**ISO/IEC JTC 1/SC 17**

Date: 2002-01-04

**ISO/IEC CD 7816-4**

ISO/IEC JTC 1/SC 17/WG 4

Secretariat: AFNOR

# Information technology — Identification cards — Integrated circuit(s) cards with contacts — Part 4: Interindustry commands for interchange

*Technologies de l'information — Cartes d'identification — Cartes à circuit(s) intégré(s) à contacts — Partie 4 : Commandes intersectorielles pour les échanges*

**Warning**

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: International Standard
Document subtype:
Document stage: (20) Preparatory
Document language: E

C:\Program Files\Adobe\Acrobat 4.0\Acrobat\plug_ins\OpenAll\Transform\temp\Kyoto4R.doc STD Version 2.0

# Contents

Page

**Foreword** ..... iv

**Introduction** ..... vi

**1 Scope** ..... 1

**2 Normative references** ..... 1

**3 Terms and definitions** ..... 2

**4 Abbreviations and notation** ..... 5

**5 Basic organizations** ..... 6
**5.1 Application identifiers** ..... 7
**5.2 Files for applications and data** ..... 8
**5.3 Command-response pairs** ..... 11
**5.4 Data objects** ..... 17
**5.5 Historical bytes** ..... 21
**5.6 File control information** ..... 24
**5.7 Security architecture** ..... 27

**6 Secure messaging** ..... 28
**6.1 SM format concept** ..... 29
**6.2 Basic SM data objects** ..... 29
**6.3 Auxiliary SM data objects** ..... 32
**6.4 Security supports** ..... 35
**6.5 Security environments** ..... 36

**7 Security attributes** ..... 37
**7.1 Compact format** ..... 38
**7.2 Expanded format** ..... 39
**7.3 Access rule references** ..... 40

**8 Interindustry commands for interchange** ..... 41
**8.1 Selection** ..... 41
**8.2 Data unit handling** ..... 44
**8.3 Record handling** ..... 47
**8.4 Data object handling** ..... 52
**8.5 Basic security handling** ..... 54
**8.6 Transmission handling** ..... 61

**9 Application-independent card services** ..... 63
**9.1 Card-originated command-response pairs** ..... 63
**9.2 Card identification** ..... 64
**9.3 Application identification and selection** ..... 65
**9.4 Data element retrieval** ..... 67

**Annex A (informative) Application identifiers using issuer identification numbers** ..... 72

**Annex B (informative) Object identifiers and tag allocation schemes** ..... 73

**Annex C (informative) Secure messaging** ..... 75

**Annex D (informative) Chains of GENERAL AUTHENTICATE commands** ..... 80

# 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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

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

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 7816 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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

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

— *Part 1: Physical characteristics*

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

— *Part 3: Electronic signals and transmission protocols*

— *Part 4: Interindustry commands for interchange*

— *Part 5: Registration procedures for application providers*

— *Part 6: Interindustry data elements for interchange*

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

— *Part 8: Interindustry commands for a cryptographic toolbox*

— *Part 9: Interindustry commands for card and file management*

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

— *Part 11: Personal verification through biometric methods*

— *Part 12: USB electrical interface and operating procedures*

— *Part 13: Registration of integrated circuit manufacturers*

— *Part 15: Cryptographic information application*

Annexes A, B, C and D are for information only.

NOTE 1 The clause *APDU message structures* and the two annexes *Transportation of APDU messages*, respectively by T = 0 and by T = 1, have been extracted from ISO/IEC 7816-4:1995, i.e., the first edition of part 4, for creating an amendment in three clauses to ISO/IEC 7816-3:1997, i.e., the second edition of part 3.

NOTE 2 After a deep reorganization and the inclusion of relevant information extracted from ISO/IEC 7816-5:1994 and from ISO/IEC 7816-6:1996, the second editions of ISO/IEC 7816-4, i.e., *Interindustry commands for interchange*, ISO/IEC 7816-8, i.e., *Interindustry commands for a cryptographic toolbox*, and ISO/IEC 7816-9, i.e., *Interindustry commands for card and file management*, together cancel and replace

— ISO/IEC 7816-4:1995, *Interindustry commands for interchange*

— ISO/IEC 7816-4:1995: Amd.1:1997, *Impact of secure messaging on the structure of APDU messages*

— ISO/IEC 7816-8:1999, *Security-related interindustry commands*

— ISO/IEC 7816-9:2000, *Additional interindustry commands and security attributes*

# Introduction

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

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

# Information technology — Identification cards — Integrated circuit(s) cards with contacts — Part 4: Interindustry commands for interchange

## 1 Scope

This part of ISO/IEC 7816 specifies

— means and mechanisms for identifying and addressing applications in the card,

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

— access methods to files and data in the card,

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

— contents of command-response pairs exchanged between the interface device and the card,

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

— structures and contents of the historical bytes sent by the card during the answer to reset,

— methods for secure messaging,

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

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

## 2 Normative references

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this part of ISO/IEC 7816. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 7816 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO 639:1998, *Code for the representation of names of languages*

ISO/IEC 646:1991, *Information technology — ISO 7-bit coded character set for information interchange*

ISO 3166-1:1997, *Codes for the representation of names of countries and their subdivisions — Part 1: Country codes*

ISO 4217:1995, *Codes for the representation of currencies and funds*

ISO 4909:2000, *Bank cards — Magnetic stripe data content for track 3*

ISO/IEC 7501-1:1997, *Identification cards — Machine-readable travel documents — Part 1: Machine-readable passport*

ISO/IEC 7812-1:2000, *Identification cards — Identification of issuers — Part 1: Numbering system*

ISO/IEC 7813:1995, *Identification cards — Financial transaction cards*

ISO/IEC 7816 (all parts), *Information technology — Identification cards — Integrated circuit(s) cards with contacts*

ISO 8583:1993, *Financial transaction card-originated messages — Interchange message specifications*

ISO/IEC 8825-1:1998, *Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

ISO/IEC 8859-1:1998, *Information technology — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1*

ISO/IEC TR 9577:1996, *Information technology — Protocol identification in the network layer*

ISO/IEC 9796 (all parts), *Information technology — Security techniques — Digital signature schemes giving message recovery*

ISO/IEC 9797 (all parts), *Information technology — Security techniques — Message authentication codes (MACs)*

ISO/IEC 9798 (all parts), *Information technology — Security techniques — Entity authentication*

ISO/IEC 9979:1999, *Information technology — Security techniques — Procedures for the registration of cryptographic algorithms*

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*

ISO/IEC 10116:1997, *Information technology — Security techniques — Modes of operation for an n-bit block cipher*

ISO/IEC 10118 (all parts), *Information technology — Security techniques — Hash-function*

ISO/IEC 10918-1:1994, *Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines*

ISO/IEC 11170 (all parts), *Information technology — Security techniques — Key management*

ISO/IEC 11544:1993, *Information technology — Coded representation of picture and audio information — Progressive bi-level image compression*

ISO/IEC 14888 (all parts), *Information technology — Security techniques — Digital signatures with appendix*

ISO/IEC 18033 (all parts), *Information technology — Security techniques — Encryption algorithms*

IETF RFC 1738:1994, *Uniform resource locators (URL)*

IETF RFC 2396:1998, *Uniform resource locators (URL): General syntax*

## 3 Terms and definitions

For the purposes of this part of ISO/IEC 7816, 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 working elementary file indicating operating characteristics of the card

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

**3.4**
**application identifier**
data element identifying an application in the card

**3.5**
**application label**
data element for use at the man-machine interface

**3.6**
**application provider**
entity providing the components required for performing an application in the card

**3.7**
**application template**
set of application-relevant data objects including one application identifier data object

**3.8**
**card-verifiable certificate**
template that can be interpreted and verified by the card by a VERIFY CERTIFICATE operation using a public key

**3.9**
**certification authority**
trusted third party who establishes a proof binding a public key and other relevant information to its owner

**3.10**
**command-response pair**
set of two messages at the interface: a command APDU followed by a response APDU in the opposite direction

**3.11**
**data element**
item of information seen at the interface for which are defined a name, a description of logical content, a format and a coding (it may be composite)

**3.12**
**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 (it may be constructed)

**3.13**
**data unit**
the smallest set of bits that can be unambiguously referenced within an elementary file of transparent structure

**3.14**
**dedicated file**
file containing file control information and, optionally, memory available for allocation

**3.15**
**DF name**
string of bytes uniquely identifying a dedicated file in the card

**3.16**
**directory file**
optional working elementary file containing a list of applications supported by the card and optional related data elements

**3.17**
**elementary file**
set of data units or records or data objects sharing the same file identifier

**3.18**
**file control parameters**
logical, structural and security attributes of a file

**3.19**
**file identifier**
two bytes used to address a file

**3.20**
**file management data**
any information about a file except the file control parameters, e.g., application label, expiration date, etc.

**3.21**
**header list**
concatenation of pairs of tag fields and length fields without delimitation

**3.22**
**internal elementary file**
elementary file for storing data interpreted by the card

**3.23**
**master file**
mandatory and unique dedicated file representing the root of the file structure

**3.24**
**parent file**
dedicated file immediately preceding a given file within the hierarchy

**3.25**
**password**
data that may be required by the application to be presented to the card by its user

**3.26**
**path**
concatenation of file identifiers without delimitation (an absolute path starts with the identifier of the master file; a relative path starts with the identifier of a dedicated file which is not the master file)

**3.27**
**provider**
authority who has or who obtained the right to create a dedicated file in the card

**3.28**
**public-key certificate**
signature that binds a particular person or object and its associated public key (it is issued by a certification authority acting also as tag allocation authority with respect to the data items present in the certificate)

**3.29**
**record**
string or bytes handled as a whole by the card and referenced by a record number or by a record identifier

**3.30**
**record identifier**
number associated with a record and used to reference that record (several records may have the same identifier within an elementary file)

**3.31**
**record number**
sequential number assigned to each record (it uniquely identifies the record within its elementary file)

**3.32**
**secure messaging**
set of means for cryptographic protection of [parts of] command-response pairs

**3.33**
**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

**3.34**
**security environment**
set of components required by an application in the card for secure messaging or for security operations

**3.35**
**tag list**
concatenation of tag fields without delimitation

**3.36**
**template**
set of BER-TLV data objects forming the value field of a constructed BER-TLV data object

**3.37**
**working elementary file**
elementary file for storing data not interpreted by the card

## 4 Abbreviations and notation

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

| | |
|---|---|
| AID | application identifier |
| 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 |
| CR | control reference |
| CRT | control reference template |
| CT | control reference template for confidentiality |
| DF | dedicated file |

| | |
|---|---|
| DIR | directory |
| DST | control reference template for digital signature |
| EF | elementary file |
| EF.ARR | access rule reference file |
| EF.ATR | Answer-to-Reset 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 |
| LCS | life cycle status |
| MF | master file |
| PIX | proprietary application identifier extension |
| P1-P2 | parameter bytes, concatenation of bytes P1 and P2 |
| (P1-P2) | value of the concatenation of bytes P1 and P2 (the first byte is the most significant byte) |
| RFU | reserved for future use |
| RID | registered application provider identifier |
| SC | security condition |
| SE | security environment |
| SEID | security environment identifier |
| SM | secure messaging |
| SW1-SW2 | status bytes, concatenation of bytes SW1 and SW2 |
| TLV | tag, length, value |
| {T-L-V} | data object (inserted for clarity, curly brackets are not significant) |
| 'XY' | notation using the hexadecimal digits '0' to '9' and 'A' to 'F', equal to XY to the base 16 |

## 5 Basic organizations

The subsequent five clauses specify the following basic organizations.

1) Application identifiers
2) Files for applications and data
3) Command-response pairs
4) Data objects
5) Historical bytes
6) File control information
7) Security architecture

## 5.1 Application identifiers

An application identifier (AID) may reference any application. This interindustry data element consists of one to sixteen bytes. Bits 8 to 5 of the first byte of every AID fix a category according to Table 1.

**Table 1 — Categories of application identifiers**

| Value | Meaning |
|---|---|
| '0' to '9' | Reserved for backward compatibility with ISO/IEC 7812-1 (see annex A) |
| 'A' | International category, registration of application providers according to ISO/IEC 7816-5 |
| 'B', 'C' | Reserved for future use by ISO/IEC JTC 1/SC 17 |
| 'D' | National category, registration of application providers according to ISO 3166-1 and ISO/IEC 7816-5 |
| 'E' | Standard category, identification of a standard by an object identifier according to ISO/IEC 8825-1 |
| 'F' | Proprietary category, no registration of application providers |

Figure 1 shows an AID of international category: it consists of a registered application provider identifier (RID) on five bytes and optionally, a proprietary application identifier extension (PIX) on up to eleven bytes.

— RID is a string of ten consecutive quartets.
  - The first quartet shall be set to 'A', i.e., bits 8 to 5 of the first byte shall be set to 1010.
  - Each subsequent quartet shall be in the range '0' to '9'. The nine digits together shall identify an application provider registered by an international registration authority as specified in ISO/IEC 7816-5.

— PIX has a free coding. It allows the application provider to identify its different applications.

| Registered application provider identifier (RID)<br>(five bytes, first byte set to 'AX') | Proprietary application identifier extension (PIX)<br>(up to eleven bytes) |
|---|---|

**Figure 1 — AID of international category**

Figure 2 shows an AID of national category: it consists of a registered application provider identifier (RID) on five bytes and optionally, a proprietary application identifier extension (PIX) on up to eleven bytes.

— RID is a string of ten consecutive quartets.
  - The first quartet shall be set to 'D', i.e., bits 8 to 5 of the first byte shall be set to 1101.
  - The subsequent three quartets shall be in the range '0' to '9', together forming a country code (numeric part only) as specified in ISO 3166-1 for identifying a national registration authority.
  - The last six quartets (six digits recommended in the range '0' to '9') together shall identify an application provider registered by the national registration authority as specified in ISO/IEC 7816-5.

— PIX has a free coding. It allows the application provider to identify its different applications.

| Registered application provider identifier (RID)<br>(five bytes, first byte set to 'DX') | Proprietary application identifier extension (PIX)<br>(up to eleven bytes) |
|---|---|

**Figure 2 — AID of national category**

Figure 3 shows an AID of standard category: it consists of four to sixteen bytes. The first byte shall be set to 'E8' (i.e., bits 8 to 1 set to 11101000); the values 'E0' to 'E7' and 'E9' to 'EF' are reserved for future use. As specified in ISO/IEC 8825-1, an object identifier shall follow; it shall denote a standard specifying an application (e.g., cryptographic information application; annex B codes ISO/IEC 7816-15 as an illustrative example). As specified by the identified standard, an application identifier extension may follow for identifying different implementations.

| 'E8' | Object identifier identifying an application (see annex B) | Application-specific application identifier extension |
|---|---|---|

**Figure 3 — AID of standard category**

Figure 4 shows an AID of proprietary category: it consists of one to sixteen bytes where the first byte shall be set to 'FX' (bits 8 to 5 set to 1111). In the proprietary category, as application providers are not registered, different application providers may use the same AID.

| Proprietary application identifier (one to sixteen bytes, first byte set to 'FX') |
|---|

**Figure 4 — AID of proprietary category**

## 5.2 Files for applications and data

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

This part of ISO/IEC 7816 supports two categories of files: dedicated file (DF) and elementary file (EF).

— The DFs are intended for hosting applications and / or for grouping files. A DF may be the parent of other files.

— The EFs are intended for storing data. An EF cannot be the parent of another file.

The logical organization of applications and data in the card relies on the following structural hierarchy of DFs.

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

— The other DFs are optional.

As an application DF is a DF hosting an application, such a structural hierarchy of DFs provides a security architecture support for applications and data in the card (see 5.7). Figure 5 illustrates an example of the logical file organization in the card.

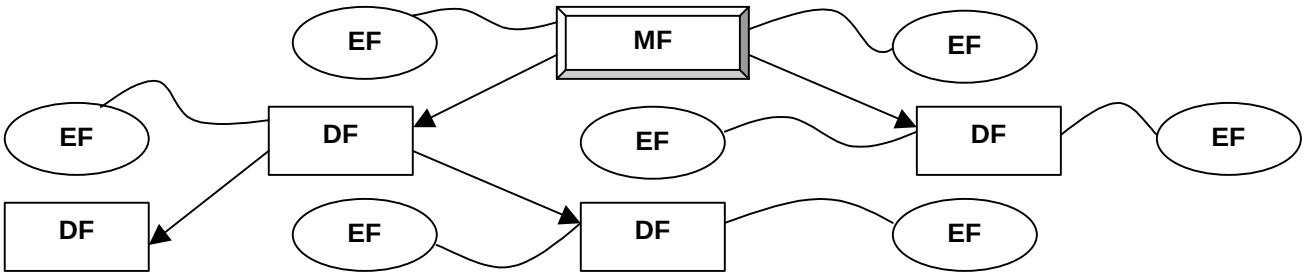


**Figure 5 — Example of logical file organization**

### 5.2.1 File referencing methods

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

**Referencing by DF name** — A DF name may reference any DF. It is a string of one to sixteen bytes. 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.

**Referencing 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 for future use. The value '3FFF' is reserved (see referencing by path). In order to select unambiguously any file by its identifier, all EFs and DFs immediately under a given DF shall have different file identifiers.

**Referencing by path** — A path may reference any file. It is a concatenation of file identifiers. The path begins with the identifier of the MF (absolute path) or of the current DF (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 path allows an unambiguous selection of any file from the MF or from the current DF.

NOTE The path '3F002F00' is reserved for referencing EF.DIR and '3F002F01' for EF.ATR.

**Referencing 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, zero references the current EF. Short EF identifiers cannot be used in a path or as an EF identifier (e.g., in a SELECT command).

NOTE The number 30, i.e., 11110 in binary, is reserved for referencing EF.DIR.

### 5.2.2 Data referencing methods

Two categories of EFs are defined.

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

— A working EF is intended for storing data not interpreted by the card, i.e., data used by the outside world exclusively.

Data may be referenced as data units, as records or as data objects. Three structures of EFs are defined.

— An EF of transparent structure is seen at the interface as a single continuous sequence of data units accessible by commands for handling data units (see 8.2).

— An EF of record structure is seen at the interface as a single continuous sequence of individually identifiable records accessible by commands for handling records (see 8.3).

— An EF of TLV structure is seen at the interface as a set of data objects accessible by commands for handling data objects (see 8.4).

Two attributes are defined for EFs of record structure.

— 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).

For structuring EFs, the card shall support at least one of the five methods shown in Figure 6.

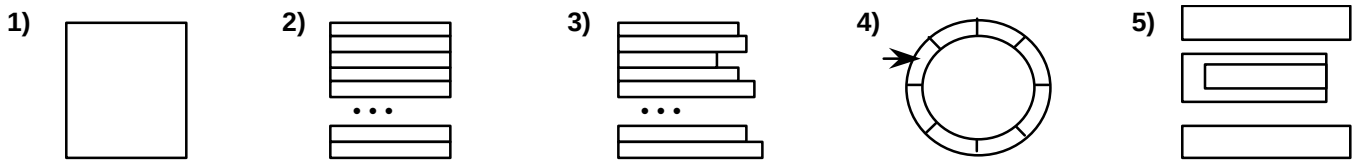


**Figure 6 — EF structures**

1) EF of transparent structure
2) EF of linear structure with records of fixed size
3) EF of linear structure with records of variable size
4) EF of cyclic structure with records of fixed size (the arrow references the most recently written record)
5) EF of TLV structure (possible constructed encoding for BER-TLV data objects, see 5.4.1)

EF.ATR and EF.DIR are two optional working EFs intended for nesting BER-TLV data objects. In the historical bytes, the card service data byte (see Table 15) may indicate their presence and how to access them, i.e., their structure.

— When present, EF.ATR indicates operating characteristics of the card (see also 5.5).

— When present, EF.DIR indicates a list of applications supported by the card (see also 9.3.3)

Data referencing methods, record numbering method and data unit size are EF-dependent features. The card can provide indications in several places: historical bytes (see Table 17, data coding byte in 5.5.2), EF.ATR, control information of any file (see Table 21, file descriptor byte in 5.6.1) within the path from the MF to a given EF. If the card provides indications in several places, then the indication valid for a given EF is in the closest position to that EF within the path from the MF to that EF.

#### 5.2.2.1 Data unit referencing

Within each EF of transparent structure, an offset shall reference each data unit. Reference to a data unit outside an EF is an error. If the data coding byte is absent in the historical bytes, in EF.ATR and in the control information of every file within the path from the MF to a given EF, then the data unit size is one byte for that EF. From zero for the first data unit of the EF, the offset is incremented by one for every subsequent data unit. In commands for handling data units (see 8.2), the offset is either

— a number in the range zero to 255, coded on the eight bits of parameter P2, or

— a number in the range zero to 32 767, coded on bits 7 to 1 of parameter P1 followed by parameter P2.

#### 5.2.2.2 Record referencing

Within each EF of record structure, a record number and / or a record identifier shall reference each record. Reference to a record outside an EF is an error. Each record number and each record identifier is a number from one to 254, coded on one byte from '01' to 'FE' in parameter P1 in commands for handling records (see 8.3). The value '00' is reserved for special purposes. The value 'FF' is reserved for future use.

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

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

**Referencing by record identifier** — Each record identifier is provided by an application. Within an EF of record structure, 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 (see 5.4.2) in a data field (see 5.3.4), then the record identifier is the first byte of the data object.

Referencing by record identifier shall induce the management of a record pointer. A reset of the card, a SELECT and any command using a valid short EF identifier for accessing an EF can affect the record pointer. Each time a reference is made with a record identifier, an indication shall fix the logical position of the target record: the first or last occurrence, the next or previous occurrence relative to the record pointer.

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

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

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

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

— 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.

— 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.

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

**Referencing by record number** — Each record number is unique and sequential.

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

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

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

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

Referencing by record number shall not affect the record pointer.

#### 5.2.2.3 Data object referencing

Within each EF of TLV structure, a tag shall reference each data object (see 5.4). Reference to a data object outside an EF is an error.

Besides the EF structures presented above, BER-TLV data objects may be found in the control information of any file (see 5.6), in the initial data string (see 9.2.1) and more generally, in any command or response data field coded in BER-TLV, e.g., when using secure messaging (see 6).

## 5.3 Command-response pairs

Table 2 illustrates a command-response pair according to ISO/IEC 7816-3. It consists of a command APDU followed by a response APDU in the opposite direction.

**Table 2 — Command-response pair**

| Field | Length | Description | Code |
|---|---|---|---|
| Class byte | 1 | Class of instruction | CLA |
| Instruction byte | 1 | Instruction code | INS |
| Parameter bytes | 2 | Instruction parameters | P1-P2 |
| Lc field | 0, 1 or 3 | The Lc field fixes number Lc | - |
| Command data field | Lc | String of Lc bytes | - |
| Le field | 0, 1, 2 or 3 | The Le field fixes number Le | - |
| Response data field | Lr | String of Lr bytes | - |
| Status bytes | 2 | Command processing status | SW1-SW2 |

ISO/IEC 7816-3 specifies the following structures for Lc and Le fields in command-response pairs.

— The Lc field may be absent. If present, the Lc field shall consist of either
  - one byte different from '00' (short Lc field), or
  - one byte set to '00' followed by two bytes different from '0000' (extended Lc field).

— The Le field may be absent. If present, the Le field shall consist of either
  - one byte with any value (short Le field when there is no Lc field or when a short Lc field is present), or
  - one byte set to '00' followed by two bytes with any value (extended Le field when there is no Lc field), or
  - two bytes with any value (extended Le field when an extended Lc field is present).

The following three numbers fix the relationships between length fields and data fields in command-response pairs.

— Number Lc is the number of bytes present in the command data field.
  - It is zero when there is no Lc field.
  - It is in the range one to 255 when a short Lc field is present.
  - It is in the range one to 65 535 when an extended Lc field is present.

— Number Le is the maximum number of bytes expected in the response data field. If the Le field contains only bytes set to '00', then number Le is maximum.
  - It is zero when there is no Le field.
  - It is in the range one to 256 when a short Le field is present.
  - It is in the range one to 65 536 when an extended Le field is present.

— Number Lr is the number of bytes present in the response data field.
  - It is in the range zero to number Le.

Consequently, the absence of Le field is the standard way for not receiving a response data field.

If the third software function table (see Table 18 in 5.5.2.7, card capabilities) does not explicitly state "extended Lc and Le fields", then only short length fields shall apply. Otherwise, extended length fields may also apply.

Any command-response pair shall be completed before initiating another one. There shall be no interleaving of command-response pairs across the interface.

The subsequent five clauses specify coding conventions for the following fields within command-response pairs. Unless otherwise specified, in those fields, RFU bytes are set to '00' and RFU bits to zero.

1) Class byte
2) Instruction byte
3) Parameter bytes
4) Data fields
5) Status bytes

Subsequently, two clauses specify mechanisms for grouping interindustry command-response pairs.

6) Command chaining
7) Logical channels

### 5.3.1 Class byte

Denoted as CLA, the class byte indicates to what extent the command-response pair complies with this part of ISO/IEC 7816. Bit 8 of CLA sets the difference between the interindustry classes of commands (bit 8 set to zero) and the other classes of commands (bit 8 set to one).

Table 3 illustrates CLA for the interindustry classes. In the basic interindustry class (bits 8 and 7 set to 00), CLA indicates a command chaining (bit 5), a format of secure messaging (bits 4 and 3) and a logical channel number (bits 6, 2 and 1) from zero to seven.

**Table 3 — CLA for the interindustry classes**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | x | x | x | x | x | x | **Basic interindustry class** |
| 0 | 0 | - | x | - | - | - | - | **Command chaining** (according to 5.3.6) |
| 0 | 0 | - | 0 | - | - | - | - | — The command is the last or only command of a chain |
| 0 | 0 | - | 1 | - | - | - | - | — The command is not the last command of a chain |
| 0 | 0 | - | - | x | x | - | - | **Secure messaging** |
| 0 | 0 | - | - | 0 | 0 | - | - | — No SM or no indication |
| 0 | 0 | - | - | 0 | 1 | - | - | — Proprietary SM format |
| 0 | 0 | - | - | 1 | 0 | - | - | — SM according to 6, command header not authenticated |
| 0 | 0 | - | - | 1 | 1 | - | - | — SM according to 6, command header authenticated (according to 6.2.3.1) |
| 0 | 0 | x | - | - | - | x | x | **Logical channel number from zero to seven** (according to 5.3.7) |
| Bits 6, 2 and 1 are set to 000 when logical channels are not used or when the basic logical channel is selected. | | | | | | | | |
| 0 | 1 | x | x | x | x | x | x | **Reserved for other interindustry classes** |

Table 4 illustrates CLA for the other classes. If bits 8 to 5 of CLA are set to 1000, 1001 or 1010, then bits 4 and 3 indicate a format of secure messaging and bits 2 and 1 code a logical channel number from zero to three.

**Table 4 — CLA for the other classes**

| Value | Meaning |
|---|---|
| '8X', '9X' | Structure of the command-response pair according to this part of ISO/IEC 7816<br>Except for quartet 'X' compliant with Table 3, coding and meaning of the command-response pair are proprietary |
| 'AX' | Unless otherwise specified by the application context, structure and coding of the command-response pair according to this part of ISO/IEC 7816 including quartet 'X' compliant with Table 3 |
| 'B0' to 'CF' | Structure of the command-response pair according to this part of ISO/IEC 7816 |
| 'D0' to 'FE' | Proprietary structure and coding of the command-response pair |
| 'FF' | Reserved for protocol and parameters selection (according to ISO/IEC 7816-3) |

### 5.3.2 Instruction byte

Denoted as INS, the instruction byte indicates the command to process. It shall allow any transmission protocol defined in ISO/IEC 7816-3. Table 5 shows the INS codes that are consequently invalid.

**Table 5 — Invalid INS codes**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Value |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | x | x | x | x | '6X' |
| 1 | 0 | 0 | 1 | x | x | x | x | '9X' |

For any interindustry command, the parity of the INS code is reserved for indicating a data field format as follows.

— When bit 1 is set to zero (even INS code), no indication is provided.

— Otherwise (odd INS code), the unsecured data fields (when present) shall be BER-TLV coded (see 5.4.1).

Table 6 shows all the INS codes defined in ISO/IEC 7816. The left side of the table lists the command names in the alphabetic order. The right side of the table lists the INS codes in the numeric order.

— In this part of ISO/IEC 7816, clause 8 specifies interindustry commands for interchange.

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

— ISO/IEC 7816-8 specifies interindustry commands for a cryptographic toolbox.

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

**Table 6 — INS codes of interindustry commands**

| Command name | INS code | Clause | | INS code | Clause | Command name |
|---|---|---|---|---|---|---|
| ACTIVATE FILE | '44' | Part 9 | | '04' | Part 9 | DEACTIVATE FILE |
| APPEND RECORD | 'E2' | 8.3.5 | | '0E' | 8.2.4 | ERASE BINARY |
| CHANGE REFERENCE DATA | '24' | 8.5.6 | | '10' | Part 7 | PERFORM SCQL OPERATION |
| CREATE FILE | 'E0' | Part 9 | | '12' | Part 7 | PERFORM TRANSACTION OPERATION |
| DEACTIVATE FILE | '04' | Part 9 | | '14' | Part 7 | PERFORM USER OPERATION |
| DELETE FILE | 'E4' | Part 9 | | '20' | 8.5.5 | VERIFY |
| DISABLE VERIFICATION REQUIREMENT | '26' | 8.5.8 | | '22' | 8.5.10 | MANAGE SECURITY ENVIRONMENT |
| ENABLE VERIFICATION REQUIREMENT | '28' | 8.5.7 | | '24' | 8.5.6 | CHANGE REFERENCE DATA |
| ENVELOPE | 'C2' | 8.6.2 | | '26' | 8.5.8 | DISABLE VERIFICATION REQUIREMENT |
| ERASE BINARY | '0E' | 8.2.4 | | '28' | 8.5.7 | ENABLE VERIFICATION REQUIREMENT |
| EXTERNAL (/ MUTUAL) AUTHENTICATE | '82' | 8.5.3 | | '2A' | Part 8 | PERFORM SECURITY OPERATION |
| GENERAL AUTHENTICATE | '86' | 8.5.4 | | '2C' | 8.5.9 | RESET RETRY COUNTER |
| GENERATE PUBLIC-KEY PAIR | '46' | Part 8 | | '44' | Part 9 | ACTIVATE FILE |
| GET CHALLENGE | '84' | 8.5.2 | | '46' | Part 8 | GENERATE PUBLIC-KEY PAIR |
| GET DATA | 'CA' | 8.4.1 | | '70' | 8.1.2 | MANAGE CHANNEL |
| GET RESPONSE | 'C0' | 8.6.1 | | '82' | 8.5.3 | EXTERNAL (/ MUTUAL) AUTHENTICATE |
| INTERNAL AUTHENTICATE | '88' | 8.5.1 | | '84' | 8.5.2 | GET CHALLENGE |
| MANAGE CHANNEL | '70' | 8.1.2 | | '86' | 8.5.4 | GENERAL AUTHENTICATE |
| MANAGE SECURITY ENVIRONMENT | '22' | 8.5.10 | | '88' | 8.5.1 | INTERNAL AUTHENTICATE |
| PERFORM SCQL OPERATION | '10' | Part 7 | | 'A0' | 8.2.5 | SEARCH BINARY |
| PERFORM SECURITY OPERATION | '2A' | Part 8 | | 'A2' | 8.3.5 | SEARCH RECORD |
| PERFORM TRANSACTION OPERATION | '12' | Part 7 | | 'A4' | 8.1.1 | SELECT |
| PERFORM USER OPERATION | '14' | Part 7 | | 'B0' | 8.2.1 | READ BINARY |
| PUT DATA | 'DA' | 8.4.2 | | 'B2' | 8.3.1 | READ RECORD (S) |
| READ BINARY | 'B0' | 8.2.1 | | 'C0' | 8.6.1 | GET RESPONSE |
| READ RECORD (S) | 'B2' | 8.3.1 | | 'C2' | 8.6.2 | ENVELOPE |
| RESET RETRY COUNTER | '2C' | 8.5.9 | | 'CA' | 8.4.1 | GET DATA |
| SEARCH BINARY | 'A0' | 8.2.5 | | 'D0' | 8.2.2 | WRITE BINARY |
| SEARCH RECORD | 'A2' | 8.3.5 | | 'D2' | 8.3.2 | WRITE RECORD |
| SELECT | 'A4' | 8.1.1 | | 'D6' | 8.2.3 | UPDATE BINARY |
| TERMINATE CARD USAGE | 'FE' | Part 9 | | 'DA' | 8.4.2 | PUT DATA |
| TERMINATE DF | 'E6' | Part 9 | | 'DC' | 8.3.3 | UPDATE RECORD |
| TERMINATE EF | 'E8' | Part 9 | | 'E0' | Part 9 | CREATE FILE |
| UPDATE BINARY | 'D6' | 8.2.3 | | 'E2' | 8.3.4 | APPEND RECORD |
| UPDATE RECORD | 'DC' | 8.3.3 | | 'E4' | Part 9 | DELETE FILE |
| VERIFY | '20' | 8.5.5 | | 'E6' | Part 9 | TERMINATE DF |
| WRITE BINARY | 'D0' | 8.2.2 | | 'E8' | Part 9 | TERMINATE EF |
| WRITE RECORD | 'D2' | 8.3.2 | | 'FE' | Part 9 | TERMINATE CARD USAGE |

— For every interindustry command, only the even INS code is present in the table; the odd INS code indicates BER-TLV coded data field(s).
— The other valid INS codes are reserved for ISO/IEC JTC 1/SC 17.

### 5.3.3 Parameter bytes

Denoted as P1-P2, the parameter bytes fix controls and options for processing the command. They may have any value. If a parameter byte provides no further qualification, then it shall be set to '00'.

### 5.3.4 Data fields

If present, each data field shall have one of the following three structures.

— Each TLV-coded data field shall consist of one or more data objects (see 5.4).

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

— ISO/IEC 7816 does not specify the structure of the proprietary-coded data fields.

### 5.3.5 Status bytes

Denoted as SW1-SW2, the status bytes tell the processing state in the card. Figure 7 shows the structural scheme for coding the values of SW1-SW2 defined in this part of ISO/IEC 7816.

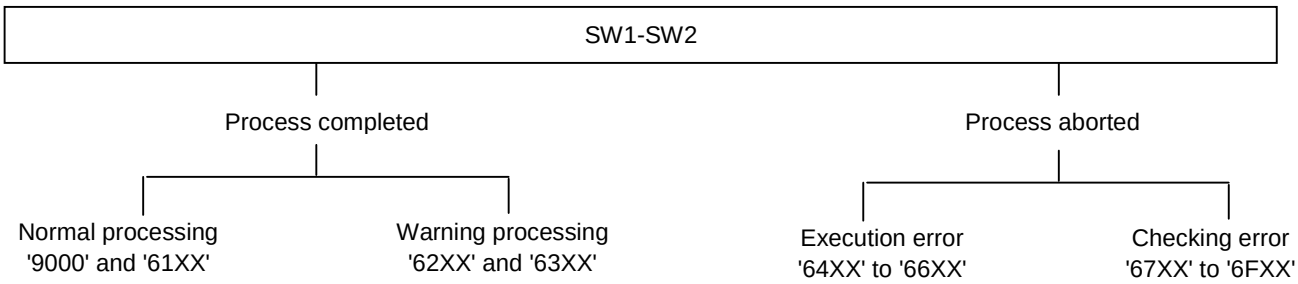


**Figure 7 — Structural scheme of status bytes**

Due to specifications in ISO/IEC 7816-3, this part does not define SW1-SW2 set to '60XX', '67XX', '6BXX', '6DXX', '6EXX', '6FXX' and '9XXX', except for '6700', '6B00', '6D00', '6E00', '6F00' and '9000'. Table 7 shows the general meaning of the values of SW1-SW2 defined in this part of ISO/IEC 7816.

**Table 7 — General meaning of SW1-SW2**

| | SW1-SW2 | Meaning |
|---|---|---|
| **Normal processing** | '9000'<br>'61XX' | No further qualification<br>SW2 tells the number of response bytes still available (see text below) |
| **Warning processing** | '62XX'<br>'63XX' | State of non-volatile memory is unchanged (further qualification in SW2)<br>State of non-volatile memory has changed (further qualification in SW2) |
| **Execution error** | '64XX'<br>'65XX'<br>'66XX' | State of non-volatile memory is unchanged (further qualification in SW2)<br>State of non-volatile memory has changed (further qualification in SW2)<br>Security-related issues |
| **Checking error** | '6700'<br>'68XX'<br>'69XX'<br>'6AXX'<br>'6B00'<br>'6CXX'<br>'6D00'<br>'6E00'<br>'6F00' | Wrong length<br>Functions in CLA not supported (further qualification in SW2)<br>Command not allowed (further qualification in SW2)<br>Wrong parameters P1-P2 (further qualification in SW2)<br>Wrong parameters P1-P2<br>Wrong length Le; SW2 tells the exact length (see text below)<br>Instruction code not supported or invalid<br>Class not supported<br>No precise diagnosis |

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

Two values of SW1 are defined whichever transmission protocol is used (see also ISO/IEC 7816-3).

— If a command is processed with a response where SW1 is set to '61', then before issuing any other command, a GET RESPONSE command may be issued using SW2 as short Le field (number of data bytes still available).

— If a command is aborted with a response where SW1 is set to '6C', then before issuing any other command, the same command may be re-issued using SW2 as short Le field (exact number of requested data bytes).

NOTE At application level, '9FXX' may offer a functionality similar to that offered by '61XX'. However, applications may use '9FXX' for other purposes.

Table 8 lists the status conditions of the interindustry commands specified in this part of ISO/IEC 7816.

**Table 8 — SW1-SW2 of interindustry commands**

| SW1 | SW2 | Meaning |
|---|---|---|
| '62' | '00'<br>'02' to '80'<br>'81'<br>'82'<br>'83'<br>'84' | No information given<br>Triggering by the card (see 9.1.1)<br>Part of returned data may be corrupted<br>End of file or record reached before reading Le bytes<br>Selected file invalidated<br>File control information not formatted according to 5.6 |
| '63' | '00'<br>'81'<br>'CX' | No information given<br>File filled up by the last write<br>Counter provided by 'X' fixing a number 0 to 15 (exact meaning depending on the command) |
| '64' | '00'<br>'01'<br>'02' to '80' | Execution error<br>Immediate reply required by the card<br>Triggering by the card (see 9.1.1) |
| '65' | '00'<br>'81' | No information given<br>Memory failure |
| '68' | '00'<br>'81'<br>'82'<br>'83'<br>'84' | No information given<br>Logical channel not supported<br>Secure messaging not supported<br>Final chained command expected<br>Command chaining not supported |
| '69' | '00'<br>'81'<br>'82'<br>'83'<br>'84'<br>'85'<br>'86'<br>'87'<br>'88' | No information given<br>Command incompatible with file structure<br>Security status not satisfied<br>Authentication method blocked<br>Referenced data invalidated<br>Conditions of use not satisfied<br>Command not allowed (no current EF)<br>Expected secure messaging data objects missing<br>Incorrect secure messaging data objects |
| '6A' | '00'<br>'80'<br>'81'<br>'82'<br>'83'<br>'84'<br>'85'<br>'86'<br>'87'<br>'88'<br>'89'<br>'8A' | No information given<br>Incorrect parameters in the command data field<br>Function not supported<br>File not found<br>Record not found<br>Not enough memory space in the file<br>Lc inconsistent with TLV structure<br>Incorrect parameters P1-P2<br>Lc inconsistent with parameters P1-P2<br>Referenced data not found<br>File already exists<br>DF name already exists |
| — The other values of SW2 are reserved for ISO/IEC JTC 1/SC 17, except for the range 'F0' to 'FF' not defined in this part of ISO/IEC 7816. | | |

### 5.3.6 Command chaining

This clause specifies a mechanism whereby consecutive interindustry command-response pairs can be chained for executing multi-step processes. Once initiated, a chain shall be completed before initiating a command-response pair not part of the chain; otherwise, the card behaviour is not specified.

For chaining interindustry commands of the basic class (bits 8 and 7 of CLA set to 00, see 5.3.1), bit 5 of CLA shall be used as follows while the other five bits (bits 6, 4, 3, 2 and 1) of CLA are constant.

— If bits 8, 7 and 5 are all set to zero in CLA, then the command is the last or only command of a chain.

— If bits 8 and 7 are set 00 and bit 5 to one in CLA, then the command is not the last command of a chain.

In response to a chained interindustry command (i.e., bits 8, 7 and 5 set to 001 in CLA), SW1-SW2 set to '9000' means that the processing has been successful so far; moreover, the following specific error conditions may occur.

— If SW1-SW2 is set to '6883', then the final chained command is expected.

— If SW1-SW2 is set to '6884', then command chaining is not supported.

### 5.3.7 Logical channels

This clause specifies a mechanism whereby command-response pairs can refer to logical channels. If the card supports the mechanism, then the card shall indicate the maximum number of available logical channels in the third software function table in the historical bytes (see Table 18 in 5.5.2.7, card capabilities) or in EF.ATR.

The mechanism is available for any interindustry command of the basic class (bits 8 and 7 set to 00 in CLA, see 5.3.1) and for three other classes of commands (bits 8 to 5 set to 1000, 1001 or 1010 in CLA, see 5.3.1). In any command referring to a certain logical channel, CLA codes the respective logical channel number as follows.

— If bits 8 and 7 are set to 00 (basic interindustry class), then bits 6, 2 and 1 code a number from zero to seven.

— If bits 8 to 5 are set to 1000, 1001 or 1010, then bits 2 and 1 code a number from zero to three.

— The number of the basic logical channel is zero. The basic logical channel is permanently available.

As ISO/IEC 7816 supports only command-response pairs that shall be completed before initiating a subsequent one, there shall be no interleaving of command-response pairs across logical channels; between the receipt of a command and the sending of the response to that command, only one logical channel shall be active. There shall be independence of activity on one logical channel from activity on another one; i.e., command interdependencies on one logical channel shall be independent of command interdependencies on another logical channel. However, logical channels may share application-dependent security statuses and therefore may have security-related command interdependencies across logical channels (e.g., password verification).

A logical channel, as seen at the interface, works as a logical link to a DF and possibly to an EF within that DF.

— There is one current DF and possibly one current EF per logical channel as a result of the behaviour of
  - SELECT and MANAGE CHANNEL commands (see 8.1, selection),
  - any command using a short EF identifier for accessing EFs (see 8.2 and 8.3, data unit or record handling).

— If not explicitly excluded by the file descriptor byte (see 5.6.1, file accessibility in Table 21) in the file control parameters, more than one logical channel may be opened to the same DF and also possibly to the same EF.

**Opening a logical channel —** A logical channel is opened by successful completion of either

— a SELECT command by assigning a logical channel number other than zero in CLA, or

— the open function of a MANAGE CHANNEL command either assigning a logical channel number other than zero in P1-P2 or requesting the card to assign a logical channel number other than zero in the response data field.

**Closing a logical channel —** Once open, a logical channel remains open until explicitly closed by successful completion of the close function of a MANAGE CHANNEL command using the logical channel number other than zero coded in CLA (see 5.3.1). After closing, the logical channel will be available for re-use. The basic logical channel shall not be closed.

## 5.4 Data objects

In ISO/IEC 7816, '00' or 'FF' bytes without any meaning may be present in data fields before, between or after data objects (e.g., due to erasure or modification of data objects within an EF). Consequently, the values '00' and 'FF' are invalid for the first byte of any data object. This part of ISO/IEC 7816 specifies two categories of data objects.

— BER-TLV data objects

— SIMPLE-TLV data objects

### 5.4.1 BER-TLV data objects

Each BER-TLV data object 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.

— The tag field consists of one or more consecutive bytes. It fixes a class, an encoding and a number.

— The length field consists of one or more consecutive bytes. It fixes a number L.

— If number L is not zero, then the value field consists of L consecutive bytes. If number L is zero, then there is no value field, i.e., the data object is empty.

#### 5.4.1.1 BER-TLV tag fields

Bits 8 and 7 of the first byte of the tag field fix a class.

— The value 00 introduces a data object of universal class.

— The value 01 introduces a data object of application class.

— The value 10 introduces a data object of context-specific class.

— The value 11 introduces a data object of private class.

Bit 6 of the first byte of the tag field fixes an encoding.

— The value 0 introduces a primitive encoding of the data object.

— The value 1 introduces a constructed encoding of the data object.

If bits 5 to 1 of the first byte of the tag field are not all set to one, then they fix a tag number from zero to thirty and the tag field consists of a single byte.

Otherwise (bits 5 to 1 all set to one), the tag field continues on one or more subsequent bytes.

— Bit 8 of each subsequent byte shall be set to one, unless it is the last subsequent byte.

— Bits 7 to 1 of the first subsequent byte shall not be all set to zero.

— Bits 7 to 1 of the first subsequent byte, followed by bits 7 to 1 of each further subsequent byte, up to and including bits 7 to 1 of the last subsequent byte fix a tag number.

ISO/IEC 7816 supports BER-TLV tag fields of one, two and three bytes. Longer tag fields are reserved for future use. Table 9 shows the first byte of BER-TLV tag fields; the values '00' and 'FF' are invalid for the first byte.

**Table 9 — First byte of BER-TLV tag fields in ISO/IEC 7816**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | - | - | - | - | - | - | Universal class, not defined in ISO/IEC 7816 ('00' is invalid) |
| 0 | 1 | - | - | - | - | - | - | Application class, unambiguous identification defined in this part of ISO/IEC 7816 |
| 1 | 0 | - | - | - | - | - | - | Context-specific class, defined in ISO/IEC 7816 |
| 1 | 1 | - | - | - | - | - | - | Private class, not defined in ISO/IEC 7816 ('FF' is invalid) |
| - | - | 0 | - | - | - | - | - | Primitive encoding |
| - | - | 1 | - | - | - | - | - | Constructed encoding |
| - | - | - | Not all set to one | | | | | Tag number from zero to thirty (short tag field, i.e., consisting of a single byte) |
| - | - | - | 1 | 1 | 1 | 1 | 1 | Tag number greater than thirty (long tag field, i.e., consisting of two or three bytes) |

In BER-TLV tag fields of two or more bytes, the values '00' to '1E' and '80' are invalid for the second byte.

— In a two-byte tag field, the second byte consists of bit 8 set to zero and bits 7 to 1 fixing a number greater than thirty. Therefore the second byte is in the range '1F' to '7F' and the tag number in the range 31 to 127.

— In a three-byte tag field, the second byte consists of bit 8 set to one and bits 7 to 1 not all set to zero; the third byte consists of bit 8 set to zero and bits 7 to 1 with any value. Therefore the second byte is in the range '81' to 'FF', the third byte in the range '00' to '7F' and the tag number in the range 128 to 16383.

#### 5.4.1.2 BER-TLV length fields

In short form, the length field consists of a single byte where bit 8 is set to zero and bits 7 to 1 fix the number of bytes in the value field. One byte can thus encode any number L from zero to 127.

NOTE Any number L from one to 127 is coded in the same way in BER-TLV length field as in APDU length field. The coding differs for zero, 128 and more.

In long form, the length field consists of two or more bytes. Bit 8 of the first byte is set to one and bits 7 to 1 are not all equal, thus fixing the number of subsequent bytes in the length field. Those subsequent bytes fix the number of bytes in the value field. A length field of three bytes can thus encode any number L up to 65 535.

NOTE ISO/IEC 7816 does not use the "indefinite length" specified by the basic encoding rules of ASN.1.

ISO/IEC 7816 supports BER-TLV length fields of one, two and three bytes as illustrated by Table 10. The values '80' and '83' to 'FF' are invalid for the first byte of BER-TLV length fields in ISO/IEC 7816.

**Table 10 — BER-TLV length fields in ISO/IEC 7816**

| Number of bytes | First byte | Second byte | Third byte | Number L |
|---|---|---|---|---|
| 1 | '00' to '7F' | - | - | 0 to 127 |
| 2 | '81' | '00' to 'FF' | - | 0 to 255 |
| 3 | '82' | '0000' to 'FFFF' | | 0 to 65 535 |

#### 5.4.2 SIMPLE-TLV data objects

Each SIMPLE-TLV data object shall consist of two or three consecutive fields.

— The tag field consists of a single byte fixing a tag number from one to 254. The values '00' and 'FF' are invalid for SIMPLE-TLV 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 different from 'FF', then the length field consists of a single byte fixing a number L in the range zero to 254.
- If the first byte is set to 'FF', then the length field continues on the two subsequent bytes fixing a number L in the range zero to 65 535.

— If number L is not zero, then the value field consists of L consecutive bytes. If number L is zero, then there is no value field, i.e., the data object is empty.

#### 5.4.3 Data fields, data objects and data elements

For any interindustry command in the basic class (bits 8 and 7 set to 00 in byte CLA, see 5.3.1), bit 1 set to one in INS indicates that, when present, the data fields of the unsecured command-response pair are BER-TLV coded.

— Any BER-TLV data object is denoted as {T-L-V} with a tag field followed by a length field fixing a number L. Depending on whether number L is zero or not, the value field is absent (the data object is empty) or present.

— Any constructed BER-TLV data object is denoted as {T-L-{T1-L1-V1} … -{Tn-Ln-Vn}} with a tag field followed by a length field fixing a number L. If number L is not zero, then the value field of the constructed data object, i.e., the template, consists of one or more BER-TLV data objects, each one consisting of a tag field, a length field fixing a number and if the number is not zero, a value field.

Some data fields (e.g., commands for handling data units, see 8.2), the value fields of SIMPLE-TLV data objects and the value fields of some primitive BER-TLV data objects are intended for nesting data elements fixed by the specifications of the command or by those of the data object.

Some data fields (e.g., commands for handling records, see 8.3) and the value fields of some primitive BER-TLV data objects are intended for nesting SIMPLE-TLV data objects.

Some data fields (e.g., commands for handling data objects, see 8.4) and the value fields of constructed BER-TLV data objects, i.e., the templates, are intended for nesting BER-TLV data objects.

### 5.4.4 Identification of data elements

The identification of data elements relies on the following principles.

1) If the number of bits representing a data element is not a multiple of eight, then the mapping into a byte string should be defined in the context of the respective data element. If not specified otherwise, the appropriate number of bits shall be set to one in the last byte starting from bit 1.

2) At the interface between the card and the interface device, a data element is generally presented in the value field of a BER-TLV data object.

3) 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.

4) Within ISO/IEC 7816, a BER-TLV tag denotes a type of data element (see ISO/IEC 8825-1).

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

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

7) The subsequent clauses and other parts of ISO/IEC 7816 allocate some tags of application class (first byte in the range '40' to '7F'). Every application class tag not defined in ISO/IEC 7816 is reserved for ISO/IEC JTC 1/SC 17.

8) All the interindustry data objects are in the application class, except for some data objects of universal class (e.g., object identifier, tag '06'). This part of ISO/IEC 7816 specifies many interindustry data elements. In addition to defining further interindustry data elements, ISO/IEC 7816-6 maintains an exhaustive list of all the interindustry data elements defined in ISO/IEC 7816.

9) There may be multiple occurrences of the same interindustry data object within the card.

10) In command and response data fields, all the data objects of the context-specific class (first byte in the range '80' to 'BF') shall be nested within interindustry templates, except for file control information (see 5.6) and secure messaging (see 6).

11) The context according to which a BER-TLV data object is identified depends either on the nesting of the data object in a BER-TLV template or on the application currently selected. When no application is selected, all BER-TLV data objects shall be interpreted according to the subsequent tag allocation schemes.

Illustrated by annex B, the subsequent three clauses define tag allocation schemes for identifying interindustry data objects present in data fields. When needed, those tag allocation schemes use the interindustry data objects shown in Table 11 for notifying an authority responsible for tag allocation.

**Table 11 — Interindustry data objects for tag allocation authority**

| Tag | Value |
|---|---|
| '06' | Object identifier (coding specified in ISO/IEC 8825-1, see examples in annex B) |
| '41' | Country code and national data (coding and registration specified in ISO 3166-1, see also 5.5.2.1) |
| '42' | Issuer identification number (coding and registration specified in ISO/IEC 7812-1, see also 5.5.2.1) |
| '4F' | Application identifier (AID, see 5.1 and also 5.5.2.2) |

#### 5.4.4.1 Compatible tag allocation schemes

These tag allocation schemes use interindustry data objects and further data objects.

These further data objects shall be nested within interindustry templates with tags '70' to '77'. In these templates, the meaning of the application class tags is not defined in ISO/IEC 7816 except for tags '41', '42' and '4F' for identifying tag allocation authorities as defined in Table 11.

The use of the context-specific class (first byte in the range '80' to 'BF') within interindustry templates with tags '65' (cardholder-related data), '66' (card data), '67' (authentication data), '6E' (application-related data) is deprecated.

In order to identify a compatible tag allocation scheme and the authority responsible for the scheme, an interindustry template with tag '78' may be used. If present, such a template shall contain one of the interindustry data objects shown in Table 11, for identifying a tag allocation authority.

— If tag '78' is present in the initial data string or in EF.ATR, then the authority is valid for the entire card.

— If tag '78' is present in the management data of a DF (see 5.6), then the authority is valid within that DF.

#### 5.4.4.2 Coexistent tag allocation schemes

These tag allocation schemes may use tags with an interpretation not defined in ISO/IEC 7816.

In order to identify a coexistent tag allocation scheme and the authority responsible for the scheme, an interindustry template with tag '79' shall be used. If present, such a template shall contain one of the interindustry data objects shown in Table 11, for identifying a tag allocation authority.

— If an authority is valid for the entire card, then tag '79' shall be present in the initial data string or in EF.ATR.

— If an authority is valid within a DF, then tag '79' shall be present in the management data of that DF (see 5.6).

All interindustry data objects shall be nested within interindustry templates with tag '7E'. In such a scheme, tags '79' and '7E' shall not be given another interpretation. Besides '79' and '7E', a coexistent tag allocation scheme shall not reallocate tags '62', '64', '6F' and '7D', reserved for interindustry templates as listed in Table 12.

**Table 12 — Interindustry templates reserved for ISO/IEC 7816**

| Tag | Value |
|---|---|
| '62' | Interindustry template for nesting file control parameter data objects (FCP template, see 5.6) |
| '64' | Interindustry template for nesting file management data objects (FMD template, see 5.6) |
| '6F' | Interindustry template for nesting file control information data objects (FCI template, see 5.6) |
| '7D' | Interindustry template for nesting secure messaging data objects (SM template, see 6) |

#### 5.4.4.3 Independent tag allocation schemes

These tag allocation schemes may use tags with another interpretation than ISO/IEC 7816, but which do not comply with 5.4.4.2. Such tag allocation schemes do not allow interindustry interchange and are not in compliance with this part of ISO/IEC 7816.

The use of interindustry discretionary data objects with tags '53' for presenting discretionary data elements and '73' for nesting discretionary data objects in discretionary templates allows the use of proprietary data objects whilst remaining compliant with this part of ISO/IEC 7816.

## 5.5 Historical bytes

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

The number (up to fifteen) of historical bytes is specified and coded as defined in ISO/IEC 7816-3. If this number is not zero, then one or more historical bytes are present; they consist of up to three consecutive fields.

— A mandatory category indicator byte

— Optional COMPACT-TLV data objects

— A conditional status indicator

The information conveyed by the historical bytes may also be found in EF.ATR (reserved path = '3F002F01').

### 5.5.1 Category indicator byte

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

**Table 13 — Category indicator byte**

| Value | Meaning |
|---|---|
| '00' | A status indicator (three bytes) shall be present at the end of the historical bytes (not in TLV) |
| '10' | Specified in 5.5.4 |
| '80' | A status indicator (one, two or three bytes) may be present in an optional COMPACT-TLV data object |
| '81' to '8F' | Reserved for future use |
| Any other value is proprietary. | |

### 5.5.2 COMPACT-TLV data objects

If the category indicator byte is set to '80', then the remaining historical bytes are one or more optional consecutive COMPACT-TLV data objects. The COMPACT-TLV data objects are deduced from interindustry BER-TLV data objects with tag field '4X' and length field '0Y'. The coding is 'XY' followed by a value field of 'Y' bytes fixing one or more data elements. In this clause, quartet 'X' is referred to as the compact tag and quartet 'Y' as the compact length.

If present in EF.ATR, any interindustry data element defined in this clause shall appear as a BER-TLV data object, i.e., a tag field '4X' and a length field '0Y' followed by a value field of 'Y' bytes.

#### 5.5.2.1 Country or issuer indicator

Referenced by either '1Y' or '2Y', this interindustry data element denotes respectively a country or an issuer (see also tags '41' and '42' in Table 11).

**Table 14 — Country or issuer indicator data object**

| Compact tag | Value |
|---|---|
| '1' | Country code and national data, according to ISO 3166-1 |
| '2' | Issuer identification number, according to ISO/IEC 7812-1 |

The compact tag '1' shall be followed by the appropriate compact length and by three quartets denoting a country as defined in ISO 3166-1. The relevant national standardization body shall choose the subsequent data (odd number of quartets).

The compact tag '2' shall be followed by the appropriate compact length and by an issuer identification number as defined in ISO/IEC 7812-1. If the issuer identification number contains an odd number of quartets, then it shall be right padded with a quartet set to 'F'.

#### 5.5.2.2 Application identifier

Referenced by 'FY', this interindustry data element is an AID (see 5.1). In the historical bytes, in EF.ATR and in the initial data string (see 9.2.1), its presence denotes an implicitly selected application in a single application card.

#### 5.5.2.3 Card service data

Referenced by '31', this interindustry data element denotes the methods available in the card for supporting services described in clause 9. In the absence of this byte, the card supports only the implicit application selection.

**Table 15 — Card service data byte**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| x | x | - | - | - | - | - | - | **Direct application selection** |
| 1 | - | - | - | - | - | - | - | — by full DF name |
| - | 1 | - | - | - | - | - | - | — by partial DF name |
| - | - | x | x | - | - | - | - | **BER-TLV data objects available** |
| - | - | 1 | - | - | - | - | - | — in EF.DIR |
| - | - | - | 1 | - | - | - | - | — in EF.ATR |
| - | - | - | - | x | x | x | x | **EF.DIR and EF.ATR access services** |
| - | - | - | - | 1 | 0 | 0 | 0 | — by the READ BINARY command (transparent structure) |
| - | - | - | - | 0 | 0 | 0 | 0 | — by the READ RECORD (S) command (record structure) |
| - | - | - | - | 0 | 1 | 0 | 0 | — by the GET DATA command (TLV structure) |
| - | - | - | - | Any other value | | | | Reserved for future use |
| EF.DIR and EF.ATR may contain information on selection methods. | | | | | | | | |

#### 5.5.2.4 Initial access data

Referenced by '41', '42' or '45', this interindustry data element fixes a command-to-perform assumed to be the first command after the answer to reset and a possible protocol and parameters selection. Consequently, the data available at this point may not be subsequently retrievable. The command-to-perform is specified in 9.2.1.

#### 5.5.2.5 Card issuer's data

Referenced by '5Y', this interindustry data element is not defined in this part of ISO/IEC 7816. The card issuer defines a length, a structure and a coding.

#### 5.5.2.6 Pre-issuing data

Referenced by '6Y', this interindustry data element is not defined in this part of ISO/IEC 7816. The card manufacturer defines a length, a structure and a coding for stating a card manufacturer, an integrated circuit name, an integrated circuit manufacturer, a ROM mask version, an operating system version, etc.

On a proprietary basis, this data element may contain an integrated circuit manufacturer identifier, numbered and registered according to ISO/IEC 7816-13 (see also 9.4.2), for identifying the manufacturer of the integrated circuit embedded in the card.

#### 5.5.2.7 Card capabilities

Referenced by '71', '72' or '73', this interindustry data element consists of up to three software function tables: either the first table or the first two tables or the three tables.

The second software function table is the "data coding byte" (see also tag '82' in Table 20).

Tables 16 to 18 respectively show the three software function tables.

**Table 16 — First software function table**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| x | x | x | x | x | - | - | - | **DF selection** |
| 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 17 — Second software function table (data coding byte)**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| - | 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 | **Data unit size in quartet** (from one quartet to 64 bytes)<br>(power of 2, e.g., 001 = 2 quartets = one byte, default value) |
| x | - | - | x | x | - | - | - | 0xx0 0xxx (any other value is reserved for future use) |

**Table 18 — Third software function table**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| - | 1 | - | - | - | - | - | - | Extended Lc and Le fields |
| | | | x | x | | | | **Logical channel number assignment** |
| | | | 1 | | | | | — by the card |
| | | | | 1 | | | | — by the interface device |
| - | - | - | 0 | 0 | - | - | - | No logical channel |
| - | - | - | - | - | x | y | z | Maximum number of logical channels (= 4x+2y+z+1) |
| x | - | x | - | - | - | - | - | 0x0x xxxx (any other value is reserved for future use) |

### 5.5.3 Status indicator

If the category indicator byte is set to '00', then the last three historical bytes shall be a status indicator (three bytes), namely a card life cycle status byte denoted as LCS followed by two processing status bytes denoted as SW1-SW2.

If the category indicator byte is set to '80', then a COMPACT-TLV data object referenced by '81', '82' or '83' may contain an interindustry data element fixing a status indicator (one, two or three bytes). The other compact lengths are reserved for ISO/IEC JTC 1/SC 17.

— If the compact length is '1', then the data element is a card life cycle status byte denoted as LCS.

— If the compact length is '2', then the data element is two processing status bytes denoted as SW1-SW2.

— If the compact length is '3', then the data element is LCS followed by SW1-SW2.

LCS shall be interpreted according to 5.6.3 and Table 22; the value '00' tells that the status is not reported.

SW1-SW2 shall be interpreted according to 5.3.5 and Table 8; the value '0000' tells that the status is not reported.

### 5.5.4 DIR data reference

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

## 5.6 File control information

The file control information is the byte string available in response to the SELECT command (see 8.1.1); it may be present for any file. Table 19 introduces three interindustry templates intended for nesting file control information when coded as BER-TLV data objects (see also Table 12). The three templates may be retrieved according to selection options of the SELECT command (see Table 47). If the FCP or FMD option is set, then the use of the corresponding template is mandatory. If the FCI option is set, then the use of the FCI template is optional.

— The FCP template is intended for nesting file control parameters as defined hereafter. Within the FCP template, the context-specific class (first byte in the range '80' to 'BF') is reserved for file control parameters.

— The FMD template is intended for nesting file management data, i.e., interindustry data objects such as an application label as defined in 9.3 and an application expiration date as defined in ISO/IEC 7816-6.

— The FCI template is intended for nesting file control parameters (context-specific class) and file management data (application class).

**Table 19 — Interindustry templates for file control information**

| Tag | Value |
|---|---|
| '62' | Interindustry template for nesting file control parameters (FCP template) |
| '64' | Interindustry template for nesting file management data (FMD template) |
| '6F' | Interindustry template for nesting file control parameters and file management data (FCI template) |

Table 20 lists the file control parameters. All the FCP data objects are in the context-specific class.

**Table 20 — File control parameter data objects**

| Tag | Length | Value | Applies to |
|---|---|---|---|
| '80' | 2 | Number of data bytes in the file, excluding structural information | Any EF of transparent structure |
| '81' | 2 | Number of data bytes in the file, including structural information if any | Any file |
| '82' | 1 | File descriptor byte (according to 5.6.1 and Table 21) | Any file |
| | 2 | File descriptor byte and data coding byte (according to Table 17) | |
| | 3 or 4 | File descriptor byte, data coding byte and maximum record size on one or two bytes | Any EF of record structure |
| | 5 or 6 | File descriptor byte, data coding byte, maximum record size on two bytes and number of records on one or two bytes | |
| '83' | 2 | File identifier (according to 5.2.1) | Any file |
| '84' | 1 to 16 | DF name (according to 5.2.1) | Any DF |
| '85' | Var. | Proprietary information, primitive encoding (i.e., not coded in BER-TLV) | Any file |
| '86' | Var. | Security attribute in proprietary format | Any file |
| '87' | 2 | Identifier of an EF containing an extension of the file control information (see below) | Any DF |
| '88' | 0 or 1 | Short EF identifier (according to 5.6.2) | Any EF |
| '8A' | 1 | Life cycle status byte (LCS according to 5.6.3 and Table 22) | Any file |
| '8B' | Var. | Security attribute referencing the expanded format (according to 7.3 and Table 44) | Any DF |
| '8C' | Var. | Security attribute in compact format (according to 7.1) | Any file |
| '8D' | 2 | Identifier of an EF containing security environment templates (see 6.5) | Any DF |
| 'A0' | Var. | Security attribute template for data objects (according to 7) | Any file |
| 'A1' | Var. | Security attribute template for physical interfaces (according to 7) | Any DF |
| 'A2' | Var. | One or more pairs of data objects:<br>short EF identifier (tag '88') - absolute or relative path (tag '51') | Any DF |
| 'A5' | Var. | Proprietary information, constructed encoding (i.e., coded in BER-TLV) | Any file |
| 'AB' | Var. | Security attribute in expanded format (according to 7.2) | Any file |
| 'AC' | Var. | Identifier of a cryptographic mechanism (according to 5.6.4) | Any DF |

Part of the control information of a DF may additionally be present in a working EF under the control of an application, referenced by tag '87' (see Table 20). The use of the FCP or FCI template is mandatory for coding file control information in such an EF.

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

— '00' or any value from 'C0' to 'FF' — The coding of the subsequent byte string is proprietary.

— Tag '53' — This interindustry data element consists of discretionary data not coded in TLV.

— Tag '73' — This interindustry template consists of discretionary BER-TLV data objects.

### 5.6.1 File descriptor byte

A data object referenced by tag '82' may be present in the control parameters of any file (see Table 20). Table 21 shows the first byte of the data element, i.e., the file descriptor byte.

**Table 21 — 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 one | | | - | - | - | **EF category** | |
| 0 | - | 0 | 0 | 0 | - | - | - | — Working EF | |
| 0 | - | 0 | 0 | 1 | - | - | - | — Internal EF | |
| 0 | - | Any other value | | | - | - | - | — Reserved for proprietary categories of EFs | |
| 0 | - | | | | | | | **EF structure** | **Access services** |
| 0 | - | Not all set to one | | | 0 | 0 | 0 | — No information given | |
| 0 | - | Not all set to one | | | 0 | 0 | 1 | — Transparent structure | Data unit handling |
| 0 | - | Not all set to one | | | 0 | 1 | 0 | — Linear structure, fixed size, no further information | Record handling |
| 0 | - | Not all set to one | | | 0 | 1 | 1 | — Linear structure, fixed size, SIMPLE-TLV | Record handling |
| 0 | - | Not all set to one | | | 1 | 0 | 0 | — Linear structure, variable size, no further information | Record handling |
| 0 | - | Not all set to one | | | 1 | 0 | 1 | — Linear structure, variable size, SIMPLE-TLV | Record handling |
| 0 | - | Not all set to one | | | 1 | 1 | 0 | — Cyclic structure, fixed size, no further information | Record handling |
| 0 | - | Not all set to one | | | 1 | 1 | 1 | — Cyclic structure, fixed size, SIMPLE-TLV | Record handling |
| 0 | - | 1 | 1 | 1 | 0 | 0 | 1 | — Working EF of TLV structure for BER-TLV data objects | Data object handling |
| 0 | - | 1 | 1 | 1 | 0 | 1 | 0 | — Working EF of TLV structure for SIMPLE-TLV data objects | Data object handling |
| 0 | - | 1 | 1 | 1 | Any other value | | | — Reserved for future use | |
| 1 | x | x | x | x | x | x | x | Reserved for future use | |
| "Shareable" means that the file supports at least concurrent access on different logical channels. | | | | | | | | | |

### 5.6.2 Short EF identifier

If the card supports short EF identifiers, then bit 3 shall be set to one in the first software function table if present (see Table 16). The following rules apply for tag '88' in the control parameters of any EF (see Table 20).

— If tag '88' is not present, then the short EF identifier shall consist of bits 5 to 1 of the file identifier (tag '83').

— An empty data object with tag '88' states that the EF has no short EF identifier.

— A byte referenced by tag '88' shall consist of bits 8 to 4 coding a short EF identifier and bits 3 to 1 set to 000.

### 5.6.3 Life cycle status byte

The card, files and other objects each have a life cycle; the life cycle status 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 as an attribute, this clause identifies and defines four primary states of the life cycle in the following order.

1) Creation state
2) Initialisation state
3) Operational state
4) Termination state

Transitions between primary life cycle states are irreversible and occur only from creation to termination. In addition, the application may define secondary life cycle states: each primary state may have reversible secondary states. Changes are controlled by the card and may be performed in a pre-defined order, reflecting reversible or irreversible changes in states. States in the life cycle may be handled by any interindustry command of the "file and card management" group (see ISO/IEC 7816-9 where a clause illustrates the actions of these commands on the life cycle states of a file). Commands may set the value of the life cycle status when they execute. However the card shall maintain the integrity of this value in accordance with this part of ISO/IEC 7816.

A life cycle status may be associated with any object in the card and with the card itself. The card shall use the life cycle status in combination with additional security attributes, to determine whether an operation on an object is in accordance with a security policy. The life cycle status reflects the use of objects according to the following rules.

— If an object is in creation state, then no security attribute for that object shall apply.

— If an object is in initialisation state, then any security attribute specific to this state may apply.

— If an object is in operational state, then every associated security attribute shall apply.

— If an object is in termination state, then the value of the object shall not be modified but the object may be used as specified by its associated security attributes, e.g., it may be deleted.

Coded on one byte, the life cycle status (LCS byte) shall be interpreted according to Table 22; the values '10' to 'FF' are proprietary.

**Table 22 — Life cycle status 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 reserved for future use. | | | | | | | | |

— A card LCS byte may be present in the historical bytes (see 5.5.3). When it has a MF, the card is in, at least, the creation state.

— Referenced by tag '8A', a file LCS byte may be present in the control parameters of any file (see Table 20).

### 5.6.4 Cryptographic mechanism identifier template

Referenced by tag 'AC', one or more cryptographic mechanism identifiers may be present in the control parameters of any DF (see Table 20): each one identifies a cryptographic algorithm and / or a mode of operation. Such a template shall consist of two mandatory data objects and an optional data object as follows.

— The first mandatory data object is a cryptographic mechanism reference data object, tag '80' (see table 29).

— The second mandatory data object is an object identifier, tag '06', as defined in ISO/IEC 8825-1, for identifying an ISO standard: either a cryptographic algorithm register (e.g., ISO/IEC 9979) or specifications of cryptographic algorithms (e.g., ISO/IEC 18033) or specifications of modes of operation (e.g., ISO/IEC 10116).

— The optional data object (tag dependent on the object identifier) indicates parameters.

EXAMPLES (see examples of object identifiers in annex B)

{'AC' - '09' - {'80'-'01'-'01'} - {'06'-'06'-'28818C710201'}}

The object identifier '28818C710201' refers to the first algorithm in ISO/IEC 18033-2.

{'AC' - '09' - {'80'-'01'-'01'} - {'06'-'04'-'28CF0401'}}

Similarly, the object identifier '28CF0401' refers to the first mode of operation in ISO/IEC 10116.

## 5.7 Security architecture

This clause describes security status, security attributes and security mechanisms. Security attributes are compared with the security status to execute commands and / or to access files, data objects and tables & views.

**Security status —** The security status represents the current state possibly achieved after completion of the answer to reset and a possible protocol and parameter selection and / or a single command or a sequence of commands possibly performing authentication procedures. The security status may also result from the completion of a security procedure related to the identification of the involved entities, if any, e.g., by proving the knowledge of a password (e.g., using a VERIFY command) or the knowledge of a key (e.g., using a GET CHALLENGE command followed by an EXTERNAL AUTHENTICATE command or a chain of GENERAL AUTHENTICATE commands), or by secure messaging (e.g., message authentication). Three security statuses are considered.

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

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

— Command-specific security status — It only exists during the execution of 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 the allowed actions and the procedures to perform to complete such actions. Security attributes may be associated with each file and fix the security conditions that shall be satisfied to allow operations on the file. The security attributes of a file depend on its category (DF or EF) and on optional parameters in its file control information and / or in that of its parent file(s).

More generally, the security attributes define the allowed actions and procedures to perform to complete such actions. Card objects that may be protected with security attributes include files, commands, data objects and tables & views. In particular, security attributes may fix any following feature.

— Specify the security status of the card to be in force before accessing data.

— Restrict access to data to certain functions if the card has a particular status.

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

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

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

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

— **Data authentication** — Using internal data, either a secret key or a public key, the card checks redundant data received from the outside world. Alternately, 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. Alternately, 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., for key management and conditional access. In addition to the cryptogram mechanism, data confidentiality can be achieved by data concealment. In this case, the card computes a string of concealing bytes and adds it by exclusive-or to bytes received from or sent to the outside world. This mechanism may be used for protecting privacy and for reducing possibilities of message filtering.

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

# 6 Secure messaging

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

Secure messaging is achieved by applying one or more security mechanisms. Each security mechanism involves an 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 execution of security mechanisms. This specification does not preclude the determination by sequential analysis of which mechanisms and which security items shall be used for processing the remaining part of the data field.

— Two or more security mechanisms may use the same cryptographic algorithm with different modes of operation. The hereafter specified padding rules do not preclude such a feature.

Besides transmission security handled by secure messaging, the card may offer security services to the outside world through security operations available in a cryptographic toolbox (see ISO/IEC 7816-8). Secure messaging, which is a global approach to security services, shares several security-related issues with the cryptographic toolbox, which is an atomic approach to security services.

Table 23 shows the secure messaging data objects defined in the subsequent clauses. All the SM data objects are in the context-specific class.

**Table 23 — Secure messaging data objects**

| Tag | Value |
|---|---|
| '80', '81' | Plain value not coded in BER-TLV |
| '82', '83' | Cryptogram (plain value coded in BER-TLV and including secure messaging data objects) |
| '84', '85' | Cryptogram (plain value coded in BER-TLV, but not including secure messaging data objects) |
| '86', '87' | Padding-content indicator byte followed by cryptogram (plain value not coded in BER-TLV) |
| '8E' | Cryptographic checksum (at least four bytes) |
| '90', '91' | Hash-code |
| '92', '93' | Certificate (not BER-TLV coded data) |
| '94', '95' | Security environment identifier (SEID byte, see 6.5) |
| '96', '97' | Number Le in the unsecured command APDU (one or two bytes) |
| '99' | Processing status of the secured response APDU (new SW1-SW2, two bytes) |
| '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 integrated) |
| 'A4', 'A5' | Control reference template for authentication (AT) |
| '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 computation of a certificate (the concatenated value fields are certified) |
| 'B0', 'B1' | Plain value coded in BER-TLV and including secure messaging data objects |
| 'B2', 'B3' | Plain value coded in BER-TLV, but not including secure messaging data objects |
| '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 any command-response pair using SM, the following specific error conditions may occur.

— If SW1-SW2 is set to '6987', then expected secure messaging data objects are missing.

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

## 6.1 SM format concept

In each command-response pair involving security mechanisms based on cryptography, the data field shall comply with the basic encoding rules of ASN.1 (see ISO/IEC 8825-1), unless otherwise indicated in CLA (see 5.3.1).

The SM format defined in this part of ISO/IEC 7816 may be selected either implicitly, i.e., known before issuing the command, or explicitly, i.e., fixed by CLA (see 5.3.1).

— The data fields shall be BER-TLV coded (see 5.4.1).

— The context-specific class (first byte in the range '80' to 'BF') is reserved for SM data objects.

— In the context-specific class, the tag number parity fixes whether the SM data object shall be integrated (odd tag number) or not (even tag number) in the computation of a data element for authentication (see 6.2.3.1).

— Data objects of other classes may be present (e.g., interindustry data objects). If present, they shall be integrated in the computation of a data element for authentication (see 6.2.3.1).

— Some SM data objects are recursive: the plain value field is still BER-TLV coded and the context-specific class is still reserved for SM data objects.

Within the interindustry template referenced by tag '7D' (see Table 12), the SM format as defined above shall apply.

There are two categories of SM data objects.

— Each basic SM data object conveys either a plain value or an input or result of a security mechanism.

— Each auxiliary SM data object conveys either a security environment identifier (SEID byte, see 6.5), or a control reference template, or a response descriptor template.

## 6.2 Basic SM data objects

### 6.2.1 Data objects for encapsulating plain values

Encapsulation is mandatory in two cases: BER-TLV, including secure messaging, data objects and data not coded in BER-TLV. Encapsulation is optional for BER-TLV, not including secure messaging, data objects. Table 24 shows data objects for encapsulating plain values.

**Table 24 — Data objects for encapsulating plain values**

| Tag | Value |
|---|---|
| 'B0', 'B1' | Plain value coded in BER-TLV and including secure messaging data objects |
| 'B2', 'B3' | Plain value coded in BER-TLV, but not including secure messaging data objects |
| '80', '81' | Plain value not coded in BER-TLV |
| '96', '97' | Number Le in the unsecured command APDU (one or two bytes) |
| '99' | Processing status of the secured response APDU (new SW1-SW2, two bytes) |

### 6.2.2 Data objects for confidentiality

Data objects for confidentiality are intended for encapsulating cryptograms whose plain values consist either of BER-TLV, including secure messaging, data objects, or of BER-TLV, not including secure messaging, data objects, or of not BER-TLV coded data. Table 25 shows data objects for confidentiality.

**Table 25 — Data objects for confidentiality**

| Tag | Value |
|---|---|
| '82', '83' | Cryptogram (plain value coded in BER-TLV and including secure messaging data objects) |
| '84', '85' | Cryptogram (plain value coded in BER-TLV, but not including secure messaging data objects) |
| '86', '87' | Padding-content indicator byte (according to Table 26) followed by cryptogram (plain value not BER-TLV coded) |

A cryptographic mechanism for confidentiality consists of an algorithm in a mode of operation, possibly indicated by a cryptographic mechanism identifier (see 5.6.4). 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 bytes to conceal with a concealing string of the same length. Concealment thus requires no padding and the data objects concealed in the value field are recovered by the same operation.

Padding and / or content have to be indicated when the plain value is not BER-TLV coded. When padding applies but is not indicated, the rules defined in 6.2.3.1 shall apply. Table 26 shows the padding-content indicator byte.

**Table 26 — Padding-content indicator byte**

| Value | Meaning |
|---|---|
| '00'<br>'01'<br>'02' | No further indication<br>Padding as defined in 6.2.3.1<br>No padding |
| '1X'<br><br><br><br>'2X'<br>'3X'<br>'4X' | One or more control words, i.e., keys for enciphering information, not for enciphering keys ('X' is a bitmap)<br>'11' indicates an odd control word<br>'12' indicates an even control word<br>'13' indicates a pair of control words (even control word first)<br>Secret key, i.e., a key for enciphering keys ('X' is a reference)<br>Private key ('X' is a reference)<br>Password ('X' is a reference) |
| '80' to '8E' | Proprietary |
| Any other value is reserved for future use. | |

### 6.2.3 Data objects for authentication

Table 27 shows data objects for authentication.

**Table 27 — Data objects for authentication**

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

#### 6.2.3.1 Cryptographic checksum data element

The computation of a cryptographic checksum (see ISO/IEC 9797) involves an initial check block, a secret key and a block cipher algorithm that need not be reversible. Under the control of the key, the algorithm basically transforms a current input block of k bytes (typically 8, 16 or 20) into a current output block of the same size. The computation of a cryptographic checksum 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 transforming auxiliary data under the control of the key. If the auxiliary data is less than k bytes, then bits set to zero head it up to the block size.

**Sequential stage(s)** — If the command header has to be authenticated, i.e., if CLA (see 5.3.1) is such that bits 4 and 3 are set to 11 and either bits 8 and 7 to 00 or bits 8 to 5 to 1000, 1001 or 1010, 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 integrate any secure messaging data object having an odd tag number and any data object with the first byte outside the range '80' to 'BF'. Those data objects shall be integrated data block by data block in the current check block. The splitting into data blocks shall be performed in the following way.

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

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

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

**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.

#### 6.2.3.2 Digital signature data element

The computation of a digital signature relies on asymmetric cryptographic techniques. There are two categories of digital signature schemes: schemes with appendix (see ISO/IEC 14888) and schemes giving message recovery (see ISO/IEC 9796). The computation generally implies a hash-function (see ISO/IEC 10118). The data input consists of the value field of a digital signature input data object, or of the concatenation of the value fields of data objects forming a digital signature input template. It may also be determined by the mechanism defined in 6.2.3.1.

### 6.2.4 SM impact on command-response pair structures

Figure 8 illustrates a command-response pair as defined in ISO/IEC 7816-3.

| Command header | Command body |
|---|---|
| CLA-INS-P1-P2 | [Lc field] - [Data field] - [Le field] |

| Response body | Response trailer |
|---|---|
| [Data field] | SW1-SW2 |

**Figure 8 — Command-response pair**

By definition, a command-response pair is unsecured when byte CLA is such that bit 4 is set to zero and either bits 8 and 7 to 00 or bits 8 to 5 to 1000, 1001 or 1010. The following rules shall apply when switching bit 4 from zero to one in byte CLA as above, i.e., for securing a command-response pair. The notation CLA* means that bit 4 is set to one and either bits 8 and 7 to 00 or bits 8 to 5 to 1000, 1001 or 1010.

— The command APDU shall be encapsulated as follows.

- If a command data field is present (Lc > 0), then either a plain value data object ('80', '81', 'B2', 'B3', see Table 24) or a data object for confidentiality ('84', '85', '86', '87', see Table 25) shall convey the Lc bytes.
- If a Le field is present, then a Le data object ('96', '97', see Table 24) shall convey number Le coded on one or two bytes. Both zero and the empty Le data object mean the maximum.

— The response APDU shall be encapsulated as follows.
  - If a response data field is present (Lr > 0), then either a plain value data object ('80', '81', 'B2', 'B3', see Table 24) or a data object for confidentiality ('84', '85', '86', '87', see Table 25) shall convey the Lr bytes.
  - If needed, a processing status data object ('99', see Table 24) shall convey the processing status bytes (new SW1-SW2). The empty processing status data object means SW1-SW2 set to '9000'.

If bit 1 is set to one in INS (odd INS code, see 5.3.2), then the unsecured data fields are BER-TLV coded; consequently, tags 'B2', 'B3', '84' and '85' shall be used for their encapsulation. Otherwise, the format of the data fields to protect is not always apparent; therefore tags '80', '81', '86' and '87' are recommended.

Figure 9 shows the corresponding secured command-response pair.

| Command header | Command body |
|---|---|
| CLA*-INS-P1-P2 | [New Lc field] - {[New data field] = [T-Lc-Data bytes] - [T-'01' or '02'-Le]} - [New Le field] |

| Response body | Response trailer |
|---|---|
| [New data field] = [T-Lr-Data bytes] - [T-'02'-New SW1-SW2] | New SW1-SW2 |

**Figure 9 — Secured command-response pair**

— The new data fields may contain further or other SM data objects, e.g., a cryptographic checksum ('8E') or a digital signature ('9E') at the end.

— The new Lc field codes the number of bytes present in the secured command data field.

— The new Le field shall not be present when no data field is expected in the secured response data field; otherwise, it shall contain only bytes set to '00'.

— The new trailer codes the status of the receiving entity after processing the secured command. Moreover as stated above, it may be encapsulated for protection.

Annex C provides illustrative examples of secure messaging.

## 6.3 Auxiliary SM data objects

Table 28 shows auxiliary SM data objects.

**Table 28 — Auxiliary SM data objects**

| Tag | Value |
|---|---|
| '94', '95' | Security environment identifier (SEID byte, see 6.5) |
| 'A4', 'A5' | Control reference template valid for authentication (AT) |
| '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 |

### 6.3.1 Control reference templates

Five control reference templates are respectively valid for authentication (AT), hash-code (HT), cryptographic checksum (CCT), digital signature (DST) and confidentiality (CT). The last possible position of a control reference template is just before the first data object to which the referred mechanism applies. For example, the last possible position of a template valid for cryptographic checksum is just before the first data object to integrate in the computation. Each control reference remains valid until a new control reference is provided for the same mechanism. For example, a command may fix control references for the next command.

Each security mechanism uses a cryptographic algorithm in a mode of operation, 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 data objects grouped in control reference templates. Within the control reference templates, the context-specific class (first byte in the range '80' to 'BF') is reserved for control reference data objects.

#### 6.3.1.1 Control reference data objects in control reference templates

Each control reference template is intended for nesting control reference data objects, all in the context-specific class: 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.

Table 29 lists control reference data objects and indicates to which control reference template they are relevant: authentication (AT), hash-code (HT), cryptographic checksum (CCT), digital signature (DST) or confidentiality (CT) using either symmetric techniques (CT-sym) or asymmetric techniques (CT-asym).

— The cryptographic mechanism reference denotes a cryptographic algorithm in a mode of operation. In the control parameters of any DF (see Table 20), cryptographic mechanism identifier templates (tag 'AC') may explicitly fix the meaning of any cryptographic mechanism reference (see 5.6.4).

— The file reference (file identifier or path, see 5.2.1) 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, fixes the initial check block. If no initial data reference is present and no initial check block is implicitly selected, then the null block shall be used. Moreover, before transmitting the first data object for confidentiality using a stream cipher, a template for confidentiality shall provide auxiliary data for initializing the computation of the string of concealing bytes.

**Table 29 — Control reference data objects in control reference templates**

| Tag | Value | AT | HT | CCT | DST | CT-asym | CT-sym |
|---|---|---|---|---|---|---|---|
| '80' | **Cryptographic mechanism reference** | x | x | x | x | x | x |
| **File and key references** | | | | | | | |
| '81' | — File identifier or path (according to 5.2.1) | x | x | x | x | x | x |
| '82' | — DF name (according to 5.2.1) | x | x | x | x | x | x |
| '83' | — Reference of a secret key (for direct use)<br>— Reference of a public key | x<br>x | x<br>x | x | <br>x | <br>x | x |
| '84' | — Reference for computing a session key<br>— Reference of a private key | x<br>x | | x | <br>x | <br>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<br>L=k, initial value block | | <br>x | x<br>x | | | x |
| **Initial data reference: Auxiliary data elements** (see also 6.4) | | | | | | | |
| '88' | — L=0, previous exchanged challenge plus one<br>L>0, no further indication | | | x | x | x | x |
| '89' to '8D' | — L=0, index of a proprietary data element<br>L>0, value of a proprietary data element | | | x<br>x | | | x |
| '90' | — L=0, hash-code provided by the card | | x | | x | | |
| '91' | — L=0, random number provided by the card<br>L>0, random number | | | x | x<br>x | x<br>x | |
| '92' | — L=0, time stamp provided by the card<br>L>0, time stamp | | x | | x<br>x | x<br>x | |
| '93' | — L=0, previous digital signature counter plus one<br>L>0, digital signature counter | | x | | x<br>x | x<br>x | x<br>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 |
| '8E' | **Cryptogram content reference** (see text below) | | | | | x | x |
| A CRT may contain interindustry data objects, e.g., header list or extended header list (tags '5D' and '4D', see 9.4.1) in HT or DST. | | | | | | | |

In any control reference template for authentication (AT), 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 7.2 and Table 42) or in compliance with the MANAGE SECURITY ENVIRONMENT command (see 8.5.10). Table 30 shows the usage qualifier byte.

**Table 30 — Usage qualifier byte**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 1 | - | - | - | - | - | - | - | Verification (DST, CCT)<br>Encipherment (CT)<br>External authentication (AT) |
| - | 1 | - | - | - | - | - | - | Computation (DST, CCT)<br>Decipherment (CT)<br>Internal authentication (AT) |
| - | - | 1 | - | - | - | - | - | Secure messaging in response data fields (CCT, CT, DST) |
| - | - | - | 1 | - | - | - | - | Secure messaging in command data fields (CCT, CT, DST) |
| - | - | - | - | 1 | - | - | - | User authentication, knowledge based (AT) |
| - | - | - | - | - | 1 | - | - | User authentication, biometry based (AT) |
| - | - | - | - | - | - | x | x | 00 (any other value is reserved for future use) |

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 31 shows the cryptogram descriptor byte.

**Table 31 — Cryptogram descriptor byte**

| Value | Meaning |
|---|---|
| '00' | No further indication |
| '1X' | One or more control words, i.e., keys for enciphering information, not for enciphering keys ('X' is a bitmap)<br>'11' indicates an odd control word<br>'12' indicates an even control word<br>'13' indicates a pair of control words (even control word first) |
| '2X' | Secret key, i.e., a key for enciphering keys ('X' is a reference) |
| '3X' | Private key ('X' is a reference) |
| '4X' | Password ('X' is a reference) |
| '80' to 'FF' | Proprietary |
| Any other value is reserved for future use. | |

### 6.3.2 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 fix the SM data objects 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 shall apply.

— The card shall fill each empty primitive basic SM data object.

— Each control reference template present in the response descriptor template shall be present in the response at the same place with the same control reference data objects for cryptographic mechanisms, files and keys.

- If the response descriptor template provides auxiliary data, then the respective data object shall be empty in the response.
- If an empty reference data object for auxiliary data is present in the response descriptor 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 data objects.

## 6.4 Security supports

This clause defines 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 data objects. The card may provide them as generic support to cryptographic mechanisms performed by an application. Applications may reference them for secure messaging and for security operations available in a cryptographic toolbox (see ISO/IEC 7816-8). This part of ISO/IEC 7816 defines neither some characteristics of the security support data elements, e.g., their lengths, nor the algorithms that alter their value.

**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 data exchange 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 security of data exchanges with data elements called progression values. Their values are increased at specific events throughout the life of the card. They are different each time the card is activated. They include a card session counter and a session identifier.

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

— A 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 has occurred; 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 data objects (tags '88', '92', '93', see Table 29) may be present in a control reference template. These tags can be used if the SE supports unambiguous use of these data elements.

— Within the interindustry template referenced by tag '7A', the context-specific class (first byte in the range '80' to 'BF') is reserved for security support data objects as listed in Table 32.

**Table 32 — Security support data objects**

| Tag | | Value |
|---|---|---|
| '7A' | | Interindustry template for nesting security support data objects with the following tags |
| | '80' | Card session counter |
| | '81' | 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) |

## 6.5 Security environments

This clause defines 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 available in a cryptographic toolbox (see ISO/IEC 7816-8). A SE consists of data elements stored in the card or resulting from some computation, to be processed by the specified algorithms. A SE may contain a mechanism to initialise non-persistent data to be used in the environment, e.g., a session key. A SE may provide directions for handling computation results, e.g., storage in the card. An interindustry SE template (tag '7B') may describe a SE.

**SE identifier —** A SE identifier (SEID) may reference any security environment, e.g., for secure messaging (see tags '94', '95' in Table 23) and for storing and restoring by a MANAGE SECURITY ENVIRONMENT command (see 8.5.10).

SEID consists of one byte.

— 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 explicitly used, the value '01' is reserved for the default SE, always available. This part of ISO/IEC 7816 does not define the content of the default SE; it may be empty.

— The value 'EF' is reserved for future use.

**SE sets** — Definitions of associated security environments may be grouped into sets.

— The card may provide one global SE set. The first SE of this set is the default SE.

— Applications may provide one or more application-specific SE set.

The global SE set shall be active by default, unless otherwise specified. A SE or a SE set may be associated with a file (DF or EF) such that after file selection the associated SE or a specific SE in the set is implicitly set. The specification of the association between a file and a SE set is outside the scope of this part of ISO/IEC 7816.

**Components —** Control reference templates (CRT, see 6.3.1) may describe various components of a SE. Any relative control reference (files, keys or data) specified with a mechanism in the environment definition shall be resolved with respect to the DF 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 current DF.

— Some components are transmitted in commands using secure messaging.

— Some components are transmitted in MANAGE SECURITY ENVIRONMENT commands.

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

The current SE is valid until there is a warm reset or a deactivation of the contacts (see ISO/IEC 7816-3), a change of context (e.g., by selecting a different application DF) or a MANAGE SECURITY ENVIRONMENT command. The MANAGE SECURITY ENVIRONMENT command may explicitly set or replace the current SE.

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

**Certificate holder authorization —** Public-key authentication procedures may use card-verifiable certificates. In such a certificate, a certificate holder authorization (e.g., a role identifier) may be contained coded in an interindustry data element referenced by tag '5F4C'. If such a certificate holder authorization is used in the security conditions to fulfil for accessing data or functions, then the data object (tag '5F4C') shall be present in the control reference template for authentication (AT) describing the public-key authentication procedure.

NOTE In the first edition of ISO/IEC 7816-9, tag '5F4B' references a certificate holder authorization (a 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 (a data element of one byte). Consequently, tag '5F4B' is now deprecated in ISO/IEC 7816.

**Access control —** The card may store security environments used for access control within EFs (see tag '8D' in Table 20) containing interindustry SE templates (tag '7B').

Within the interindustry template referenced by tag '7B', the context-specific class (first byte in the range '80' to 'BF') is reserved for security environment data objects. As listed in Table 33, the security environment template contains, for every included SE, a SEID data object (tag '80'), an optional LCS data object (tag '8A'), one or more optional cryptographic mechanism identifier template (tag 'AC') and one or more CRTs (tags 'A4', 'AA', 'B4', 'B6', 'B8').

**Table 33 — Security environment data objects**

| Tag | | Value |
|---|---|---|
| '7B' | | Interindustry template for nesting security environment data objects with the following tags |
| | '80'<br>'8A'<br>'AC'<br>'A4', 'AA', 'B4', B6', 'B8' | SEID byte, mandatory<br>LCS byte, optional<br>Cryptographic mechanism identifier template, optional<br>CRTs |

If present in the SE template, the LCS data object indicates for which life cycle state 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 data object is present, then the SE is valid for the activated operational state.

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

# 7 Security attributes

This clause specifies security attributes and the means to retrieve them from the card. The control parameters of any file may include security attributes (see tags '86', '8B', '8C', 'A0', 'A1', 'AB' in Table 20 in 5.6). Any object in the card (e.g., commands, files, data objects, tables & views) may be associated with a reference contained in a security attribute. Any object in the card may be associated with more than one security attribute.

A security attribute template for data objects (tag 'A0') may be present in the control parameters of any file. Such a template is the concatenation of a security attribute data object (tags '86', '8B', '8C', 'A0', 'A1', 'AB') and a tag list data object (tag '5C', see 9.4.1) indicating the relevant data objects in the file.

A security attribute template for physical interfaces (tag 'A1') may be present in the control parameters of any DF. Such a template specifies access conditions for contact interface and / or for radio frequency interface. It shall contain one or more pairs of data objects: each pair consists of a physical interface data object with tag '91' followed by a security attribute data object with tags '86', '8B', '8C', 'A0' or 'AB'. Table 34 shows a physical interface byte referenced by tag '91'.

**Table 34 — Physical interface byte**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Access by contacts according to ISO/IEC 7816-3 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | Access by radio frequency |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | Dual access |
| Any other value is reserved for future use. | | | | | | | | |

In SCQL environment (see ISO/IEC 7816-7, interindustry 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 part of ISO/IEC 7816 are used, then they shall be conveyed in a data object with tags '8B', '8C' or 'AB' in the security attribute parameters of an SCQL operation.

**Formats —** This part of ISO/IEC 7816 defines two formats for binding objects and security attributes.

— The compact format is based on bitmaps.

— The expanded format extends the compact format by TLV list management.

## 7.1 Compact format

In compact format, an access rule consists of an access mode byte followed by one or more security condition bytes. Access control to an object is managed by binding access rules to the related object. If several access rules are present in the value field of a data object with tag '8C' (see table 20), they represent an OR condition.

**Access mode bytes —** In an access mode byte, each bit 7 to 1 indicates either the absence of security condition byte when set to zero or the presence of a security condition byte in the same order (bits 7 to 1) when set to one. When bit 8 is set to one, bits 7 to 4 may be used for additional commands, e.g., application-specific commands.

Tables 35 to 38 define access mode bytes respectively for DFs, EFs, data objects and tables & views.

**Table 35 — Access mode byte for DFs**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 0 | - | - | - | - | - | - | - | Coding of bits 7 to 1 according to this table |
| 1 | - | - | - | - | - | - | - | Coding of bits 3 to 1 according to this table (bits 7 to 4 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 36 — Access mode byte for EFs**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 0 | - | - | - | - | - | - | - | Coding of bits 7 to 1 according to this table |
| 1 | - | - | - | - | - | - | - | Coding of bits 3 to 1 according to this table (bits 7 to 4 proprietary) |
| 0 | 1 | - | - | - | - | - | - | DELETE FILE |
| 0 | - | 1 | - | - | - | - | - | TERMINATE EF |
| 0 | - | - | 1 | - | - | - | - | ACTIVATE FILE |
| 0 | - | - | - | 1 | - | - | - | DEACTIVATE FILE |
| - | - | - | - | - | 1 | - | - | WRITE BINARY, WRITE RECORD, APPEND RECORD |
| - | - | - | - | - | - | 1 | - | UPDATE BINARY, UPDATE RECORD, ERASE BINARY |
| - | - | - | - | - | - | - | 1 | READ BINARY, READ RECORD, SEARCH BINARY, SEARCH RECORD |

**Table 37 — Access mode byte for data objects**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 0 | - | - | - | - | - | - | - | Coding of bits 7 to 1 according to this table |
| 1 | - | - | - | - | - | - | - | Coding of bits 3 to 1 according to this table (bits 7 to 4 proprietary) |
| 0 | x | x | x | x | - | - | - | 0000 (any other value is reserved for future use) |
| - | - | - | - | - | 1 | - | - | MANAGE SECURITY ENVIRONMENT |
| - | - | - | - | - | - | 1 | - | PUT DATA |
| - | - | - | - | - | - | - | 1 | GET DATA |

**Table 38 — Access mode byte for tables & views**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 0 | - | - | - | - | - | - | - | Coding of bits 7 to 1 according to this table |
| 1 | - | - | - | - | - | - | - | Coding of bits 3 to 1 according to this table (bits 7 to 4 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 |

**Security condition bytes —** Each security condition byte specifies which security mechanisms are necessary to conform to the access rule. Bits 8 to 5 fix the required security conditions. If not all equal, bits 4 to 1 code a SEID from one to fourteen and the mechanisms defined in the security environment shall be used according to the indications in bits 7 to 5 for command protection and / or external authentication and / or user authentication.

— If bit 8 is set to one, then all the conditions set in bits 7 to 5 shall be satisfied.

— If bit 8 is set to zero, then at least one of the conditions set in bits 7 to 5 shall be satisfied.

— If bit 7 is set to one, then the control reference template of the SE identified in bits 4 to 1, i.e., a SEID from one to fourteen, describes whether secure messaging shall apply to the command data field and / or to the response data field (see usage qualifier byte, Table 30).

Table 39 shows the security condition byte.

**Table 39 — Security condition byte**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | No condition |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | Never |
| - | - | - | - | 0 | 0 | 0 | 0 | No reference to a SE |
| - | - | - | - | Not all equal | | | | SEID from one to fourteen |
| - | - | - | - | 1 | 1 | 1 | 1 | Reserved for future use |
| 0 | - | - | - | - | - | - | - | At least one condition |
| 1 | - | - | - | - | - | - | - | All conditions |
| - | 1 | - | - | - | - | - | - | Secure messaging |
| - | - | 1 | - | - | - | - | - | External authentication |
| - | - | - | 1 | - | - | - | - | User authentication (e.g., password) |

## 7.2 Expanded format

In expanded format, an access rule consists of an access mode data object followed by one or more security condition data objects. Access control to an object is managed by referencing access rules from the related object. A template with tag 'AB' may be present in the control parameters of any file (see table 20) for such access rules.

**Access mode data objects —** An access mode data object contains either an access mode byte (see Tables 35 to 38) or a list of command descriptions or a proprietary state machine description; subsequent security condition data objects are relevant for all the indicated commands. Table 40 shows access mode data objects.

**Table 40 — Access mode data objects**

| Tag | Length | Value | Meaning |
|---|---|---|---|
| '80' | 1 | Access mode byte | According to Tables 35 to 38 |
| '81' to '8F' | Var. | Command header description | List of [part of] command headers according to Table 41 |
| '9C' | Var. | | Proprietary state machine description |

If the tag is in the range '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 4 to 1 of the tag, the list contains only values as described in table 41. 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 41 — Tags '81' to '8F' for access mode data objects**

| 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 indicate the basic logical channel (logical channel number set to zero) with the meaning that the description is independent from the particular logical channel in use.<br>The value of INS shall be even with the meaning that the description is independent from the indication of the data field format. | | | | | | | | |

**Security condition data objects —** According to Table 42, the security condition data objects define the security actions required for accessing an object protected through the particular access mode data object. When a control reference template with tag 'A4' (AT), 'B4' (CCT), 'B6' (DST) or 'B8' (CT) is used as a security condition, it shall contain a usage qualifier data object (see Table 30 in 6.3.1.1) for fixing the security action.

**Table 42 — Security condition data objects**

| Tag | Length | Value | Meaning |
|---|---|---|---|
| '90' | 0 | - | Always |
| '97' | 0 | - | Never |
| '9E' | 1 | Security condition byte | According to Table 39 |
| 'A4' | Var. | Control reference template | External or user authentication depending on the usage qualifier |
| 'B4', 'B6', 'B8' | Var. | Control reference template | SM in command and / or response depending on the usage qualifier |
| 'A0' | Var. | Security condition data objects | At least one security condition shall be fulfilled (OR template) |
| 'AF' | Var. | Security condition data objects | Every security condition shall be fulfilled (AND template) |

Several security condition data objects may be attached to the same operation.

— If security condition data objects are grouped in an OR template (tag 'A0'), then at least one security condition shall be fulfilled before the operation is allowed.

— If security condition data objects are not grouped in an OR template (tag 'A0') or if they are grouped in an AND template (tag 'AF'), then every security condition shall be fulfilled before the operation is allowed.

## 7.3 Access rule references

Access rules in expanded format may be stored in an EF of 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 43 illustrates the layout of an EF.ARR.

**Table 43 — EF.ARR layout**

| **Record number** (ARR byte) | **Record content** (one or more access rules) |
|---|---|
| 1 | Access mode data object, one or more security condition data objects, access mode data object, … |
| 2 | Access mode data object, one or more security condition data objects, … |

Table 44 shows security attribute data objects referencing expanded format. Referenced by tag '8B', they may be present in the control parameters of any application DF (see Table 20).

— 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 a SEID byte followed by an ARR byte; the SEID byte fixes the security environment where the access rules referenced by the ARR byte apply.

**Table 44 — Security attribute data objects referencing expanded format**

| Tag | Length | Value |
|---|---|---|
| '8B' | 1 | <ARR byte> |
| | 3 | <File identifier> - <ARR byte> |
| | Even, > 3 | <File identifier> - <SEID byte - ARR 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 SEID is set in a former MANAGE SECURITY ENVIRONMENT command, then the default SE is the current SE.

# 8 Interindustry commands for interchange

The subsequent six clauses specify interindustry commands for interchange, presented in six groups.

1) Selection
2) Data unit handling
3) Record handling
4) Data object handling
5) Basic security handling
6) Transmission handling

It shall not be mandatory for all cards complying with this part of ISO/IEC 7816 to support all those commands or all the options of a supported command. When international interchange is required, a set of application-independent card services and related commands and options shall be used as defined in clause 9.

For each command, a table provides a non-exhaustive list of status conditions (see also 5.3.5).

## 8.1 Selection

After the answer to reset, the MF is implicitly selected through the basic logical channel (see 5.3.7), unless specified differently in the historical bytes (see 5.5) or in the initial data string (see 9.2.1).

### 8.1.1 SELECT command

A successful SELECT command sets a current file within a logical channel (see 5.3.7). Subsequent commands may implicitly refer to the current file through that logical channel.

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

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

NOTE A direct selection by DF name allows the selection by application identifiers.

The following conditions shall apply to each open logical channel. Unless otherwise specified, the correct execution of the command modifies the security status according to the following rules.

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

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

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

**Table 45 — SELECT command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1-P2 | As defined in 5.3.1<br>'A4'<br>Selection control and options according to Tables 46 and 47 |
| Lc field | Number of bytes in the command data field or absent |
| Data field | File identifier or path from the MF or path from the current DF or DF name, according to P1-P2 or absent |
| Le field | Maximum number of bytes expected in the response data field or absent |

| | |
|---|---|
| Data field | File control information according to P2 or absent |
| SW1-SW2 | Status bytes according to Table 48 |

See 5.2.1 for file selection methods. See 5.5.2.7 and Table 16 for the file selection methods supported by the card.

— If parameter P1 is set to '00', then the card knows either because of a specific coding of the file identifier or because of the context of execution of the command whether the file to select is the MF, a DF or an EF.

  - If parameters P1-P2 are set to '0000' 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.
  - If parameters P1-P2 are set to '0000' and the command data field is absent or set to '3F00', then the MF shall be selected.

— If parameter P1 is set to '04', then the command data field is a DF name, 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.

**Table 46 — Selection control in P1**

| 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 | DF identifier |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | — Select EF under the current DF | EF identifier |
| 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 | — Direct selection by DF name | DF name |
| 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 |
| Any other value is reserved for future use. | | | | | | | | | |

**Table 47 — Selection options in P2**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | - | - | x | x | **File occurrence option** |
| 0 | 0 | 0 | 0 | - | - | 0 | 0 | — First or only occurrence |
| 0 | 0 | 0 | 0 | - | - | 0 | 1 | — Last occurrence |
| 0 | 0 | 0 | 0 | - | - | 1 | 0 | — Next occurrence |
| 0 | 0 | 0 | 0 | - | - | 1 | 1 | — Previous occurrence |
| 0 | 0 | 0 | 0 | x | x | - | - | **File control information option** (see 5.6 and Table 19) |
| 0 | 0 | 0 | 0 | 0 | 0 | - | - | — Return FCI, optional template |
| 0 | 0 | 0 | 0 | 0 | 1 | - | - | — Return FCP template |
| 0 | 0 | 0 | 0 | 1 | 0 | - | - | — Return FMD template |
| 0 | 0 | 0 | 0 | 1 | 1 | - | - | — Proprietary |
| Any other value is reserved for future use. | | | | | | | | |

If the Le field contains only bytes set to '00', then within the limit of 256 for a short Le field or 65 536 for an extended Le field, all the bytes corresponding to the selection option should be returned. If the Le field is absent, then the response data field shall also be absent; this is the standard way for returning no file control information.

**Table 48 — Status conditions in SW1-SW2**

| | SW1 | SW2 | Meaning |
|---|---|---|---|
| **Warning** | '62' | '83'<br>'84' | Selected file invalidated<br>File control information not formatted according to 5.6 |
| **Error** | '6A' | '81'<br>'82'<br>'86'<br>'87' | Function not supported<br>File not found<br>Incorrect parameters P1-P2<br>Lc inconsistent with parameters P1-P2 |

### 8.1.2 MANAGE CHANNEL command

A successful MANAGE CHANNEL command opens or closes logical channels (see 5.3.7).

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

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

After a successful open function performed from

— the basic logical channel (bits 6, 2 and 1 all set to zero in CLA, coding number zero), the MF shall be implicitly selected as the current DF and the security status for the new logical channel should be the same as for the basic logical channel after the answer to reset. The security status of the new logical channel should be separate from that of any other logical channel.

— a non-basic logical channel (bits 6, 2 and 1 not all set to zero in CLA, coding a number from one to seven), the current DF of the logical channel from which the command was issued shall be selected as the current DF and the security status for the new logical channel should be the same as for the logical channel from which the open function was performed.

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

**Table 49 — MANAGE CHANNEL command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1-P2 | As defined in 5.3.1<br>'70'<br>'0000' to '0007' for opening a logical channel<br>'8000' for closing a logical channel (any other value is reserved for future use) |
| Lc field | Absent |
| Data field | Absent |
| Le field | '01' if parameters P1-P2 are set to '0000', or absent otherwise |

| | |
|---|---|
| Data field | Logical channel number on bits 3, 2 and 1 if parameters P1-P2 are set to '0000', or absent otherwise |
| SW1-SW2 | Status bytes according to Table 50 |

— If parameter P1 is set to '00', i.e., bit 8 set to zero in P1, then MANAGE CHANNEL shall open a logical channel numbered as follows.

- If parameters P1-P2 are set to '0001' to '0007', then bits 3 to 1 not all set to zero in P2 shall code an externally assigned number from one to seven.
- If parameters P1-P2 are set to '0000', then the Le field shall be one byte set to '01'. The response data field shall be a single byte from '01' to '07' where bits 3, 2 and 1 not all set to zero shall code a number from one to seven assigned by the card; the other values are invalid.

— If parameters P1-P2 are set to '8000', i.e., bit 8 set to one in P1, then MANAGE CHANNEL shall close the logical channel (bits 6 to 1 not all set to zero in CLA, coding a number from one to seven).

**Table 50 — Status conditions in SW1-SW2**

| | SW1 | SW2 | Meaning |
|---|---|---|---|
| **Warning** | '62' | '00' | No information given |

## 8.2 Data unit handling

Such a command shall be aborted if applied to an EF without transparent structure. It can be performed only if the security status satisfies the security attributes defined for this EF for the function, namely, read, write, update, erase or search.

**Parameters P1-P2 —** The commands of this group shall use parameters P1-P2 for coding an offset and optionally a short EF identifier as follows.

— If bit 8 is set to zero in parameter P1, then parameters P1-P2 code the offset from zero to 32767 (fifteen bits).

— If bit 8 is set to one in parameter P1, then bits 7 and 6 of P1 are set to 00 (RFU bits), bits 5 to 1 of P1 code a short EF identifier and bits 8 to 1 of parameter P2 code the offset from zero to 255 (eight bits).

If the short EF identifier is valid, then the command sets this EF as current EF. If there is a current EF at the time of issuing the command, then the command may be processed without indication of the EF.

The offset indicates the first byte to be read, written, updated, erased or searched. From zero for the first data unit at the beginning of the EF, the offset is incremented by one for each subsequent data unit.

### 8.2.1 READ BINARY command

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

**Table 51 — READ BINARY command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1-P2 | As defined in 5.3.1<br>'B0'<br>According to 8.2 |
| Lc field | Absent |
| Data field | Absent |
| Le field | Number of bytes to be read |

| | |
|---|---|
| Data field | Data read (Le bytes) |
| SW1-SW2 | Status bytes according to Table 52 |

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

**Table 52 — Status conditions in SW1-SW2**

| | SW1 | SW2 | Meaning |
|---|---|---|---|
| **Warning** | '62' | '81'<br>'82' | Part of returned data may be corrupted<br>End of file reached before reading Le bytes |
| **Error** | '67' | '00' | Wrong length (wrong Le field) |
| | '69' | '81'<br>'82'<br>'86' | Command incompatible with file structure<br>Security status not satisfied<br>Command not allowed (no current EF) |
| | '6A' | '81'<br>'82' | Function not supported<br>File not found |
| | '6B' | '00' | Wrong parameters (offset outside the EF) |
| | '6C' | 'XX' | Wrong length (wrong Le field; 'XX' fixes the exact length) |

### 8.2.2 WRITE BINARY command

The WRITE BINARY command initiates one of the following operations into an EF according to the file attributes:

— the one-time write in the card of the bits given in the command data field;

— 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);

— 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., if the data coding byte is absent in the historical bytes (see Table 17 in 5.5), in EF.ATR and in all the control parameters (see Table 20 in 5.6) from the MF to an EF, then the logical-OR shall apply for that EF.

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

**Table 53 — WRITE BINARY command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1-P2 | As defined in 5.3.1<br>'D0'<br>According to 8.2 |
| Lc field | Number of bytes in the command data field |
| Data field | String of data units to be written |
| Le field | Absent |

| | |
|---|---|
| Data field | Absent |
| SW1-SW2 | Status bytes according to Table 54 |

**Table 54 — Status conditions in SW1-SW2**

| | SW1 | SW2 | Meaning |
|---|---|---|---|
| **Warning** | '63' | 'CX' | Counter (successful writing, but after using an internal retry routine, 'X' > '0' tells the number of retries; 'X' = '0' means that no counter is provided) |
| **Error** | '65' | '81' | Memory failure (unsuccessful writing) |
| | '67' | '00' | Wrong length (wrong Lc field) |
| | '69' | '81'<br>'82'<br>'86' | Command incompatible with file structure<br>Security status not satisfied<br>Command not allowed (no current EF) |
| | '6A' | '81'<br>'82' | Function not supported<br>File not found |
| | '6B' | '00' | Wrong parameters (offset outside the EF) |

### 8.2.3 UPDATE BINARY command

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

**Table 55 — UPDATE BINARY command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1-P2 | As defined in 5.3.1<br>'D6'<br>According to 8.2 |
| Lc field | Number of bytes in the command data field |
| Data field | String of data units to be updated |
| Le field | Absent |

| | |
|---|---|
| Data field | Absent |
| SW1-SW2 | Status bytes according to Table 54 |

### 8.2.4 ERASE BINARY command

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

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

**Table 56 — ERASE BINARY command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1-P2 | As defined in 5.3.1<br>'0E'<br>According to 8.2 |
| Lc field | '02' or absent |
| Data field | Offset of the first data unit not to be erased or absent |
| Le field | Absent |

| | |
|---|---|
| Data field | Absent |
| SW1-SW2 | Status bytes according to Table 54 |

### 8.2.5 SEARCH BINARY command

The SEARCH BINARY command initiates a search within an EF of transparent structure. 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.

**Table 57 — SEARCH BINARY command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1-P2 | As defined in 5.3.1<br>'A0'<br>According to 8.2 |
| Lc field | Number of bytes in the command data field or absent |
| Data field | Search string or absent |
| Le field | '02' or absent |

| | |
|---|---|
| Data field | Offset of the first data unit matching the command data field or absent |
| SW1-SW2 | Status bytes according to Table 58 |

The response data field is absent either because the Le field is absent or because no match is found.

If the command data field is absent, then the response data field gives the offset of the first data unit in a logically erased state.

**Table 58 — Status conditions in SW1-SW2**

| | SW1 | SW2 | Meaning |
|---|---|---|---|
| **Warning** | '62' | '82' | End of file found before finding matching string |
| **Error** | '69' | '82' | Security status not satisfied |

## 8.3 Record handling

Such a command shall be aborted if applied to an EF without record structure. It can be performed only if the security status satisfies the security attributes defined for this EF for the function, namely, read, write, append, update or search.

**Parameter P2 —** The commands of this group shall reserve or use bits 8 to 4 in parameter P2 for coding a short EF identifier according to Table 59.

**Table 59 — 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 from one to thirty |
| 1 | 1 | 1 | 1 | 1 | - | - | - | Reserved for future use |

If the short EF identifier is valid, then the command sets this EF as current EF and resets the record pointer. If there is a current EF at the time of issuing such a command, then the command may be processed without indication of the EF.

### 8.3.1 READ RECORD (S) command

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

**Table 60 — READ RECORD (S) command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1<br>P2 | As defined in 5.3.1<br>'B2'<br>Record number or record identifier ('00' indicates the current record)<br>Reference control according to 8.3 and Table 61 |
| Lc field | Absent |
| Data field | Absent |
| Le field | Number of bytes to be read |

| | |
|---|---|
| Data field | Lr (may be equal to Le) bytes |
| SW1-SW2 | Status bytes according to Table 63 |

If the Le field contains only bytes set to '00', then depending on bits 3, 2 and 1 of parameter P2 and within the limit of 256 for a short Le field or 65 536 for an extended Le field, the command should read completely either the single requested record or the requested sequence of records.

**Table 61 — Reference control in P2**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| x | x | x | x | x | - | - | - | Reserved for short EF identifier according to Table 59 |
| - | - | - | - | - | 0 | x | x | **Record identifier in parameter 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 parameter P1** |
| - | - | - | - | - | 1 | 0 | 0 | — Read record P1 |
| - | - | - | - | - | 1 | 0 | 1 | — Read all records from P1 up to the last |
| - | - | - | - | - | 1 | 1 | 0 | — Read all records from the last up to P1 |
| | | | | | 1 | 1 | 1 | Reserved for future use |

If the records are SIMPLE-TLV data objects (see 5.4.2), then Table 62 illustrates the response data field. The comparison of the length of the data field with its TLV structure gives the nature of the data: the unique record (read one record) or the last record (read all records) is incomplete, complete or padded.

**Table 62-1 — Response data field when reading for one record**

Case a — Partial read of one record (the Le field does not contain only bytes set to '00')

| $T_n$ (one byte) | $L_n$ (one or three bytes) | First bytes of $V_n$ |
|---|---|---|

<---------- Le bytes ---------->

Case b — Complete read of one record (the Le field contains only bytes set to '00')

| $T_n$ (one byte) | $L_n$ (one or three bytes) | All the bytes of $V_n$ ($L_n$ bytes) |
|---|---|---|

**Table 62-2 — Response data field when reading for several records**

Case c — Partial read of a sequence of records (the Le 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) |
|---|---|---|

<---------- Le bytes ---------->

Case d — Read several records up to the file end (the Le field contains only bytes set to '00')

| $T_n$ - $L_n$ - $V_n$ | … | $T_{n+m}$ - $L_{n+m}$ - $V_{n+m}$ |
|---|---|---|

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

**Table 63 — Status conditions in SW1-SW2**

| | SW1 | SW2 | Meaning |
|---|---|---|---|
| **Warning** | '62' | '81'<br>'82' | Part of returned data may be corrupted<br>End of record reached before Le bytes |
| **Error** | '67' | '00' | Wrong length (no Le field) |
| | '69' | '81'<br>'82' | Command incompatible with file structure<br>Security status not satisfied |
| | '6A' | '81'<br>'82'<br>'83' | Function not supported<br>File not found<br>Record not found |
| | '6C' | 'XX' | Wrong length (wrong Le field; 'XX' fixes the exact length) |

### 8.3.2 WRITE RECORD command

The WRITE RECORD command initiates one of the following operations within an EF:

— the write-once of a record given in the command data field;

— 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;

— 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., if the data coding byte is absent in the historical bytes (see Table 17 in 5.5), in EF.ATR and in all the control parameters (see Table 20 in 5.6) from the MF to an EF, then the logical-OR shall apply for that EF.

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

If applied to an EF of cyclic structure with records of fixed size, the "previous" option (bits 3, 2 and 1 set to 011 in parameter P2) behaves as APPEND RECORD.

**Table 64 — WRITE RECORD command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1<br>P2 | As defined in 5.3.1<br>'D2'<br>Record number ('00' indicates the current record)<br>According to 8.3 and Table 65 |
| Lc field | Number of bytes in the command data field |
| Data field | Record to be written |
| Le field | Absent |

| | |
|---|---|
| Data field | Absent |
| SW1-SW2 | Status bytes according to Table 66 |

**Table 65 — Reference control in P2**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| x | x | x | x | x | - | - | - | Reserved for short EF identifier according to Table 59 |
| - | - | - | - | - | 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 parameter P1** |
| Any other value is reserved for future use. | | | | | | | | |

If the records are SIMPLE-TLV data objects (see 5.4.2), then Table 66 illustrates the command data field.

**Table 66 — Command data field (one complete record)**

| | | |
|---|---|---|
| $T_n$ (one byte) | $L_n$ (one or three bytes) | All the bytes of $V_n$ ($L_n$ bytes) |

**Table 67 — Status conditions in SW1-SW2**

| | SW1 | SW2 | Meaning |
|---|---|---|---|
| **Warning** | '63' | 'CX' | Counter (successful writing, but after using an internal retry routine, 'X' > '0' tells the number of retries; 'X' = '0' means that no counter is provided) |
| **Error** | '65' | '81' | Memory failure (unsuccessful writing) |
| | '67' | '00' | Wrong length (no Lc field) |
| | '69' | '81'<br>'82'<br>'86' | Command incompatible with file structure<br>Security status not satisfied<br>Command not allowed (no current EF) |
| | '6A' | '81'<br>'82'<br>'83'<br>'84'<br>'85' | Function not supported<br>File not found<br>Record not found<br>Not enough memory space in the file<br>Lc inconsistent with TLV structure |

### 8.3.3 UPDATE RECORD command

The UPDATE RECORD command initiates the updating of a specific record with the bits given in the command data field. The command shall set the record pointer on the successfully updated record.

— If applied to an EF of linear or cyclic structure with records of fixed size, then the command shall be aborted if the record size is different from the size of the existing record.

— If applied to an EF of linear structure with records of variable size, then the command may be carried out when the record size is different from the size of the existing record.

— If applied to an EF of cyclic structure with records of fixed size, the "previous" option (bits 3, 2 and 1 set to 011 in parameter P2) behaves as APPEND RECORD.

**Table 68 — UPDATE RECORD command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1<br>P2 | As defined in 5.3.1<br>'DC'<br>Record number ('00' indicates the current record)<br>According to 8.3 and Table 65 |
| Lc field | Number of bytes in the command data field |
| Data field | Record to be updated |
| Le field | Absent |

| | |
|---|---|
| Data field | Absent |
| SW1-SW2 | Status bytes according to Table 67 |

If the records are SIMPLE-TLV data objects (see 5.4.2), then Table 66 illustrates the command data field.

### 8.3.4 APPEND RECORD command

The APPEND RECORD command initiates either the writing of a new record at the end of an EF of linear structure or the writing of record number one in an EF of cyclic structure.

If this command applies to an EF of linear structure full of records, then the command is rejected because there is not enough memory space in the file.

If this command applies to an EF of cyclic structure full of records, then the record with the highest record number is replaced. This record becomes record number one.

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

**Table 69 — APPEND RECORD command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1<br>P2 | As defined in 5.3.1<br>'E2'<br>'00' (any other value is invalid)<br>According to 8.3 and Table 59 (any other value is reserved for future use) |
| Lc field | Number of bytes in the command data field |
| Data field | Record to be appended |
| Le field | Absent |

| | |
|---|---|
| Data field | Absent |
| SW1-SW2 | Status bytes according to Table 67 |

If the records are SIMPLE-TLV data objects (see 5.4.2), then Table 66 illustrates the command data field.

### 8.3.5 SEARCH RECORD command

The SEARCH RECORD command initiates a simple search, an enhanced search or a 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 SEARCH RECORD response data field gives the numbers of records matching the search criteria within an EF of record structure. The command shall set the record pointer on the first record matching the search criteria.

NOTE The response data field does not give record identifiers because they may not be unique.

In an EF of linear structure with records of variable size, the search shall not take into account the records shorter than the search string. In an EF of linear or cyclic structure with records of fixed size, if the search string is longer than the records, then the card shall abort the command with an appropriate error code.

**Table 70 — SEARCH RECORD command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1<br>P2 | As defined in 5.3.1<br>'A2'<br>Record number or record identifier ('00' indicates the current record)<br>Reference control according to 8.3 and Table 71 |
| Lc field | Number of bytes in the command data field |
| Data field | Search string if bits 3 and 2 not set to 11 in parameter P2 (simple search), or<br>Search indication (2 bytes) followed by search string if bits 3, 2 and 1 set to 110 in P2 (enhanced search), or<br>Proprietary coding if bits 3, 2 and 1 set to 111 in parameter P2 (proprietary search) |
| Le field | Maximum number of bytes expected in the response data field or absent |

| | |
|---|---|
| Data field | Record numbers or absent |
| SW1-SW2 | Status bytes according to Table 73 |

**Table 71 — Reference control in P2**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| x | x | x | x | x | - | - | - | Reserved for short EF identifier, see Table 59 |
| - | - | - | - | - | 0 | x | x | **Simple search with record identifier in parameter 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 parameter P1** |
| - | - | - | - | - | 1 | 0 | 0 | — Forward from P1 |
| - | - | - | - | - | 1 | 0 | 1 | — Backward from P1 |
| - | - | - | - | - | 1 | 1 | 0 | **Enhanced search** (see Table 72) |
| - | - | - | - | - | 1 | 1 | 1 | **Proprietary search** |

In an enhanced search (bits 3, 2 and 1 set to 110 in parameter P2), the command data field consists of a search indication on two bytes followed by a search string. Table 72 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 (absolute position) or after the first occurrence of this value.

NOTE The response data field is absent either because the Le field is absent or because no match is found.

**Table 72 — 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 parameter 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 parameter 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 reserved for future use. | | | | | | | | |

**Table 73 — Status conditions in SW1-SW2**

| | SW1 | SW2 | Meaning |
|---|---|---|---|
| **Warning** | '62' | '82' | End of file found before finding matching string |
| **Error** | '69' | '82' | Security status not satisfied |
| | '6C' | 'XX' | Wrong length (wrong Le field; 'XX' fixes the exact length) |

## 8.4 Data object handling

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

**Parameters P1-P2 —** The commands of this group shall use parameters P1-P2 for referencing data, either as application data or as data objects. Table 74 shows the data references in parameters P1-P2.

— If parameter P1 is set to '00', then parameter P2 from '40' to 'FE' shall be a BER-TLV tag on a single byte. The value '00FF' is reserved either for obtaining all the common BER-TLV data objects readable in the context or for indicating that the command data field is BER-TLV coded.

— If parameter P1 is set to '01', then parameter P2 from '00' to 'FF' shall be an identifier reserved for card internal tests and for proprietary services meaningful within a given application context.

— If parameter P1 is set to '02', then parameter P2 from '01' to 'FE' shall be a SIMPLE-TLV tag. The value '0200' is reserved for future use. The value '02FF' is reserved either for obtaining all the common SIMPLE-TLV data objects readable in the context or for stating that the command data field is SIMPLE-TLV coded.

— If parameter P1 is set to '03', then parameter P2 set to '4X' or '5X' (BER-TLV) or to 'CX' or 'DX' (SIMPLE-TLV) shall code a short EF identifier on bits 5 to 1 for accessing a working EF of TLV structure; the other values of P2 are reserved for future use. If the short EF identifier is valid, then the command sets this EF as current EF. If there is a current EF at the time of issuing such a command, then the command may be processed without indication of the EF.

— If parameters P1-P2 lie from '4000' to 'FFFF', then they shall be a BER-TLV tag on two bytes. The values that are not valid BER-TLV tags on two bytes (see 5.4.1.1) are reserved for future use, e.g., '4000' and 'FFXX'.

**Table 74 — Data references in P1-P2**

| Value | Meaning |
|---|---|
| '0000' | Reserved for card-originated command-response pairs (see 9.1) |
| '0001' to '003F' | Reserved for future use |
| '0040' to '00FF' | BER-TLV tag (one byte) in parameter P2 |
| '0100' to '01FF' | Application data (proprietary coding) |
| '0200' to '02FF' | SIMPLE-TLV tag in parameter P2 |
| '0300' to '03FF' | Short EF identifier in parameter P2 set to '4X' or '5X' (BER-TLV) or 'CX' or 'DX' (SIMPLE-TLV) |
| '0400' to '3FFF' | Reserved for future use |
| '4000' to 'FFFF' | BER-TLV tag (two bytes) in parameters P1-P2 |

**Data fields —** The commands of this group shall use the data fields as follows.

— If a data object is requested or provided within the current context (e.g., application-specific environment or current DF), then the data field 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 BER-TLV data object.

— If the content of a working EF is requested or provided, then the data field shall contain the data object(s).

### 8.4.1 GET DATA command

The GET DATA command retrieves either the content of a working EF of TLV structure or one data object, possibly constructed, within the current context (e.g., application-specific environment or current DF).

**Table 75 — GET DATA command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1-P2 | As defined in 5.3.1<br>'CA'<br>According to 8.4 and Table 74 |
| Lc field | Absent |
| Data field | Absent |
| Le field | Maximum number of bytes expected in the response data field |

| | |
|---|---|
| Data field | Lr (may be equal to Le) bytes according to parameters P1-P2 |
| SW1-SW2 | Status bytes according to Table 76 |

If the Le field contains only bytes set to '00', then within the limit of 256 for a short Le field or 65 536 for an extended Le field, all the required information should be returned.

**Table 76 — Status conditions in SW1-SW2**

| | SW1 | SW2 | Meaning |
|---|---|---|---|
| **Warning** | '62' | '81' | Part of returned data may be corrupted |
| **Error** | '67' | '00' | Wrong length (no Le field) |
| | '69' | '81'<br>'82'<br>'85' | Command incompatible with file structure<br>Security status not satisfied<br>Conditions of use not satisfied |
| | '6A' | '81'<br>'88' | Function not supported<br>Referenced data (data objects) not found |
| | '6C' | 'XX' | Wrong length (wrong Le field; 'XX' tells the exact length) |

### 8.4.2 PUT DATA command

The PUT DATA command initiates the management of either the content of a working EF of TLV structure or one data object, possibly constructed, within the current context (e.g., application-specific environment or current DF).

The definition or the nature of the data objects shall induce the exact management functions, e.g., writing once and / or updating and / or appending.

**Table 77 — PUT DATA command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1-P2 | As defined in 5.3.1<br>'DA'<br>According to 8.4 and Table 74 |
| Lc field | Number of bytes in the command data field |
| Data field | Lc bytes according to parameters P1-P2 |
| Le field | Absent |

| | |
|---|---|
| Data field | Absent |
| SW1-SW2 | Status bytes according to Table 78 |

**Table 78 — Status conditions in SW1-SW2**

| | SW1 | SW2 | Meaning |
|---|---|---|---|
| **Warning** | '63' | 'CX' | Counter (successful storing, but after using an internal retry routine, 'X' > '0' tells the number of retries; 'X' = '0' means that no counter is provided) |
| **Error** | '65' | '81' | Memory failure (unsuccessful writing) |
| | '67' | '00' | Wrong length (wrong Lc field) |
| | '69' | '81'<br>'82'<br>'85' | Command incompatible with file structure<br>Security status not satisfied<br>Conditions of use not satisfied |
| | '6A' | '80'<br>'81'<br>'84'<br>'85' | Incorrect parameters in the command data field<br>Function not supported<br>Not enough memory space in the file<br>Lc inconsistent with TLV structure |

## 8.5 Basic security handling

**Parameter P1 —** Unless differently specified, the commands of this group shall reserve parameter P1 for referencing the algorithm to use: either a cryptographic algorithm or a biometric algorithm (see ISO/IEC 7816-11). Parameter P1 set to '00' means that no information is given; then either the reference is known before issuing the command or the command data field provides it.

**Parameter P2 —** Unless differently specified, the commands of this group shall reserve parameter P2 for qualifying reference data according to Table 79. Parameter P2 set to '00' means that no qualifier is given; then either the qualifier is known before issuing the command or the command data field provides it.

NOTE 1 The qualifier may be for example a password number or a key number or a short EF identifier.

NOTE 2 A MANAGE SECURITY ENVIRONMENT command may set an algorithm reference and / or a reference data qualifier.

**Table 79 — Reference data qualifier in P2**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | No information is given |
| 0<br>1 | -<br>- | -<br>- | -<br>- | -<br>- | -<br>- | -<br>- | -<br>- | Global reference data (e.g., MF specific password or key)<br>Specific reference data (e.g., DF specific password or key) |
| - | x | x | - | - | - | - | - | 00 (any other value is reserved for future use) |
| - | - | - | x | x | x | x | x | Reference data number or number of the secret |

Table 80 provides specific warning and error conditions for the basic security handling commands.

**Table 80 — Status conditions in SW1-SW2**

| | SW1 | SW2 | Meaning |
|---|---|---|---|
| **Warning** | '63' | '00'<br>'CX' | No information given (verification failed)<br>Counter (verification failed; 'X' tells the number of further allowed retries) |
| **Error** | '65' | '81' | Memory failure (unsuccessful change) |
| | '67' | '00' | Wrong length (no Lc field) |
| | '69' | '82'<br>'83'<br>'84' | Security status not satisfied<br>Authentication method blocked<br>Referenced data invalidated |
| | '6A' | '81'<br>'82'<br>'86'<br>'88' | Function not supported<br>File not found<br>Incorrect parameters P1-P2<br>Referenced data not found |

### 8.5.1 INTERNAL AUTHENTICATE command

The INTERNAL AUTHENTICATE 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.

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

If there is a current key and a current algorithm when issuing the command, then the command may implicitly use them.

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.

**Table 81 — INTERNAL AUTHENTICATE command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1-P2 | As defined in 5.3.1<br>'88'<br>According to 8.5 and Table 79 |
| Lc field | Number of bytes in the command data field |
| Data field | Authentication-related data (e.g., challenge) |
| Le field | Maximum number of bytes expected in the response data field |

| | |
|---|---|
| Data field | Authentication-related data (e.g., response to a challenge) |
| SW1-SW2 | Status bytes according to Table 80 |

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

### 8.5.2 GET CHALLENGE command

The GET CHALLENGE 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. No further condition is specified in this part of ISO/IEC 7816.

**Table 82 — GET CHALLENGE command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1<br>P2 | As defined in 5.3.1<br>'84'<br>According to 8.5<br>'00' (any other value is reserved for future use) |
| Lc field | Absent |
| Data field | Absent |
| Le field | Maximum number of bytes expected in the response data field |

| | |
|---|---|
| Data field | Challenge |
| SW1-SW2 | Status bytes according to Table 80 |

### 8.5.3 EXTERNAL AUTHENTICATE command

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

The successful execution of the command requires that the last challenge obtained from the card is valid. The card may record unsuccessful comparisons (e.g., to limit the number of further uses of the reference data).

**Table 83 — EXTERNAL AUTHENTICATE command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1-P2 | As defined in 5.3.1<br>'82'<br>According to 8.5 and Table 79 |
| Lc field | Number of bytes in the command data field or absent |
| Data field | Authentication-related data (e.g., response to a challenge) or absent |
| Le field | Absent |

| | |
|---|---|
| Data field | Absent |
| SW1-SW2 | Status bytes according to Table 80 |

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 ISO/IEC 9798-2 and 3.

The operation can be performed only if the security status satisfies the security attributes for this operation.

**Table 84 — Command-response pair for MUTUAL AUTHENTICATE function**

| | |
|---|---|
| CLA<br>INS<br>P1-P2 | As defined in 5.3.1<br>'82'<br>According to 8.5 and Table 79 |
| Lc field | Number of bytes in the command data field |
| Data field | Authentication-related data |
| Le field | Maximum number of bytes expected in the response data field |

| | |
|---|---|
| Data field | Authentication-related data |
| SW1-SW2 | Status bytes according to Table 80 |

### 8.5.4 GENERAL AUTHENTICATE command

The GENERAL AUTHENTICATE 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). As a matter of fact, the EXTERNAL and INTERNAL AUTHENTICATE commands preclude authentication protocols involving more than a challenge / response pair. For example, some protocols based on zero-knowledge techniques involve a commitment / challenge / response triple (see ISO/IEC 9798-5). Such a zero-knowledge triple requires two or more GENERAL AUTHENTICATE command-response pairs. Such commands shall be chained (bit 5 set to one in CLA, see 5.3.1).

The function (either INTERNAL, or EXTERNAL, or MUTUAL AUTHENTICATE) can be performed only if the security status satisfies the security attributes for this operation. The successful execution of a chain may be subject to successful completion of prior commands (e.g., VERIFY, SELECT) or selections (e.g., the relevant secret). If there is a current key and a current algorithm when issuing a chain, then the chain may implicitly use them. Such a chain may conditionally update the security status using the result (yes or no) of a control performed by the card.

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. The card may record unsuccessful comparisons (e.g., to limit the number of further uses of the reference data).

**Table 85 — GENERAL AUTHENTICATE command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1-P2 | As defined in 5.3.1<br>'86'<br>According to 8.5 and Table 79 |
| Lc field | Number of bytes in the command data field |
| Data field | Authentication-related data |
| Le field | Maximum number of bytes expected in the response data field, or<br>Absent in the last command of a chain for EXTERNAL AUTHENTICATE function |

| | |
|---|---|
| Data field | Authentication-related data, or<br>Absent either according to Le or if the card rejects the command |
| SW1-SW2 | Status bytes according to Table 80 |

If INS is set to '87' (odd INS code, see 5.3.2), then each data field shall contain an interindustry template referenced by tag '60'. In the dynamic authentication template, the context-specific class (first byte in the range '80' to 'BF') is reserved for dynamic authentication data objects as listed in Table 86.

**Table 86 — Dynamic authentication data objects**

| Tag | Meaning |
|---|---|
| '60' | Interindustry template for nesting dynamic authentication data objects with the following tags |
| '80'<br>'81'<br>'82'<br>'83'<br>'84'<br>'85' | Commitment (e.g., a positive number less than the public RSA modulus in use)<br>Challenge (e.g., a number, possibly zero, less than the public RSA exponent in use)<br>Response (e.g., a positive number less than the public RSA modulus in use)<br>Committed challenge (e.g., the hash-code of a commitment data object)<br>Authentication code (e.g., the hash-code of one or more data fields and a commitment data object)<br>Exponential (e.g., a public positive number for establishing a session key by a DH method) |
| 'A0' | Identification data template |

The following rules shall apply within the interindustry dynamic authentication templates, tag '60'.

— If an empty data object is present in a template, then it shall be complete in the template in the next data field.

— In the first data field of a chain, the template indicates the dynamic authentication function as follows.

- A commitment request, e.g., an empty commitment, denotes an INTERNAL AUTHENTICATE function.
- A challenge request, e.g., an empty challenge, denotes an EXTERNAL AUTHENTICATE function.
- The absence of empty data object denotes a MUTUAL AUTHENTICATE function. Then unless the card stops the chain for rejecting the function, for every command-response pair in the chain, the template in the response data field shall contain the same data objects as the template in the command data field.

The dynamic authentication may protect data fields exchanged during a session. Both entities maintain a current hash-code, updated by integrating one data field at a time. The data element with tag '84' is an authentication code; it results from updating the current hash-code by integrating a commitment data object with tag '80'. The verifier successively reconstructs a commitment and an authentication code: if the reconstructed commitment is not zero and if the reconstructed code is identical to the received code, then the authentication is successful.

The MUTUAL AUTHENTICATE function allows establishing a session key between two entities using a pair of data elements with tag '85' as "exponentials" in a key establishment protocol (see ISO/IEC 11170-3).

Annex D illustrates chains of GENERAL AUTHENTICATE commands for implementing INTERNAL, EXTERNAL and MUTUAL AUTHENTICATE functions, with extensions to data field authentication and key establishment.

### 8.5.5 VERIFY command

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

**Table 87 — VERIFY command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1<br>P2 | As defined in 5.3.1<br>'20'<br>'00' (any other value is reserved for future use)<br>According to 8.5 and Table 79 |
| Lc field | Number of bytes in the command data field or absent |
| Data field | Verification data or absent |
| Le field | Absent |

| | |
|---|---|
| Data field | Absent |
| SW1-SW2 | Status bytes according to Table 80 |

### 8.5.6 CHANGE REFERENCE DATA command

The CHANGE REFERENCE DATA 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 88 — CHANGE REFERENCE DATA command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1<br>P2 | As defined in 5.3.1<br>'24'<br>'00' or '01' (any other value is reserved for future use)<br>According to 8.5 and Table 79 |
| Lc field | Number of bytes in the command data field |
| Data field | Verification data followed by new reference data if parameter P1 is set to '00', or<br>New reference data if parameter P1 is set to '01' |
| Le field | Absent |

| | |
|---|---|
| Data field | Absent |
| SW1-SW2 | Status bytes according to Table 80 |

The card knows the length of the stored reference data; therefore no delimitation is present when P1 is set to '00'.

### 8.5.7 ENABLE VERIFICATION REQUIREMENT command

The ENABLE VERIFICATION REQUIREMENT 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 89 — ENABLE VERIFICATION REQUIREMENT command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1<br>P2 | As defined in 5.3.1<br>'28'<br>'00' or '01' (any other value is reserved for future use)<br>According to 8.5 and Table 79 |
| Lc field | Number of bytes in the command data field or absent |
| Data field | Verification data if parameter P1 is set to '00' or absent if parameter P1 is set to '01' |
| Le field | Absent |

| | |
|---|---|
| Data field | Absent |
| SW1-SW2 | Status bytes according to Table 80 |

### 8.5.8 DISABLE VERIFICATION REQUIREMENT command

The DISABLE VERIFICATION REQUIREMENT command switches off 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 90 — DISABLE VERIFICATION REQUIREMENT command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1<br>P2 | As defined in 5.3.1<br>'26'<br>'00' or '01' (any other value is reserved for future use)<br>According to 8.5 and Table 79 |
| Lc field | Number of bytes in the command data field or absent |
| Data field | Verification data if parameter P1 is set to '00' or absent if parameter P1 is set to '01' |
| Le field | Absent |

| | |
|---|---|
| Data field | Absent |
| SW1-SW2 | Status bytes according to Table 80 |

### 8.5.9 RESET RETRY COUNTER command

The RESET RETRY COUNTER 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 91 — RESET RETRY COUNTER command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1<br>P2 | As defined in 5.3.1<br>'2C'<br>'00', '01', '02' or '03' (any other value is reserved for future use)<br>According to 8.5 and Table 79 |
| Lc field | Number of bytes in the command data field or absent |
| Data field | Resetting code followed by new reference data if parameter P1 is set to '00', or<br>Resetting code if parameter P1 is set to '01', or<br>New reference data if parameter P1 is set to '02', or<br>Absent if parameter P1 is set to '03' |
| Le field | Absent |

| | |
|---|---|
| Data field | Absent |
| SW1-SW2 | Status bytes according to Table 80 |

The card knows the length of the stored resetting code; therefore no delimitation is present when P1 is set to '00'.

### 8.5.10 MANAGE SECURITY ENVIRONMENT command

The MANAGE SECURITY ENVIRONMENT 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 coded in P2;

— RESTORE, i.e., replacing the current SE by a SE stored in the card and identified by the SEID coded in P2;

— ERASE, i.e., erasing a SE stored in the card and identified by the SEID coded in P2.

Moreover, it initialises security operations for secure messaging (see 6) and for security commands (e.g., EXTERNAL, INTERNAL and GENERAL AUTHENTICATE, see also PERFORM SECURITY OPERATION in ISO/IEC 7816-8).

**Table 92 — MANAGE SECURITY ENVIRONMENT command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1<br>P2 | As defined in 5.3.1<br>'22'<br>According to Table 93<br>According to Table 94 |
| Lc field | Absent in the cases of STORE, RESTORE and ERASE, or<br>Number of bytes in the command data field in the case of SET |
| Data field | Absent in the cases of STORE, RESTORE and ERASE, or<br>Concatenation of control reference data objects in the case of SET |
| Le field | Absent |

| | |
|---|---|
| Data field | Absent |
| SW1-SW2 | Status bytes according to Table 95 |

**Table 93 — Options in P1**

| 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 and internal authentication |
| 1 | - | - | - | - | - | - | - | Verification, encipherment and external authentication |
| - | - | - | - | 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 | 0 | 0 | ERASE |
| Any other value is reserved for future use. | | | | | | | | |

**Table 94 — Control in P2**

| Value | Meaning |
|---|---|
| 'XY' | SEID in the cases of STORE, RESTORE and ERASE |
| 'A4'<br>'AA'<br>'B4'<br>'B6'<br>'B8' | Tag of the control reference template present in the command data field in the case of SET<br>— Control reference template for authentication (AT)<br>— Control reference template for hash-code (HT)<br>— Control reference template for cryptographic checksum (CCT)<br>— Control reference template for digital signature (DST)<br>— Control reference template for confidentiality (CT) |
| Any other value is reserved for future use. | |

**Table 95 — Status conditions in SW1-SW2**

| | SW1 | SW2 | Meaning |
|---|---|---|---|
| **Error** | '66' | '00' | The environment cannot be set or modified (no further information) |
| | '69' | '87'<br>'88' | Expected secure messaging data objects missing<br>Incorrect secure messaging data objects |
| | '6A' | '88' | Referenced data not found |

**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 96 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).

**Table 96 — Command-response pair for KEY DERIVATION function**

| | |
|---|---|
| CLA<br>INS<br>P1<br>P2 | As defined in 5.3.1<br>'22'<br>'X1' (SET, see Table 93)<br>CRT tag (e.g., 'A4' if an EXTERNAL AUTHENTICATE follows, or 'B4' if a VERIFY CRYPTOGRAPHIC CHECKSUM follows) |
| Lc field | Number of bytes in the command data field |
| Data field | {'94' - L - Data for deriving a key (mandatory)}; secure messaging data objects may be present |
| Le field | Absent |

| | |
|---|---|
| Data field | Absent |
| SW1-SW2 | Status bytes according to Table 95 |

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.

## 8.6 Transmission handling

### 8.6.1 GET RESPONSE command

The GET RESPONSE command is used to transmit [part of] response APDUs that otherwise could not be transmitted by the available transmission protocol. See examples in ISO/IEC 7816-3.

According to the physical interface specified in ISO/IEC 7816-3, the card shall answer to any cold or warm reset operation through the contacts. The answer to reset is a sequence of asynchronous characters, namely an initial character TS followed by up to 32 characters, namely a mandatory format character T0, optional interface characters, optional historical characters (up to 15 historical characters) and a conditional check character TCK.

The Answer-to-Reset is the string of up to 32 bytes conveyed by the answer to reset, namely a mandatory format byte T0 followed by optional interface bytes, optional historical bytes (up to 15 historical bytes coded according to 5.5) and a conditional check byte TCK. When TCK is present, the exclusive-oring of all the bytes T0 to TCK inclusive shall be null (see ISO/IEC 7816-3).

NOTE The initial character TS indicates conventions for decoding bytes in characters and offers an alternate measurement of the elementary time unit. Despite it may be decoded according to the indicated conventions, TS is a synchronization pattern, not a byte.

According to selection options indicated in P1-P2, the GET RESPONSE command may retrieve specific information in the card, e.g., ATR information when the physical interface does not allow the card to answer to reset, e.g., a universal serial bus or an access by radio frequency. Either one of the following specific information may be retrieved in the card.

— The historical bytes are a string of up to 15 bytes according to 5.5.

— The Answer-to-Reset is a string of up to 32 bytes, namely, a mandatory format byte T0 followed by an optional string of interface bytes, by an optional string of historical bytes and by a conditional check byte TCK.

— The content of EF.ATR is a set of BER-TLV data objects.

— The content of EF.DIR is a set of BER-TLV data objects.

**Table 97 — GET RESPONSE command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1-P2 | As defined in 5.3.1<br>'C0'<br>Either '0000', or 'FFFC' to 'FFFF' (any other value is reserved for future use) |
| Lc field | Absent |
| Data field | Absent |
| Le field | Maximum number of bytes expected in the response data field |

| | |
|---|---|
| Data field | [Part of] a response APDU according to Le if parameters P1-P2 are set to '0000', or<br>Content of EF.DIR if parameters P1-P2 are set to 'FFFC', or<br>Content of EF.ATR if parameters P1-P2 are set to 'FFFD', or<br>Answer-to-Reset (string of up to 32 bytes) if parameters P1-P2 are set to 'FFFE', or<br>Historical bytes (string of up to 15 bytes) if parameters P1-P2 are set to 'FFFF', or<br>Absent in any error case |
| SW1-SW2 | Status bytes according to Table 98 |

If the Le field contains only bytes set to '00', then within the limit of 256 for a short Le field or 65 536 for an extended Le field, all the available bytes should be returned.

**Table 98 — Status conditions in SW1-SW2**

| | SW1 | SW2 | Meaning |
|---|---|---|---|
| **Normal** | '61' | 'XX' | More bytes are available ('XX' tells the number of extra bytes still available by a subsequent GET RESPONSE) |
| **Warning** | '62' | '81' | Part of returned data may be corrupted |
| **Error** | '67' | '00' | Wrong length (incorrect Le field) |
| | '6A' | '81'<br>'82'<br>'86' | Function not supported<br>File not found<br>Incorrect parameters P1-P2 |
| | '6C' | 'XX' | Wrong length (wrong Le field; 'XX' tells the exact length). |

NOTE For ATR information, if the Le field codes a number less than the exact length, then rather than returning the beginning of the information followed by SW1-SW2 set to '61XX', the card should preferably reject the command by returning only SW1-SW2 set to '6CYY' for telling the exact length.

### 8.6.2 ENVELOPE command

The ENVELOPE command is used to transmit [part of] a command APDU or any data string that otherwise could not be transmitted by the available transmission protocol. See examples in ISO/IEC 7816-3.

NOTE Annex C shows the usage of the ENVELOPE command for secure messaging.

**Table 99 — ENVELOPE command-response pair**

| | |
|---|---|
| CLA<br>INS<br>P1-P2 | As defined in 5.3.1<br>'C2'<br>'0000' (any other value is reserved for future use) |
| Lc field | Number of bytes in the command data field |
| Data field | [Part of] a command APDU |
| Le field | Maximum number of bytes expected in the response data field or absent |

| | |
|---|---|
| Data field | [Part of] a response APDU according to Le or absent |
| SW1-SW2 | Status bytes according to Table 100 |

When protocol T = 0 uses ENVELOPE commands for transmitting data strings (see ISO/IEC 7816-3), the absence of command data field means "end of data string".

**Table 100 — Status conditions in SW1-SW2**

| | SW1 | SW2 | Meaning |
|---|---|---|---|
| **Error** | '67' | '00' | Wrong length (incorrect Lc field) |

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

# 9 Application-independent card services

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

Card services result from any combination of historical bytes, contents of one or more reserved EFs (EF.DIR is an optional working EF referenced by '3F002F00' as absolute path or by 30 as short EF identifier; EF.ATR is an optional working EF referenced by '3F002F01' as absolute path) and sequences of interindustry commands. Unless otherwise specified, every command-to-perform uses CLA set to '00' (see 5.3.1), 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 part of ISO/IEC 7816 for achieving similar functions. Therefore such solutions may not guarantee interchange.

The subsequent four clauses define the following card services.

1) Card-originated command-response pairs
2) Card identification
3) Application identification and selection
4) Data element retrieval

## 9.1 Card-originated command-response pairs

This clause specifies a mechanism whereby the card can originate command-response pairs, thus allowing communication service from card to interface device and also, from card to card, possibly through a network.

For clarity, this clause speaks of a query as [part of] a command APDU sent by the card and of a reply as [part of] a response APDU sent by an entity in the outside world; therefore, a complete set of queries and a complete set of replies together form a command-response pair.

This clause defines the following three features.

— How the card shall use bytes SW1-SW2 as a trigger telling that the card wants to issue a command message, for which the card possibly expects a response message.

— How the interface device shall use the GET DATA command (see 8.4.1) for retrieving a query from the card and the PUT DATA command (see 8.4.2) for transmitting a reply, if any, to the card.

— How the command-response pairs shall be formatted.

### 9.1.1 Triggering by the card

SW1-SW2 set to '62XY' with 'XY' in the range '02' to '80' means that the card has a query of 'XY' bytes that the interface device should retrieve and for which the card possibly expects a reply.

SW1-SW2 set to '64XY' with 'XY' in the range '02' to '80' means that the card has rejected the command; a possible execution of the command is conditioned by the recovery of a query of 'XY' 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.

The rejection of a reply with SW1-SW2 set to '64XY' with 'XY' in the range '02' to '80' means that the card wants to send at least one more query of 'XY' bytes.

The rejection of a command with SW1-SW2 set to '6401' means that the card is expecting an immediate reply.

### 9.1.2 Message retrieval and reply by the interface device

For retrieving a query of 'XY' bytes available in the card, the interface device shall send a GET DATA command with P1-P2 set to '0000' and a Le field set to 'XY'.

— SW1-SW2 set to '62XY' with 'XY' in the range '02' to '80' means that the interface device should retrieve a further query of 'XY' bytes and concatenate it to the already retrieved query before processing the command message in the outside world.

— SW1-SW2 set to '9000' means that the retrieved command message 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 parameters P1-P2 set to '0000'. If the reply is too long for a single command, then command chaining shall apply as defined in 5.3.6.

### 9.1.3 Command-response pair formats

The value of the first byte of the command message retrieved from the card fixes the formats as follows.

— If the value of the first byte is different from 'FF', then the command message shall be a command APDU and the response message a response APDU, according to ISO/IEC 7816-3.

— If the value of the first byte is 'FF', then the subsequent bytes shall code an initial protocol identifier according to ISO/IEC TR 9577; the command-response pair shall comply with the indicated protocol.

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 bytes SW1-SW2. This clause makes no assumption on the need for a reply and on the entity responsible for the contents of the possible reply.

## 9.2 Card identification

This service allows the interface device to identify the card and to deal with it. The card provides information to the outside world on its logical content directly, e.g., through card service data (see 5.5.2.3), and / or indirectly, e.g., through initial access data (see 5.5.2.4) indicating an access to a file implicitly selected immediately after the answer to reset and a possible protocol and parameters selection. Consequently, data available at this point may not be subsequently retrievable.

### 9.2.1 Initial data string recovery

Referenced by '41', '42' or '45', a COMPACT-TLV data object called "initial access data" may be present in the historical bytes (see 5.5.2.4). The referenced interindustry data element fixes a command-to-perform as follows.

If '41' introduces the COMPACT-TLV data object, then the data element is one byte. The command-to-perform shall be a READ BINARY command (see 8.2.1), i.e., CLA-INS-P1-P2 set to '00B0 0000' followed by a Le field consisting of the first and only byte of initial access data.

If '42' introduces the COMPACT-TLV data object, then the data element is two bytes.

— The first byte of initial access data indicates the structure (bit 8) and the short identifier (bits 5 to 1) of the EF to read, according to Table 101.

**Table 101 — First byte of initial access data referenced by '42'**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| **x** | **x** | **x** | **-** | **-** | **-** | **-** | **-** | **EF structure** |
| 0 | 0 | 0 | - | - | - | - | - | Record structure |
| 1 | 0 | 0 | - | - | - | - | - | Transparent structure |
| 0 | 1 | 0 | - | - | - | - | - | TLV structure for BER-TLV data objects |
| 1 | 1 | 0 | - | - | - | - | - | TLV structure for SIMPLE-TLV data objects |
| Any other value | | | - | - | - | - | - | Reserved for future use |
| - | - | - | x | x | x | x | x | Short EF identifier |

— The second byte of initial access data shall be the Le field in the appropriate READ command.
  - If bits 8 and 7 are set to 00 in the first byte, then the command-to-perform shall be a READ RECORD (S) command (see 8.3.1), i.e., CLA-INS-P1 set to '00B201', P2 consisting of bits 8 to 4 set to bits 5 to 1 (short EF identifier) of the first byte of initial access data and bits 3 to 1 set to 110, followed by a Le field consisting of the second and last byte of initial access data.
  - If bits 8 and 7 are set to 10 in the first byte, then the command-to-perform shall be a READ BINARY command (see 8.2.1), i.e., CLA-INS set to '00B0', P1 consisting of the first byte of initial access data, P2 set to '00' and a Le field consisting of the second and last byte of initial access data.
  - If bits 8 and 7 are set to 01 (BER-TLV) or 11 (SIMPLE-TLV) in the first byte, then the command-to-perform shall be a GET DATA command (see 8.4.1), i.e., CLA-INS-P1 set to '00CA03', P2 consisting of the first byte of initial access data (set to '4X' or '5X' for BER-TLV data objects or to 'CX' or 'DX' for SIMPLE-TLV data objects) and a Le field consisting of the second and last byte of initial access data.

If '45' introduces the COMPACT-TLV data object, then the data element is five bytes. The command-to-perform shall consist of the five bytes of initial access data.

A successful execution of the command-to-perform fixed by "initial access data" shall result in a response data field which is the "initial data string", i.e., a string of interindustry data objects every application might be interested in.

### 9.2.2 EF selection and access

When the path to an EF is known, the number of SELECT commands to issue equals the number of file identifiers in the path, minus one (the path always starts with the MF identifier or with the current DF identifier).

— If the path is more than four bytes, then from bytes 3 and 4 of the path until all available DF identifiers have been used, one or more SELECT commands shall be performed with CLA-INS-P1-P2 set to '00A4 0100'.

— The last and possibly only selection is an EF selection using the last two bytes of the path (an EF identifier) with CLA-INS-P1-P2 set to '00A4 0200'.

Once an EF has been selected, the contents relevant to interchange shall be the response data field to a READ command according to the file descriptor byte (see Table 21 in 5.6.1), if present in the control parameters.

— The READ BINARY command (see 8.2.1) consists of CLA-INS-P1-P2 set to '00B0 0000' followed by a Le field containing only bytes set to '00'.

— The READ RECORD (S) command (see 8.3.1) consists of CLA-INS-P1-P2 set to '00B2 0005' followed by a Le field containing only bytes set to '00'.

— The GET DATA command (see 8.4.1) consists of CLA-INS set to '00CA' followed by P1-P2 set to '00FF' for BER-TLV data objects or to '02FF' for SIMPLE-TLV data objects, followed by a Le field containing only bytes set to '00'.

In the absence of file descriptor byte in the control parameters of an EF, the command-to-perform is as follows.

— If the first software function table (see Table 16 in 5.5.2.7, card capabilities) is absent in the historical bytes or does not denote the support of records, then the command-to-perform is a READ BINARY as above.

— If the table denotes the support of records, then the command-to-perform is a READ RECORD (S) as above.

## 9.3 Application identification and selection

This service allows the interface device to know what application is active in the card (if any) as well as how to identify, select and start any application supported by the card.

The following interindustry data elements provide a generic support to application identification and selection.

**Application identifier —** Referenced by tag '4F', this interindustry data element of up to 16 bytes may be present in the initial data string, in EF.ATR, in EF.DIR and in the management data of any DF.

**Application label —** Referenced by tag '50', this interindustry data element consists of up to 16 bytes of free coding. Application providers may use it at the man-machine interface, e.g., a trademark to display to the customer.

**Path —** Referenced by tag '51', this interindustry data element is either empty, or consists of

- an absolute path, i.e., a concatenation of file identifiers starting with the MF identifier ('3F00'), or
- a relative path, i.e., a concatenation of file identifiers starting with the current DF identifier, or
- a qualified path, i.e., either an absolute path without '3F00' or a relative path without the current DF identifier, followed by a byte to use as parameter P1 in the relevant SELECT command(s).

The number of bytes is even in an absolute or relative path, but odd in a qualified path.

**Command-to-perform —** Referenced by tag '52', this interindustry data element is a command APDU.

**Discretionary data (or template) —** Referenced by tag '53' (or '73'), this interindustry data element (or template) groups relevant data elements (or data objects) 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 and IETF RFC 2396. It points to part of the software required in the interface device to communicate with the application in the card.

**Application template —** Referenced by tag '61', this interindustry template shall contain one application identifier, but it should not contain two or more application identifiers. It may optionally contain other interindustry data objects relating to the application as listed in Table 102 and defined above. It may be present in EF.ATR, in EF.DIR and in the management data of any DF. If several application identifiers are valid names for the same DF, then each one should be presented in a different application template.

**Table 102 — Interindustry data objects for application identification and selection**

| Tag | Value |
|---|---|
| '4F' | Application identifier (see 5.1) |
| '50' | Application label (up to 16 bytes, free coding) |
| '51' | Path (see 5.2.1) |
| '52' | Command-to-perform |
| '53', '73' | Discretionary data, discretionary template |
| '5F50' | Uniform resource locator (as defined in IETF RFC 1738 and IETF RFC 2396) |
| '61' | Interindustry template for nesting one application identifier data object and other application-related data objects |

The card shall support at least one of the following application selection methods.

1) Implicit application selection
2) Direct application selection by application identifier
3) Application selection using EF.DIR or EF.ATR

### 9.3.1 Implicit application selection

If an application is implicitly selected, then an application identifier should be present in the historical bytes or in the initial data string. The presence of an application identifier in the historical bytes 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.

NOTE The implicit application selection is not recommended for multi-application cards.

### 9.3.2 Direct application selection by application identifier

The card in a multi-application environment shall respond positively to a SELECT command specifying any AID as DF name. The interface device may thus explicitly select an application without previously checking the presence of the application in the card.

The card shall support a SELECT command with CLA-INS-P1-P2 set to '00A4 0400' for the first selection with a given and preferably complete application identifier in the command data field (see Tables 46 and 47). Depending on whether the application is present or not, the card shall either execute or reject the command.

In the case of a selection with a truncated DF name, the full DF name will be made available in the response data field as the file control parameter with tag '84' (see Table 20). The first selection is implementation-dependent, e.g., the first occurrence in a static list or the last activated application in a previous session.

If the card supports selection with a truncated DF name, then for the next selections, if any, the card shall support a SELECT command with CLA-INS-P1-P2 set to '00A4 0402' with the same command data field.

### 9.3.3 Application selection using EF.DIR or EF.ATR

If present in the card, EF.DIR and / or EF.ATR indicate applications and determine which commands shall be performed in order to select the indicated applications. For a multi-application interface device, the use of EF.DIR or EF.ATR may be more efficient than the previous method.

EF.DIR contains a set of application templates and / or application identifier data objects, in any order. Application templates and application identifier data objects may be present in EF.ATR. In EF.DIR and EF.ATR, sequences of '00' or 'FF' bytes without any meaning may replace erased data objects.

— If an application identifier data object is not part of an application template and not accompanied by a path data object or by a command-to-perform data object, then direct application selection applies with the indicated AID as DF name.

— If an application identifier data object is part of an application template together with a path data object, either empty or with two bytes set to '3F00', then the MF supports the application.

— If an application identifier data object is part of an application template together with a path data object with an even number of bytes, greater than two, then the path is either absolute or relative and the application selection should be done by one or more SELECT commands, with CLA-INS-P1-P2-Lc set to '00A4 0100 02' (see 9.3.2).

— If an application identifier data object is part of an application template together with a path data object with an odd number of bytes, greater than two, i.e., a path followed by a parameter P1 (see 9.3), then the path is qualified and the application selection depends on the value of parameter P1.
  - If parameter P1 is set to '08' or '09', then the card shall support a SELECT command where the qualified path specifies P1, Lc and the data field and where parameter 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-Lc set to '0002'. Every file along the path shall be selected sequentially.

— If an application identifier data object is part of an application template together with one or more command-to-perform data objects, then the application selection is done by the indicated command(s). If several, they shall be performed in the order presented in the template.

## 9.4 Data element retrieval

This service allows the interface device to retrieve interindustry data elements used for application-independent international interchange.

Before selecting an application, interindustry data objects should be retrieved directly or indirectly from the historical bytes, the initial data string retrieved by the use of initial access data (see 5.5.2.4 and 9.2.1), EF.ATR and EF.DIR possibly indicated by the card service data byte (see 5.5.2.3), in that order, when present. These interindustry data objects shall be interpreted according to tag allocation schemes specified in 5.4.4.

Once an application is selected, interindustry data objects should be retrieved directly or indirectly from the management data (see 5.6) of the DF and from specific EFs within the current DF.

— Interindustry data objects may be present in the management data of any file (see 5.6).

— Interindustry data elements may be retrieved in files denoted by their path in a wrapper (see 9.4.1). Selection and reading of an EF known by its path are defined in 9.2.2.

— Interindustry data objects may be retrieved by GET DATA commands (see 8.4.1).

### 9.4.1 Indirect references to data elements

Element lists, tag lists, header lists, extended header lists and wrappers are interindustry data elements for indirectly referencing data elements in byte strings, e.g., contents of EFs of transparent structure, data fields resulting from commands-to-perform (see 9.2.2), byte strings to sign (see ISO/IEC 7816-8).

**Element list —** Referenced by tag '5F41', this interindustry data element denotes that the information to retrieve is not presented as data objects, but under application control. It shall be used only within the wrapper template. Its structure and the returned information are outside the scope of ISO/IEC 7816.

**Tag list —** Referenced by tag '5C', this interindustry data element is a concatenation of tag fields without delimitation. The byte string consists of the data objects, in the same order as the tag list.

**Header list —** Referenced by tag '5D', this interindustry data element is a concatenation of pairs of tag fields and length fields without delimitation. The byte string consists of the value fields, in the same order as the header list.

**Extended header list —** Referenced by tag '4D', this interindustry data element is a concatenation of pairs of tag fields and length fields without delimitation. The byte string is built as follows.

— If a tag denotes a primitive encoding, then the pair of tag field and length field is replaced by data referenced by the tag. A zero length means that the complete data object or data element is included in the byte string. A non-zero length means a maximum length, i.e., a request for a possible truncation.

— A tag denoting a constructed encoding followed by a non-zero length introduces a subsequent value field that is an extended header list. A tag denoting a constructed encoding followed by a zero length is ignored.

The byte string consists of either

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

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

The encoding of the byte string, namely, data objects or data elements, is indicated by an appropriate parameter of the command, e.g., either an appropriate encoding of the data field (either constructed for those containing data objects or primitive for those containing data elements), or tags 'AC' or 'BC' (see Table 27) used in the PERFORM SECURITY OPERATION command (see ISO/IEC 7816-8).

EXAMPLES The following extended header list references the subsequent three primitive data objects.

| Primitive $T_1$ | '00' | Constructed tag T | L = 4 | Primitive $T_2$ | '00' | Primitive $T_3$ | L = 5 |
|---|---|---|---|---|---|---|---|

| Primitive $T_1$ | $L_1$ | $Value_1$ |
|---|---|---|

| Primitive $T_2$ | $L_2$ | $Value_2$ |
|---|---|---|

| Primitive $T_3$ | $L_3$ (≥5) | $Value_3$ |
|---|---|---|

Case 1: The byte string is a concatenation of data elements.

| $Value_1$ | $Value_2$ | The first five bytes of $value_3$ |
|---|---|---|

Case 2: The byte string is a concatenation of data objects.

| $T_1$ | $L_1$ | $Value_1$ | T | $L = L_2 + 9$ | $T_2$ | $L_2$ | $Value_2$ | $T_3$ | L = 5 | The first five bytes of $value_3$ |
|---|---|---|---|---|---|---|---|---|---|---|

**Wrapper —** Referenced by tag '63', this interindustry template shall consist of two parts.

— The first part is either an element list (tag 5F41'), or a tag list (tag '5C'), or a header list (tag '5D'), or an extended header list (tag '4D').

— The second part is a path to an EF (tag '51') and / or one or more commands-to-perform (tag '52'). If several, the commands-to-perform shall be executed in the presented order.

A data object referenced e.g., in a tag list, or a data element referenced e.g., in a header list, either shall be contained in a file denoted by its path, or shall be (part of) the response data field to the last command-to-perform. Only one indirect reference shall be given in a wrapper. There may be more than one wrapper.

EXAMPLE The following wrapper template consists of a tag list and one command-to-perform.

{'63'-L-{'5C'-L-(Tag1-Tag2-Tag3)}-{'52'-L-Command-to-perform}}

### 9.4.2 Specific interindustry data elements

According to its needs, any application may use the following interindustry data elements and templates.

**Name of an individual —** Referenced by tag '5B', this interindustry data element consists of up to 39 bytes; each byte is a character as defined in ISO/IEC 7501-1. The data element consists of surname, i.e., family name, given name(s), i.e., forename(s), name suffix, e.g., Jr., number, and filler(s), all coded according to ISO/IEC 8859-1.

National languages with non-Latin characters shall be transliterated or transcribed into the Latin alphabet using the appropriate ISO standard. In cases where names cannot be shown in full or a special alphabet is needed or the transliteration or transcription is not sufficient, the qualified name template should be used.

**Proprietary login data —** Referenced by tag '5E', this interindustry data element consists of login data with proprietary structures not specified in ISO/IEC 7816.

**Magnetic stripe data —**

— Referenced respectively by tags '5F21', '5F22' and '5F23', these interindustry data elements shall code card tracks 1, 2 and 3. Such a tag shall be used when the data element is identical to the data coded on the corresponding track on the magnetic stripe of the card (see ISO/IEC 7813 and ISO 4909).

— Referenced respectively by tags '56', '57' and '58', these interindustry data elements shall code application tracks 1, 2 and 3. Such a tag shall be used when, while formatted according to ISO/IEC 7813 and ISO 4909, the data element may differ from the data coded on the corresponding track of the magnetic stripe of the card.

**PIN usage policy —** Referenced by tag '5F2F', this interindustry data element shall consist of two bytes. It tells the tests the terminal shall perform in order to determine whether a PIN (personal identification number) is applicable to the current transaction, and, therefore, whether the terminal should prompt for the PIN. If set to one, bit 8 of the first byte specifies that a PIN applies to this application and the terminal should prompt for the PIN. The meaning of the other fifteen bits is application-dependent. If all bits are set to zero, then the terminal should not prompt for the PIN. If bit 8 of the first byte is set to one or if any test implies a PIN, but the PIN cannot be presented, then the action to take is application-dependent.

**Integrated circuit manufacturer identifier —** Referenced by tag '5F4D', this interindustry data element shall be numbered and registered according to ISO/IEC 7816-13. It identifies the manufacturers of integrated circuits embedded in contact and / or contactless cards. It may be present in pre-issuing data (compact tag '6' in the historical bytes and interindustry tag '46' in EF.ATR) on a proprietary basis.

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

**Login template** — Referenced by tag '6A', this interindustry template shall consist of one or more primitive data objects. Within the login template, the context-specific class (first byte in the range '80' to 'BF') is reserved for login data objects such as qualifiers, numbers, texts and delay indicators, as listed in Table 103 and specified hereafter.

**Table 103 — Login data objects**

| Tag | | Meaning |
|---|---|---|
| '6A' | | Interindustry template for nesting login data objects with the following tags |
| | '80'<br>'81'<br>'82'<br>'83', '84' | Qualifier<br>Number<br>Text<br>Delay indicators |

— **Qualifier —** Referenced by tag '80' in a login template, this data element shall consist of one to nine bytes: a mandatory first byte coding a rank, followed by up to eight optional bytes coding a mnemonic. It shall qualify the subsequent objects in the template, until the next qualifier, if any.

- The rank is a number from zero to 255. If two or more qualifiers have the same rank within the same context, then only the set of objects qualified by the most recent one is valid.
- The mnemonic is a string of up to eight bytes consisting of 7-bit characters (bit 8 set to 0, see ISO/IEC 646) to display at the man-machine interface.

— **Number —** Referenced by tag '81' in a login template, this data element shall consist of an even number of quartets where each quartet codes one character for representing a telephone number according to table 104.

**Table 104 — Telephone number**

| Quartet | Character | Meaning |
|---|---|---|
| '0' to '9' | 0 to 9 | Decimal digits |
| 'A' | ( | Opening bracket |
| 'B' | ) | Closing bracket |
| 'C' | C | Requirement for connecting to the line before continuing |
| 'D' | + | Introduction to an international telephone number |
| 'E' | - | If first, introduction of a number to use without prefix<br>If not first, requirement for a delay (two seconds) before continuing |
| 'F' | | Reserved for padding |

— **Text —** Referenced by tag '82' in a login template, this data element shall consist of one or more bytes where each byte codes one character. Bit 8 sets the difference between data characters (bit 8 set to zero) and control characters (bit 8 set to one). The byte string consists of one or more strings of data characters (7-bit character, see ISO/IEC 646) separated by strings of control characters. The following control characters are defined.

- '80' — A message has to be received before sending the next character.
- 'C0' — A modulation has to be present before sending the next character.
- '8X' — X characters have to be received in echo before waiting for a message.

— **Delay indicators —** Referenced by tag '83' or '84' in a login template, this data element shall consist of one byte as specified in Table 105.

- When present, a delay indicator data object with tag '83' fixes the time for detecting an end of message. The default value shall be two seconds.
- When present, a delay indicator data object with tag '84' fixes the time for detecting an absence of response. The default value shall be sixty seconds.

**Table 105 — Delay indicator byte**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | | | | | | | Any other value is reserved for future use. |
| - | - | x | x | - | - | - | - | The time unit is |
| - | - | 0 | 0 | - | - | - | - | — 100 milliseconds |
| - | - | 0 | 1 | - | - | - | - | — 1 second |
| - | - | 1 | 0 | - | - | - | - | — 10 seconds |
| - | - | 1 | 1 | - | - | - | - | — 100 seconds |
| | | | | x | x | x | x | Number of time units from zero to fifteen |

**Qualified name template —** Referenced by tag '6B', this interindustry template shall consist of

— one or more object identifiers (tag '06') referring to the standards defining the qualified name presentation;

— a name (tag '80' or 'A0'), the value and coding of which are defined by the aforementioned standards;

— other related optional information (e.g., sex, nationality, place of birth).

**Cardholder image template —** Referenced by tag '6C', this interindustry template shall contain at least one data object as defined hereafter, possibly preceded by an authority indicator as listed in Table 11, for identifying the authority responsible for the data object format.

— **Cardholder biometric data —** Referenced by tag '5F2E', this interindustry data element contains biometric data for verifying the claimed identity of the person presenting the card. Examples of biometric data are fingerprints, palm prints, voiceprints, dynamic signatures, etc.

— **Cardholder portrait image —** Referenced by tag '5F40', this interindustry data element shall be formatted as defined in ISO/IEC 10918-1, unless otherwise specified and / or requested by an authority.

— **Cardholder handwritten signature image —** Referenced by tag '5F43', this interindustry data element shall be formatted as defined in ISO/IEC 11544 unless otherwise specified and/or requested by an authority.

NOTE The use of this interindustry data object should be associated with appropriate security measures.

**Application image template —** Referenced by tag '6D', this interindustry template shall contain at least an application image (tag '5F44'), i.e., an icon or a logo related to the application. It may also contain an authority indicator (see Table 11) identifying the authority responsible for the data format of the application image. In the absence of authority indicator, the format shall be as defined in ISO/IEC 10918-1.

**Display control template —** Referenced by tag '7F20', this interindustry template may contain one or more data objects, the value of which, either directly or indirectly through templates, is not intended to be displayed and should only be used, when relevant, for processing of transmission.

### 9.4.3 Interchange profile

The specification of data objects associated with the interchange profile of the card (e.g., available authentication methods and security functions) may be further detailed in future parts of ISO/IEC 7816. Table 106 shows interindustry data objects reserved for interchange profile.

**Table 106 — Interindustry data objects reserved for interchange profile**

| Tag | Value |
|---|---|
| '5F29' | Interchange profile |
| '5F37' | Static internal authentication (one-step) |
| '5F38' | Static internal authentication – first associated data |
| '5F39' | Static internal authentication – second associated data |
| '5F3A' | Dynamic internal authentication |
| '5F3B' | Dynamic external authentication |
| '5F3C' | Dynamic mutual authentication |

# Annex A
(informative)

# Application identifiers using issuer identification numbers

## A.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.

## A.2 Format

In any AID where bits 8 to 5 of the first byte together code a digit in the range '0' to '9', the first and possibly only field shall be an issuer identification number according to ISO/IEC 7812-1.

NOTE In the previous edition of ISO/IEC 7812, an issuer identification number might consist of an odd number of quartets in the range '0' to '9'. Then it was mapped into a byte string by setting bits 4 to 1 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 A.1 shows an AID using an issuer identification number; it is a string of two to sixteen bytes.

| | | |
|---|---|---|
| Issuer identification number according to ISO/IEC 7812-1 (two or more bytes) | 'FF' | Proprietary application identifier extension (PIX) |

**Figure A.1 — AID using an issuer identification number**

# Annex B
(informative)

# Object identifiers and tag allocation schemes

## B.1 Object identifiers denoting ISO standards

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 8 is set to zero in the last byte of a series and to one in the previous bytes, if there is more than one byte. The concatenation of bits 7 to 1 of the bytes of a series codes 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/IEC 7816-15 is denoted as {iso standard 7816 15}.

— 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 8 in each byte, the coding of the first series is therefore 1011 1101 0000 1000, equal to 'BD08'.

— The second series is '0F' for identifying part 15.

The data element '28 BD08 0F' identifies ISO/IEC 7816-15. It may be used in an AID of standard category.

AID = 'E8 28 BD08 0F XX … XX' (ISO/IEC 7816-15 specifies the application identifier extension 'XX … XX').

As a second example, ISO 9992-2 is denoted as {iso standard 9992 2}. 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 8 in each byte, the coding 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 data object (universal tag '06', see ISO/IEC 8825-1).

DO = {'06 04 28 CE 08 02'}

## B.2 Default tag allocation scheme

DO1 = {'59 02 95 02'}

DO2 = {'5F 24 03 97 03 31'}

DO1 (tag '59', card expiration date) indicates February 1995 as card expiration date (see ISO/IEC 7816-6).

DO2 (tag '5F24', application expiration date) indicates March 31$^{st}$ 1997 as application expiration date.

## B.3 Compatible tag allocation scheme

### First example

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 0A' {'5F 29 03 XX XX XX'} {'81 02 XX XX'}}

DO1 (tag '78', compatible tag allocation authority) indicates a compatible tag allocation scheme defined in ISO 9992-2 denoted by its object identifier. If DO1 appears either in the initial data string or in EF.ATR, then the tag allocation authority is valid for the entire card. If DO1 appears in the management data of a DF, then the tag allocation authority is valid within that DF.

DO2 (tag '5F24', application expiration date) indicates March 31st 1997 as application expiration date.

DO3 (tag '70', interindustry template according to the included tag allocation authority) contains a data object, tag '80', defined in ISO 9992-2; the meaning of tag '70' is also defined in ISO 9992-2.

DO4 (tag '67', authentication data template) contains the interchange profile data object, tag '5F29', and a data object, tag '81', defined in ISO 9992-2; the meaning of tag '67' is defined in ISO/IEC 7816-6.

### Second example

DO2 = {'5F 24 03 97 03 31'}

DO3 = {'70 0C' {'06 04 28 CE 08 02'} {'80 04 XX XX XX XX'}}

DO4 = {'67 06' {'5F 29 03 XX XX XX'}}

DO2 (tag '5F24', application expiration date) indicates March 31st 1997 as application expiration date.

DO3 (tag '70', interindustry template defined according to the included object identifier) contains a data object, tag '06', which specifies that the subsequent data object, tag '80', is defined in ISO 9992-2. The meaning of tag '70' is also defined in ISO 9992-2.

DO4 (tag '67', interindustry authentication data template) contains the interchange profile data object, tag '5F29'. Note that it cannot contain data objects defined in ISO 9992-2, because of the choice not to transmit the interindustry data object with tag '78'.

## B.4 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 denoted 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 or in EF.ATR if the tag allocation authority is valid for the entire card, or

— in the management data of a DF if the tag allocation authority is valid within that DF.

DO2 (tag '7E') is an interindustry template for nesting interindustry data objects. Note that the interindustry data object "application expiration date", tag '5F24', is present, indicating 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 C
(informative)

# Secure messaging

## C.1 Cryptographic checksum

This clause shows the use of secure messaging (see 6) and cryptographic checksums (see 6.2.3.1) for each of the four cases of command-response pairs defined in ISO/IEC 7816-3.

In the examples, the notation CLA* means the use of secure messaging in the data fields: in byte CLA (see 5.3.1), bit 4 set to one and either bits 8 and 7 set to 00 (basic class) or bits 8 to 5 set to 1000, 1001 or 1010.

In the examples, the notation CLA** means a byte CLA* where bit 3 is set to 1, i.e., that the command header shall be included in the computation of a data element for authentication.

In the examples, the notation T* means a SM tag where the tag number is odd, i.e., that the SM data object shall be included in the computation of a data element for authentication.

— **Case 1 — No data, no data**

The unsecured command-response pair is as follows.

| Command header | Command body |
|---|---|
| CLA-INS-P1-P2 | Absent |

| Response body | Response trailer |
|---|---|
| Absent | SW1-SW2 |

— **Case 1.a — Status not protected**

The secured command APDU is as follows.

| Command header | Command body |
|---|---|
| CLA*-INS-P1-P2 | {New Lc field} - {New data field (= T - L - Cryptographic checksum)} |

If the length of the cryptographic checksum is four bytes, then New Lc field is set to '06'.

New data field = One data object = {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum (bit 3 set to one in CLA*) =
One block = {CLA** - INS - P1 - P2 - Padding}

The secured response APDU is as follows.

| Response body | Response trailer |
|---|---|
| Absent | New SW1-SW2 |

— **Case 1.b — Status protected**

The secured command APDU is as follows.

| Command header | Command body |
|---|---|
| CLA*-INS-P1-P2 | {New Lc field} - {New data field (= T - L - Cryptographic checksum)} - {New Le field (= '00')} |

New data field = One data object = {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum (bit 3 set to one in CLA*) =
One block = {CLA** - INS - P1 - P2 - Padding}

The secured response APDU is as follows.

| Response body | Response trailer |
|---|---|
| New data field (={T* - L - New SW1-SW2} - {T - L - Cryptographic checksum}) | New SW1-SW2 |

New data field = Two data objects = {T* - L - New SW1-SW2} - {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum = One block = {T* - L - New SW1-SW2 - Padding}

— **Case 2 — No data, data**

The unsecured command-response pair is as follows.

| Command header | Command body |
|---|---|
| CLA-INS-P1-P2 | Le 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 Lc field - New data field - {New Le field (one or two bytes set to '00')} |

New data field = Two data objects = {T* - L – Number Le} - {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum =

— If bit 3 is set to zero in CLA*, then one block = {T* - L - Number Le - Padding}

— If bit 3 is set to one in CLA*, then two blocks =
{CLA** - INS - P1 - P2 - Padding} - {T* - L - Number Le - Padding}

The secured response APDU is as follows.

| Response body | Response trailer |
|---|---|
| New data field<br>(={T* - L - Plain value} - {T* - L - New SW1-SW2} - {T - L - Cryptographic checksum}) | New SW1-SW2 |

New data field = Three data objects =
{T* - L - Plain value} - {T* - L - New SW1-SW2} - {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum =
One or more blocks = {T* - L - Plain value – T* - L - New SW1-SW2 - Padding}

— **Case 3 — Data, no data**

The unsecured command-response pair is as follows.

| Command header | Command body |
|---|---|
| CLA-INS-P1-P2 | Lc 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 Lc field - New data field |

New data field = Two data objects = {T* - L - Plain value} - {T - L -Cryptographic checksum}

Data covered by the cryptographic checksum =

— If bit 3 is set to zero in CLA*, one or more blocks = {T* - L - Plain value - Padding}

— If bit 3 is set to one in CLA*, two or more blocks =
{CLA** - INS - P1 - P2 - Padding} - {T* - L - Plain value - Padding}

The secured response APDU is as follows.

| Response body | Response trailer |
|---|---|
| Absent | New 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 Le field (= '00') |

New data field = Two data objects = {T* - L - Plain value} - {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum =

— If bit 3 is set to zero in CLA*, one or more blocks = {T* - L - Plain value - Padding}

— If bit 3 is set to one in CLA*, two or more blocks =
{CLA** - INS - P1 - P2 - Padding} - {T* - L - Plain value - Padding}

The secured response APDU is as follows.

| Response body | Response trailer |
|---|---|
| New data field (= {T* - L - New SW1-SW2} - {T - L - Cryptographic checksum}) | New SW1-SW2 |

New data field = Two data objects = {T* - L - New SW1-SW2} - {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum = One block = {T* - L - New SW1-SW2 - Padding}

— **Case 4 — Data, data**

The unsecured command-response pair is as follows.

| Command header | Command body |
|---|---|
| CLA-INS-P1-P2 | Lc field - Data field - Le 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 Lc field - New data field - New Le field (one or two bytes set to '00') |

New data field = Three data objects =
{T* - L - Plain value} - {T* - L - Number Le} - {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum =

— If bit 3 is set to zero in CLA*, one or more blocks =
{T* - L - Plain value – T* - L - Number Le - Padding}

— If bit 3 is set to one in CLA*, two or more blocks =
{CLA** - INS - P1 - P2 - Padding} - {T* - L - Plain value – T* - L - Number Le - Padding}

The secured response APDU is as follows.

| Response body | Response trailer |
|---|---|
| New data field<br>(={T* - L - Plain value} - {T* - L - New SW1-SW2} - {T - L - Cryptographic checksum}) | New SW1-SW2 |

New data field = Three data objects =
{T* - L - Plain value} - {T* - L - New SW1-SW2} - {T - L - Cryptographic checksum}

Data covered by the cryptographic checksum = One or more blocks =
{T* - L - Plain value – T* - L - New SW1-SW2 - Padding}

## C.2 Cryptograms

The use of cryptograms with and without padding (see 6.2.2) is shown in command and response data fields. Instead of the plain value data objects in the previous examples, data objects for confidentiality shall be used as follows.

— **Case a — Plain value not coded 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 coded in BER-TLV, possibly padded according to the indicator byte

— **Case b — Plain value coded in BER-TLV**

Data field = {T - L - Cryptogram}

Plain value conveyed by the cryptogram = String of concealed bytes =
BER-TLV data objects (padding depending on the algorithm and its mode of operation)

## C.3 Control references

The use of control references (see 6.3.1 and 6.3.1.1) is shown.

Command data field = {T - L - Control reference template},
where control reference template = {T - L - File reference} - {T - L - Key reference}

## C.4 Response descriptor

The use of response descriptor (see 6.3.2) 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}

## C.5 ENVELOPE command

The use of the ENVELOPE command (see 8.6.2) 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

# Annex D
(informative)

# Chains of GENERAL AUTHENTICATE commands

## D.1 Introduction

This annex illustrates data objects in data fields of GENERAL AUTHENTICATE commands when using zero-knowledge techniques. ISO/IEC 9798-5 specifies entity authentication using zero-knowledge techniques.

— A proving entity knows a secret solution to a public problem, e.g., a private number and a public key.

— A verifying entity knows the public problem, e.g., the corresponding public number and the public key.

— As a result of zero-knowledge protocols, the verifying entity is convinced that the public problem has a solution and that the proving entity knows it. Therefore, the secret solution, i.e., the private number, remains secret.

NOTE ISO/IEC 9798-5 specifies the GQ1 technique, namely, how to verify an RSA digital signature without knowing its value, or alternately, how to prove the knowledge of an RSA digital signature without revealing its value. An issuing authority knows and uses an RSA private signature key for computing digital signatures according to a digital signature scheme based on RSA (e.g., ISO/IEC 9796-2). As specified by the RSA digital signature scheme, a format mechanism transforms the identification data template of every proving entity into a public number G. The corresponding private number Q is the RSA digital signature of the identification data template. The issuing authority computes private numbers. Every proving entity knows its private number Q. Every proving entity and every verifying entity know the RSA public verification key linking every pair of numbers: a public number G and a private number Q.

The basic protocol involves three numbers together forming a zero-knowledge triple, namely a commitment, a challenge and a response. There are two modes of production of such triples: a public mode and a private mode.

— The private mode works in two steps: as a first step, the proving entity privately produces a random number and transforms it into a commitment under the control of the public verification key; as a second step, the proving entity uses the random number and the private number for getting the response to a single challenge and then erases the random number.

— The public mode works in one step: the verifying entity uses the public number and the public verification key for getting a commitment from any challenge / response pair.

NOTE The use of committed challenges allows the proving entity to produce commitments after challenge selection and therefore to be sure that challenges are selected independently from commitments.

This annex illustrates three authentication functions.

— INTERNAL AUTHENTICATE function — An entity in the outside world authenticates an entity in the card.

— EXTERNAL AUTHENTICATE function — An entity in the card authenticates an entity in the outside world.

— MUTUAL AUTHENTICATE function — Both entities authenticate each other.

Such a function is implemented as a chain of two or more GENERAL AUTHENTICATE command-response pairs.

— CLA is set to 00x1 00xx in the first command of the chain up to the penultimate one and to 0x10 00xx in the last one: bit 5 chains the command-response pairs (see 5.3.6); bits 6, 2 and 1 code a logical channel number, constant within a chain (see 5.3.7).

— INS-P1-P2 is set to '87 00 00'. The odd INS code ('87') indicates that the data fields are BER-TLV coded.

— The Lc field is appropriate, i.e., according to the data objects present in the command data field.

— Depending upon whether a response data field is expected or not, the Le field is either set to '00' or absent.

## D.2 INTERNAL AUTHENTICATE function

If the first data field conveys a commitment request, namely, either an empty commitment ('80 00') or an empty authentication code ('84 00'), then the chain implements an INTERNAL AUTHENTICATE function.

— **Case 1 — Basic protocol** (two command-response pairs)

**Commitment from the card**

| | |
|---|---|
| Command data field | {'60'-'04'-{'A0'-'00'}-{'80'-'00'}} |
| Response data field | {'60'-L-{'A0'-L-Identification Data Template}-{'80'-L-Commitment}} |

**Challenge from the outside world and response from the card**

| | |
|---|---|
| Command data field | {'60'-L-{'81'-L-Challenge}-{'82'-'00'}} |
| Response data field | {'60'-L-{'82'-L-Response}} |

— **Case 2 — Committed challenge** (two command-response pairs)

**Commitment from the card**

| | |
|---|---|
| Command data field | {'60'-L-{'83'-L-Committed Challenge}-{'A0'-'00'}-{'80'-'00'}} |
| Response data field | {'60'-L-{'A0'-L-Identification Data Template}-{'80'-L-Commitment}} |

**Challenge from the outside world and response from the card**

| | |
|---|---|
| Command data field | {'60'-L-{'81'-L-Challenge}-{'82'-'00'}} |
| Response data field | {'60'-L-{'82'-L-Response} if the challenge is correct<br>Absent if the challenge is incorrect |

— **Case 3 — Extension to data field authentication** (two command-response pairs)

The card has hashed previously exchanged data fields: the result is a current hash-code. The card integrates its commitment data object for getting an authentication code and transmits it with tag '84'.

**Commitment from the card**

| | |
|---|---|
| Command data field | {'60'-'04'-{'A0'-'00'}-{'84'-'00'}} |
| Response data field | {'60'-L-{'A0'-L-Identification Data Template}-{'84'-L-Authentication code}} |

**Challenge from the outside world and response from the card**

| | |
|---|---|
| Command data field | {'60'-L-{'81'-L-Challenge}-{'82'-'00'}} |
| Response data field | {'60'-L-{'82'-L-Response}} |

## D.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 chain implements an EXTERNAL AUTHENTICATE function.

— **Case 1 — Basic protocol** (two command-response pairs)

**Commitment from the outside world and challenge from the card**

| | |
|---|---|
| Command data field | {'60'-L-{'A0'-L-Identification Data Template}-{'80'-L-Commitment}-{'81'-'00'}} |
| Response data field | {'60'-L-{'81'-L-Challenge}} |

**Response from the outside world and control from the card**

| | |
|---|---|
| Command data field | {'60'-L-{'82'-L-Response}} |
| Response data field | Absent |

— **Case 2 — Committed challenge** (three command-response pairs)

**Committed challenge from the card**

| | |
|---|---|
| Command data field | {'60'-L-{'A0'-L-Identification Data Template}-{'83'-'00'}} |
| Response data field | {'60'-L-{'83'-L-Committed challenge}-{'80'-'00'}} |

**Commitment from the outside world and challenge from the card**

| | |
|---|---|
| Command data field | {'60'-L-{'80'-L-Commitment}-{'81'-'00'}} |
| Response data field | {'60'-L-{'81'-L-Challenge}} |

**Response from the outside world and control from the card**

| | |
|---|---|
| Command data field | {'60'-L-{'82'-L-Response}} |
| Response data field | Absent |

— **Case 3 — Extension to data field authentication** (two command-response pairs)

A proving entity (i.e., knowing the private key) has hashed previously exchanged data fields: the result is a current hash-code. It integrates its commitment data object for getting an authentication code and transmits it with tag '84'.

**Commitment from the outside world and challenge from the card**

| | |
|---|---|
| Command data field | {'60'-L-{'A0'-L-Identification Data Template}-{'84'-L-Authentication code}-{'81'-'00'}} |
| Response data field | {'60'-L-{'81'-L-Challenge}} |

**Response from the outside world and control from the card**

| | |
|---|---|
| Command data field | {'60'-L-{'82'-L-Response}} |
| Response data field | Absent |

## D.4 MUTUAL AUTHENTICATE function

If the first data field conveys no empty data object, then the chain implements a MUTUAL AUTHENTICATE function; the outside world requests the same data objects in the response data field as in the command data field.

— **Case 1 — Basic protocol** (three command-response pairs)

**Commitment**

| | |
|---|---|
| Command data field | {'60'-L-{'A0'-L-Identification Data Template}-{'81'-L-Commitment}} |
| Response data field | {'60'-L-{'A0'-L-Identification Data Template}-{'81'-L-Commitment}} |

**Challenge**

| | |
|---|---|
| Command data field | {'60'-L-{'81'-L-Challenge}} |
| Response data field | {'60'-L-{'81'-L-Challenge}} |

**Response**

| | |
|---|---|
| Command data field | {'60'-L-{'82'-L-Response}} |
| Response data field | {'60'-L-{'82'-L-Response}} if the response is correct<br>Absent if the response is incorrect |

— **Case 2 — Committed challenge** (four command-response pairs)

**Committed challenge**

| | |
|---|---|
| Command data field | {'60'-L-{'A0'-L-Identification Data Template}-{'83'-L-Committed challenge}} |
| Response data field | {'60'-L-{'A0'-L-Identification Data Template}-{'83'-L-Committed challenge}} |

| | **Commitment** |
|---|---|
| Command data field | {'60'-L-{'80'-L-Commitment}} |
| Response data field | {'60'-L-{'80'-L-Commitment}} |
| | **Challenge** |
| Command data field | {'60'-L-{'81'-L-Challenge}} |
| Response data field | {'60'-L-{'81'-L-Challenge}} if the challenge is correct<br>Absent if the challenge is incorrect |
| | **Response** |
| Command data field | {'60'-L-{'82'-L-Response}} |
| Response data field | {'60'-L-{'82'-L-Response}} if the response is correct<br>Absent if the response is incorrect |

— **Case 3 — Extension to key establishment** (four command-response pairs)

A pair of exponential data elements allows establishing a session key (see ISO/IEC 11170-3).

The first command-response pair exchanges dynamic authentication templates nesting an identification data template and 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 integrated for getting a current hash-code; then the response data field, i.e., the second dynamic authentication template is integrated for updating the current hash-code; the current hash-code should be the same for both entities. Finally a commitment data object (not zero and not transmitted, different for each entity) is integrated for getting an authentication code (different for each entity).

The second command-response pair exchanges dynamic authentication templates nesting authentication codes with tag '84'.

| | **Exponential** |
|---|---|
| Command data field | {'60'-L-{'A0'-L-Identification Data Template}-{'85'-L-Exponential}} |
| Response data field | {'60'-L-{'A0'-L-Identification Data Template}-{'85'-L-Exponential}} |
| | **Commitment** |
| Command data field | {'60'-L-{'84'-L-Authentication code}} |
| Response data field | {'60'-L-{'84'-L-Authentication code}} |
| | **Challenge** |
| Command data field | {'60'-L-{'81'-L-Challenge}} |
| Response data field | {'60'-L-{'81'-L-Challenge}} |
| | **Response** |
| Command data field | {'60'-L-{'82'-L-Response}} |
| Response data field | {'60'-L-{'82'-L-Response}} if the response is correct<br>Absent if the response is incorrect |