or setting a pointer on this instance by one or several GET NEXT DATA C-RPs.

Figure F.12 shows a DO'B1' to be updated by an UPDATE DATA C-RP. Its tree structure is comprised of:

A primitive DO'82' (2nd generation) and a constructed DO'B2' (2nd generation).

The constructed DO'B2' (2nd generation) nests two primitive DOs '90' and '91', (3rd generation) and one constructed DO'B3' (3rd generation).

The constructed DO'B3' (3rd generation) nests two primitive DOs '84' and '86' (4th generation).

'B1 1C'	Value 1 =						
		'82 04'	'21 22 23 24'				
		'B2 14'	Value 2 =				
				'90 03'	'01 02 03'		
				'91 04'	'11 12 13 14'		
				'B3 07'	Value 3 =		
						'84 01'	'46'
						'86 02'	'63 64'

Figure F.12 — DO'B1' to be updated

Figure F.13 shows the argument of an UPDATE DATA command. It is a DO based on DO'B1'. It is a sub-tree where, with respect to the original DO 'B1':

— some "branches" (DOs) may be present; those will be updated.

— some may be missing; those will not be updated, hence kept as in the original DO 'B0'.

— some may be added, they will be created.

'B1 10'	Value 1' =						
		'B2 09''	Value 2' =				
				'B3 07'	Value 3' =		
						'84 00'	
						'86 04'	'65 66 67 68'
		'8E 03'	'E1 E2 E3'				

Figure F.13 — Argument of the UPDATE DATA command

Figure F.14 shows the content of DO'B1' after the success of the UPDATE DATA C-RP:

In the 2nd generation, DO'82' is untouched, DO'B2' updated and DO'8E' created.

In the 3rd generation, primitive DOs '90' and '91' are untouched and DO'B3' is updated.

In the 4th generation, primitive DO'84' is now empty and DO'86' is updated. The new value of DO'86' transmitted in the command (see Figure F.13) replaces the previous value (see Figure F.12)

'B1 22'	**Value 1" =**						
		'82 04'	'21 22 23 24'				
		'B2 15'	**Value 2" =**				
				'90 03'	'01 02 03'		
				'91 04'	'11 12 13 14'		
				'B3 08'	**Value 3" =**		
						'84 00'	
						'86 04'	**'65 66 67 68'**
		'8E 03'	**'E1 E2 E3'**				

Figure F.14 — Updated DO'B1', where modified or added DOs show in bold type.

F.4 Security attribute for one DO

'UV' L	Value					// constructed DO				
		'XY' L1	V1			// DO under DO'UV'				
		'62' L2	Value 2			// data control parameters DO				
				'A0 L3'	Value 3	// Security attribute for DOs				
						'9C' L4	V4	//Security attribute referencing security parameters template #1		
						'5C01'	'XY'	// Tag list referencing DO'XY'		
				'AD' L5	Value 5			// Security parameters template		
						'8001'	'01'	// Security parameters template number		
						'A1' L6	Value 6	// Protection uses of a private key		
						'63' L7	Value 7	//Wrapper referencing the key		

Figure F.15 — Current template with DO'XY' and its security attribute (general layout)

Figure F.15 illustrates the coding of the security attributes of a DO'XY' nested in a DO'UV'. The security attribute DO'A0' is nested in the control parameters DO'62', which also nests a Security parameters template DO'AD'. DO'62' may contain other CP DOs not represented in the figure, because irrelevant in this example.

The tag given in the tag list nested in the security attribute DO'A0' must be significant, i.e point to a DO under DO'UV', in other words a 1st generation DO if DO'UV' is selected. The DO'AD' referenced by its number must be under DO'62' under DO'UV'.

F.5 Example of key referencing in a self-controlled DO

1st	2nd	3rd	4th	< Generation	
62		Control parameter template			
	A0	Security attribute template for DOs			
		90	AMF || SCB (ref#1) || SCB (ref#2) || SCB (ref#3)		Tag '9D' can also be used here under tag 'AB' when using expanded format
		5C	tag list, contents not detailed in this table		
	73	proprietary information encoded in BER-TLV			
	AC	Cryptographic mechanisms identifier template			
	8A	LCS (Life cycle status)			
	AD	Security parameter template #1 referenced above			
		06	OID pointing to the description of e.g. algorithm and use of proprietary information		
		80	value = '01', sequence number of DO'AD'		
		A0	*Security attribute extension for Authentication Object*		
			8C	Security attributes in compact format, SE referencing	Choice between those
			AB	Security attributes in expanded format, SPT referencing	
			9C	Security attributes compact format, SPT referencing	
			5C	Tag list, contents not detailed in this table	
		AD	Security parameter template, contents not detailed in this table		
		…	Further DOs not detailed in this table, 3rd generation		
		B3	OID-related information encoded in BER-TLV		
		63	Wrapper, contents not detailed in this table		
	AD	Security parameter template #2 referenced above			
		80	value = '02', sequence number of DO'AD'		
		A4	*Security attribute extension for Secret Key*		
			'IJ'	Security attributes referencing SEs in compact ('IJ'=8C') or expanded ('IJ'= 'AB') format	
		63	Wrapper, contents not detailed in this table		
	AD	Security parameter template #3 referenced above			
		06	OID pointing to the description of e.g. algorithm and use of proprietary info		
		80	value = '03', sequence number of DO'AD'		
		A1	*Security attribute extension for Private Key*		
			'ZT'	Security attributes in compact ('ZT'=9C') or expanded ('ZT'= 'AB') format	
		63	Wrapper, contents not detailed in this table		
		73	OID-related information encoded in BER-TLV		

Figure F.16 — Value of a self-controlled DO 'XY' with security attributes referencing three keys

NOTE Figure F.16 does not indicate the length field of DOs, which would bring small information. Bold lines indicate on the left the tag of a constructed DO, on the right the tag(s) of the DO(s) nested within.

Figure F.16 illustrates the coding of the security attributes in a DO'XY' The security attribute DO'A0' is nested in the control parameters DO'62', which also nests three instances of the security parameters template DO'AD'. DO'XY' and DO'62' may contain other CP DOs not represented in the figure, because irrelevant in this example.

The tag(s) given in a non empty tag list nested in the security attribute DO'A0' must be significant, i.e. point to a DO under DO'XY', in other words a 1st generation DO. The security parameter template DO'AD' belongs to the same generation as the security attribute DO'A0' which references the DO'AD'.

NOTE A security attribute DO'A0' is nested in a DO'62'. A security attribute extension DO'A0' is nested in a DO'AD'. Their syntaxes are different.

Annex G
(informative)

Template extension by tagged wrapper

G.1 General

Template extension by tagged wrappers allows a card to emulate, within a given VA, one or several "distant" DOs which do not actually belong to the VA (see 8.2.2 and 8.4.8 and 8.4.9). This annex assumes that the security status allows reading a distant DO by a GET DATA C-RP. The figure below shows both the tagged wrapper created in a base template (e.g. by a PUT DATA command) and the outcome of this tagged wrapper in the template extension. Only tags appear:

<table>
<tr><th>n^{th} generation</th><th>$(n+1)^{th}$ generation</th><th colspan="2">Meaning</th></tr>
<tr><td>'63'</td><td></td><td colspan="2">Tagged wrapper present in the base template</td></tr>
<tr><td></td><td>'XY' (empty)</td><td colspan="2">The DO will be referenced locally by tag 'XY'.</td></tr>
<tr><td></td><td>'5C' or '4D'</td><td colspan="2">Indirect reference to DO'ZT' in a transient VA, see 1) below.</td></tr>
<tr><td rowspan="2"></td><td>'4F'</td><td>References the curDF in the transient VA</td><td rowspan="2">The automatic resolution of the tagged wrapper uses these DOs to set transient VA where the reference to DO'ZT' is valid. See 2) and 3) below.</td></tr>
<tr><td>'51'</td><td>References the curEF in the transient VA</td></tr>
<tr><td>'XY'</td><td></td><td colspan="2">Emulation of DO'ZT' in the template extension</td></tr>
</table>

Figure G.1 — Syntax and outcome of a tagged wrapper

1) Other tags are available to reference the DO (see 8.4.8).
2) There are other means to set the transient VA (see 8.4.8).
3) When elements of the transient VA belong to the current VA, there is no need to repeat them in the tagged wrapper, only to confirm them when they are both absent (see following clauses).

G.2 Referencing within the current EF

The example below shows the content of an EF '12 34' (file identifier) under an application DF 'A0 01 02 03 04' (application identifier). This figure shows the hexadecimal coding of tag and length fields. Except in the value of the tagged wrapper, the values of the primitive DOs are not significant.

1st generation		2nd generation		3rd generation		4th generation		5th generation
'62 0E'	Value							
		'A0 0C'	Value					
				'5C 00'				
				'8B 08'	'B1 B2 B3 B4 B5 B6 B7 B8'			
'80 01'	**'XY'**							
'A2 13'	Value							
		'B2 11'	Value					
				'80 04'	'11 12 13 14'			
				'81 02'	'21 22'			
				'82 03'	'31 32 33'			
				'63 0B'	Value			
						'91 00'		
						'4D05'	Value	
								'7F 70 02 **80** 00'
						'51 00'		
				'91 01'	***'XY'***			

NOTE The bold italic font used for ***'9101XY'*** indicates that it is part of the template extension coded in the DO'63' above (tagged wrapper).

Figure G.2 — Contents of EF '12 34' under the application DF named 'A0 01 02 03 04'

The value of DO'B2' under DO'A2' consists of a base template (DOs '80', '81', '82', and '63') and a template extension (DO'91'), because the tagged wrapper DO'63' references by an extended header the DO'80' in the first generation (leftmost column). This DO'80' will be referenced by tag '91' in the template it extends, here to avoid a collision of tags '80' possibly referencing different types. Tag '80' appears twice in bold type, in the 1st generation (the DO belongs to the base template); and in the 5thgeneration (in an extended header data element). The value 'XY' appears twice in bold type, in the 1st generation (where it is defined); and in the 3rd generation, as the value of DO'91'. Tag '91' appears twice in bold type, in the 4th generation where it is defined); and in the 3rd generation (the DO'91' belongs to the template extension).

The empty file reference DO'51' states that the indirection is valid in the VA referencing DO'B2', which nests the DO'63', thus confirms that the transient VA references the application 'A0 01 02 03 04', and the EF '12 34'. When the current constructed DO is DO'B2', DO'80' may be addressed locally by tag '91':

— the command APDU '00 CA 00 91 00' will return 'XY 9000'.

— the command APDU '00 CB 00 00 03' {5C 01 91} '00' will return {91 01 XY} '9000'.

G.3 Referencing within the current application DF, first example

1st generation		2nd generation		3rd generation		4th generation
'62 0E'	Value					
		'A0 0C'	Value			
				'5C 00'		
				'8B 08'	'B1 B2 B3 B4 B5 B6 B7 B8'	
'80 01'	**'XY'**					
'A2 13'	Value					
		'B2 11'	Value			
				'80 04'	'11 12 13 14'	
				'81 02'	'21 22'	
				'82 03'	'31 32 33'	

Figure G.3 — Alternative contents of EF '12 34' under the application DF named 'A0 01 02 03 04'

In Figure G.3, the contents of EF '12 34' does not include the tagged wrapper, which may be present (if valid) in any VA referencing the application DF 'A0 01 02 03 04', but not the EF '12 34': The syntax of a tagged wrapper, referencing the same DO as in G.2, will be:

'6309'			
	'92 00'		In this example, the DO will be referenced locally by tag '92'
	'5C 01'	'80'	Indirect reference to DO'80' in the first generation DOs in EF '1234'
	'51 02'	'1234'	References EF '1234' under the current application

Figure G.4 — Syntax of a tagged wrapper referencing an EF; it uses a tag list

As compared to Figure G.2:

— An explicit referencing of the EF states that the indirect reference is valid in EF '12 34'.

— As DO'80' belongs to the 1st generation in EF '1234' the extended header may be replaced by a tag list.

When issued from any VA referencing the application named 'A0 01 02 03 04' and the parent of the tagged wrapper above, DO'80' may be addressed locally by tag '92':

— the command APDU '00 CA 00 92 00' will return 'XY 9000'.

— the command APDU '00 CB 00 00 03' {5C 01 92} '00' will return {92 01 XY} '9000'.

An analogous (but not identical) function may be supported by another GET DATA C-RP. When issued from the same VA, the command APDU '00 CB **12 34** 03' {5C 01 **80**} '00' will return {**80** 01 XY} '9000'. The differences with the C-RP above are shown in bold type. In order to use this alternate solution, the outside world must know the file identifier '12 34', the tag '80' of the DO and its generation. All this can depend on the implementation.

G.4 Referencing in the current application DF, second example

One wants to address DO'A0' under DO'62'. An extended header is necessary; this DO'A0' belongs to the 2nd generation in the EF. The tagged wrapper will be:

'630C'			
	'A000'		In this example, the DO will be referenced locally by tag 'A0'
	'4D04'	'6202A080'	Indirect reference to DO'A0' in the 2nd generation DOs under DO'62'
	'5102'	'1234'	References EF '1234' under the current application

Figure G.5 — Syntax of a tagged wrapper referencing an EF; it uses an extended header

The tag of DO'A0' under DO'62' has the standard meaning of security attribute for DOs. Addressing it locally with the same tag, under another DO'62', avoids duplication of this security attribute, e.g. if several EFs within an application DF use the same default security attribute for DOs.

When issued from any VA referencing the application named 'A0 01 02 03 04' and the tagged wrapper above,

— the command APDU '00 CA 00 A0 00' will return {5C 00} {8B 08 B1 B2 B3 B4 B5 B6 B7 B8} '9000'.

— the command APDU '00 CB 00 00 03' {'5C 01 A0'} '00'
will return {A0 0C {5C 00} {8B 08 B1 B2 B3 B4 B5 B6 B7 B8}} '9000'.

G.5 Referencing out of the current application DF

The current VA does not reference the application DF 'A0 01 02 03 04'. Therefore, referencing the application and the EF is necessary. For referencing the same DO as in the first example of G.3, the tagged wrapper will be:

'6310'			
	'9200'		In this example, the DO will be referenced locally by tag '92'
	'5C01'	'80'	Indirect reference to DO'80' in the first generation DOs
	'4F05'	'A0 01 02 03 04'	References the application named 'A0 01 02 03 04'
	'5102'	'12 34'	References EF '12 34' under the referenced application

Figure G.6 — Syntax of a tagged wrapper referencing an application DF and an EF; it uses a tag list

When issued from any VA including the tagged wrapper above,

— the command APDU '00 CA 00 92 00' will return 'XY 9000'.

— the command APDU '00 CB 00 00 03' {5C 01 92} 00' will return {92 01 XY} '9000'.

G.6 Warnings

Indirect referencing of a DO belonging to a template extension may result in circular referencing.

Using tagged wrappers trades complexity of commands against complexity in data. Thus, care must be taken when choosing the tag by which a distant DO may be seen locally, in order to avoid addressing:

— a constructed DO by a tag indicating a primitive one, which is awkward, but possible.

— a primitive DO by a tag indicating a constructed one; this will result in a DO or a template of invalid syntax.

— a standard DO, of a defined type, by another tag which does not indicate this type. This implies that the specification or standard using this mentions the type.

— a DO by a tag which is already used in the same template as the tagged wrapper, without being aware of the duplication of instances.

Annex H
(informative)

Parsing an extended header against its target DO

The procedure described in this annex accounts for three use cases of an extended header:

— Either complete extended headers built according to 8.4.5, knowing the referenced constructed DO.

— Or built according to 8.4.5, knowing a previous value of the referenced DO, updated since. This may result in different structures for the extended header and its target DO.

— Or built without redundant references to DOs to be skipped, in order to shorten the extended header.

To follow the procedure, display the extended header and the target DO as in Annex F: one header per line in the header table, one DO per line in the DO table. The order of lines shows the order of headers and DOs, the generations show as column shifts. The following rules shall apply sequentially, unless "Go to…" is mentioned:

1) If at least one table is empty, the procedure is completed.

2) If the extended header is meant to reference only one DO, the procedure is completed if a non-skipped match (L≠'80' in the header) with a primitive DO, or a complete match (L='80' in the header) with a constructed DO is achieved.

3) Read the first (topmost) line in the header table and search a match of the header in the DO table. Unless stated otherwise (see 8)), the search shall be done in the first (topmost) line of the DO table.

4) If a match (same tag, same generation) is achieved, go to 10).

5) If the DO generation is lower than the header generation, go to 7).

6) If searching has not reached the last line of the DO table, search continues in the next line. Go to 1).

NOTE When searching a template, DOs in higher generations or with the wrong tag are ignored.

7) If the searched match was on a primitive DO, delete the line from the header table. Go to 1).

NOTE The referenced DO was never in the searched template, or got deleted after a match (see 10)).

8) If the searched match was on a constructed DO, either skipped (L='00' in the header) or complete (L='80' in the header), delete the line from the header table. Go to 1).

NOTE The referenced DO was never in the searched template, or got deleted after a match (see 10)).

9) Delete from the header table the lines referencing the constructed DO and its contents. Go to 1).

NOTE The referenced constructed DO was never in the searched template, or got deleted after a match (see 10)). The references to its contents become useless.

10) Apply 8.4.6. Delete the matching line from both tables. In the DO table, delete all lines above, if any.

NOTE Any contribution of those deleted DOs to the byte string would not match the order mandated by the extended header.

11) If a match with a primitive DO was achieved, go to 1).

12) If a skipped match (L='00' in the header) or a complete match (L='80' in the header) with a constructed DO was achieved, delete from the DO table all lines of the value of the matching DO. Go to 1).

13) If another match with a constructed DO was achieved, go to 1).

NOTE Parsing and searching will then take place in the next generation.

Bibliography

[1] EN 14890-1:2008, *Application Interface for smart cards used as Secure Signature Creation Devices — Part 1: Basic services*

[2] EN 14890-2:2012, *Application Interface for smart cards used as Secure Signature Creation Devices — Part 1: Additional Services*

[3] ISO 3166-1:1997, *Codes for the representation of names of countries and their subdivisions — Part 1: Country codes*

[4] ISO/IEC 7810:2003, *Identification cards — Physical characteristics*

[5] ISO/IEC 7812-1:2000, *Identification cards — Identification of issuers — Part 1: Numbering system*

[6] ISO/IEC 7816 (all parts), *Identification cards — Integrated circuit cards*

[7] ISO/IEC TR 9577:1999, *Information technology — Protocol identification in the network layer*

[8] ISO/IEC 9796 (all parts), *Information technology — Security techniques — Digital signature schemes giving message recovery*

[9] ISO/IEC 9797 (all parts), *Information technology — Security techniques — Message Authentication Codes (MACs)*

[10] ISO/IEC 9798 (all parts), *Information technology — Security techniques — Entity authentication*

[11] ISO/IEC 9979:1999, *Information technology — Security techniques — Procedures for the registration of cryptographic algorithms*

[12] ISO 9992-2:1998, *Financial transaction cards — Messages between the integrated circuit card and the card accepting device — Part 2: Functions, messages (commands and responses), data elements and structures*

[13] ISO/IEC 10116:1997, *Information technology — Security techniques — Modes of operation for an n-bit block cipher*

[14] ISO/IEC 10118 (all parts), *Information technology — Security techniques — Hash-functions*

[15] ISO/IEC 10536 (all parts), *Identification cards — Contactless integrated circuit cards — Close-coupled cards*

[16] ISO/IEC 10646, *Information technology — Universal Coded Character Set (UCS)*

[17] ISO/IEC 11770 (all parts), *Information technology — Security techniques — Key management*

[18] ISO/IEC 14443 (all parts), *Identification cards — Contactless integrated circuit cards — Proximity cards*

[19] ISO/IEC 14888 (all parts), *Information technology — Security techniques — Digital signatures with appendix*

[20] ISO/IEC 15693 (all parts), *Identification cards — Contactless integrated circuit cards — Vicinity cards*

[21] ISO/IEC 18033 (all parts), *Information technology — Security techniques — Encryption algorithms*

[22] ISO/IEC 18092, *Information technology — Telecommunications and information exchange between systems — Near Field Communication — Interface and Protocol (NFCIP-1)*

[23] ISO/IEC 24727 (all parts), *Identification cards — Integrated circuit card programming interfaces*

[24] ISO/IEC 24727-2, *Identification cards — Integrated circuit card programming interfaces — Part 2: Generic card interface*

[25] ISO/IEC 24727-3, *Identification cards — Integrated circuit card programming interfaces — Part 3: Application interface*

[26] IETF RFC 1738:1994, *Uniform resource locators (URL)*

[27] IETF RFC 2396:1998, *Uniform resource locators (URL): General syntax*

ICS 35.240.15

Price based on 150 pages