EMV ‘96
Integrated Circuit Card Specification for Payment Systems

Version 3.1.1
May 31, 1998

Table of Contents

Tables

Figures

1. Scope

The *Integrated Circuit Card (ICC) Specification for Payment Systems* describes the minimum functionality required of integrated circuit cards (ICCs) and terminals to ensure correct operation and interoperability. Additional proprietary functionality and features may be provided, but these are beyond the scope of this specification and interoperability cannot be guaranteed.

This specification consists of four parts:

Part I - *Electromechanical Characteristics, Logical Interface, and Transmission Protocols*
Part II - *Data Elements and Commands*
Part III - *Application Selection*
Part IV - *Security Aspects*

Part 1 defines electromechanical characteristics, logical interface, and transmission protocols as they apply to the exchange of information between an ICC and a terminal. In particular it covers:

- Mechanical characteristics, voltage levels, and signal parameters as they apply to both ICCs and terminals.
- An overview of the card session.
- Establishment of communication between the ICC and the terminal by means of the answer to reset.
- Character- and block-oriented asynchronous transmission protocols.

Part II defines data elements and commands as they apply to the exchange of information between an ICC and a terminal. In particular it covers:

- Data elements for financial interchange and their mapping onto data objects.
- Structure and referencing of files.
- Structure and coding of messages between the ICC and the terminal to achieve application level functions.

Part III defines the application selection process from the standpoint of both the card and the terminal. The logical structure of data and files within the card that is required for the process is specified, as is the terminal logic using the card structure.

Part IV defines the security aspects of the processes specified in this specification. In particular it covers:

- Offline static data authentication.

- Offline dynamic data authentication.
- Offline PIN encipherment
- Secure messaging.

This specification does not cover the details of Transaction Certificate generation by the ICC, the internal implementation in the ICC, its security architecture, and its personalisation.

This specification is based on the ISO/IEC 7816 series of standards and should be read in conjunction with those standards. However, if any of the provisions or definitions in this specification differ from those standards, the provisions herein shall take precedence.

This specification is intended for a target audience that includes manufacturers of ICCs and terminals, system designers in payment systems, and financial institution staff responsible for implementing financial applications in ICCs.

1.1 EMV Specification Version Numbering

To facilitate future reference of the EMV specifications and to differentiate between technical updates and editorial clarifications, with the publication of this version of the EMV specifications, EMV has introduced the following version numbering scheme:

version X.Y.Z,

where:

X indicates the phase number of the specifications
Y indicates technical change(s) from the previous version
Z indicates editorial change(s) from the previous version

Therefore, this version of the EMV specifications is version 3.1.1, since the basis for this specification is version 3.0 and both technical and editorial changes have been made.

2. Normative References

The following standards contain provisions that are referenced in this specification.

Europay, MasterCard, and Visa (EMV): March 31, 1998	Integrated Circuit Card Application Specification for Payment Systems
Europay, MasterCard, and Visa (EMV): March 31, 1998	Integrated Circuit Card Terminal Specification for Payment Systems
FIPS Pub 180-1:1995	Secure Hash Standard
IEC 512-2:1979	Specifications for electromechanical components for electromechanical equipment - Part 2: Contact resistance tests, insulation tests, and voltage stress tests
ISO 639:1988	Codes for the representation of names and languages
ISO 3166:1997	Codes for the representation of names of countries
ISO 4217:1995	Codes for the representation of currencies and funds
ISO/IEC 7811-1:1995	Identification cards - Recording technique - Part 1: Embossing
ISO/IEC 7811-3:1995	Identification cards - Recording technique - Part 3: Location of embossed characters on ID-1 cards
ISO/IEC 7813:1995	Identification cards - Financial transaction cards
ISO/IEC DIS 7816-1:1998	Identification cards - Integrated circuit(s) cards with contacts - Part 1: Physical characteristics
ISO/IEC DIS 7816-2:1998	Identification cards - Integrated circuit(s) cards with contacts - Part 2: Dimensions and location of contacts
ISO/IEC 7816-3:1989	Identification cards - Integrated circuit(s) cards with contacts - Part 3: Electronic signals and transmission protocols
ISO/IEC 7816-3:1992	Identification cards - Integrated circuit(s) cards with contacts - Part 3, Amendment 1: Protocol type T=1, asynchronous half duplex block transmission protocol

ISO/IEC 7816-3:1994	Identification cards - Integrated circuit(s) cards with contacts - Part 3, Amendment 2: Protocol type selection (Draft International Standard)
ISO/IEC 7816-4:1995	Identification cards - Integrated circuit(s) cards with contacts - Part 4, Inter-industry commands for interchange
ISO/IEC 7816-5:1994	Identification cards - Integrated circuit(s) cards with contacts - Part 5: Numbering system and registration procedure for application identifiers
ISO/IEC 7816-6:1996	Identification cards - Integrated circuit(s) cards with contacts - Part 6: Inter-industry data elements (Draft International Standard)
ISO 8731-1:1987	Banking - Approved algorithms for message authentication - Part 1: DEA
ISO 8372:1987	Information processing - Modes of operation for a 64-bit block cipher algorithm
ISO/IEC 8825:1990	Information technology - Open systems interconnection - Specification of basic encoding rules for abstract syntax notation one (ASN.1)
ISO 8583:1987	Bank card originated messages - Interchange message specifications - Content for financial transactions
ISO 8583:1993	Financial transaction card originated messages - Interchange message specifications
ISO 8859:1987	Information processing - 8-bit single-byte coded graphic character sets
ISO/IEC 9796-2: 1997	Information technology - Security techniques - Digital signature scheme giving message recovery - Part 2: Mechanism using a hash function
ISO/IEC 9797:1994	Information technology - Security techniques - Data integrity mechanism using a cryptographic check function employing a block cipher algorithm
ISO/IEC 10116: 1997	Information technology - Modes of operation of an n-bit block cipher algorithm
ISO/IEC 10118-3: 1998	Information technology - Security techniques - Hash functions - Part 3: Dedicated hash functions

ISO/IEC 10373:1993 Identification cards - Test methods

3. Definitions

The following terms are used in this specification.

Application - The application protocol between the card and the terminal and its related set of data.

Asymmetric Cryptographic Technique - A cryptographic technique that uses two related transformations, a public transformation (defined by the public key) and a private transformation (defined by the private key). The two transformation have the property that, given the public transformation, it is computationally infeasible to derive the private transformation.

Block - A succession of characters comprising two or three fields defined as prologue field, information field, and epilogue field.

Byte - 8 bits.

Card - A payment card as defined by a payment system.

Certification Authority - Trusted third party that establishes a proof that links a public key and other relevant information to its owner.

Ciphertext - Enciphered information.

Cold Reset - The reset of the ICC that occurs when the supply voltage (VCC) and other signals to the ICC are raised from the inactive state and the reset (RST) signal is applied.

Command - A message sent by the terminal to the ICC that initiates an action and solicits a response from the ICC.

Concatenation - Two elements are concatenated by appending the bytes from the second element to the end of the first. Bytes from each element are represented in the resulting string in the same sequence in which they were presented to the terminal by the ICC, that is, most significant byte first. Within each byte bits are ordered from most significant bit to least significant. A list of elements or objects may be concatenated by concatenating the first pair to form a new element, using that as the first element to concatenate with the next in the list, and so on.

Contact - A conducting element ensuring galvanic continuity between integrated circuit(s) and external interfacing equipment.

Cryptogram - Result of a cryptographic operation.

Cryptographic Algorithm - An algorithm that transforms data in order to hide or reveal its information content.

Data Integrity - The property that data has not been altered or destroyed in an unauthorised manner

Decipherment - The reversal of a corresponding encipherment

Digital Signature - An asymmetric cryptographic transformation of data that allows the recipient of the data to prove the origin and integrity of the data, and protect the sender and the recipient of the data against forgery by third parties, and the sender against forgery by the recipient.

Embossing - Characters raised in relief from the front surface of a card.

Encipherment - The reversible transformation of data by a cryptographic algorithm to produce ciphertext.

Epilogue Field - The final field of a block. It contains the error detection code (EDC) byte(s).

Financial Transaction - The act between a cardholder and a merchant or acquirer that results in the exchange of goods or services against payment.

Function - A process accomplished by one or more commands and resultant actions that are used to perform all or part of a transaction.

Guardtime - The minimum time between the trailing edge of the parity bit of a character and the leading edge of the start bit of the following character sent in the same direction.

Hash Function - A function that maps strings of bits to fixed-length strings of bits, satisfying the following two properties:

- It is computationally infeasible to find for a given output an input which maps to this output.
- It is computationally infeasible to find for a given input a second input that maps to the same output.

Additionally, if the hash function is required to be collision-resistant, it must also satisfy the following property:

- It is computationally infeasible to find any two distinct inputs that map to the same output.

Hash Result - The string of bits that is the output of a hash function.

Inactive - The supply voltage (VCC) and other signals to the ICC are in the inactive state when they are at a potential of 0.4 V or less with respect to ground (GND).

Integrated Circuit(s) - Electronic component(s) designed to perform processing and/or memory functions.

Integrated Circuit(s) Card - A card into which one or more integrated circuits are inserted to perform processing and memory functions.

Integrated Circuit Module - The sub-assembly embedded into the ICC comprising the IC, the IC carrier, bonding wires, and contacts.

Interface Device - That part of a terminal into which the ICC is inserted, including such mechanical and electrical devices that may be considered part of it.

Key - A sequence of symbols that controls the operation of a cryptographic transformation.

Magnetic Stripe - The stripe containing magnetically encoded information.

Message - A string of bytes sent by the terminal to the card or vice versa, excluding transmission-control characters.

Message Authentication Code - A symmetric cryptographic transformation of data that protects the sender and the recipient of the data against forgery by third parties.

Nibble - The four most significant or least significant bits of a byte.

Padding - Appending extra bits to either side of a data string.

Path - Concatenation of file identifiers without delimitation.

Payment System - For the purposes of this specification, Europay International S.A., MasterCard International Incorporated, or Visa International Service Association.

Payment Systems Environment - The set of logical conditions established within the ICC when a payment system application conforming to this specification has been selected, or when a directory definition file (DDF) used for payment system application purposes has been selected.

Plaintext - Unenciphered information.

Private Key - That key of an entity's asymmetric key pair that should only be used by that entity. In the case of a digital signature scheme, the private key defines the signature function.

Prologue Field - The first field of a block. It contains subfields for node address (AD), protocol control byte (PCB), and length (LEN).

Public Key - That key of an entity's asymmetric key pair that can be made public. In the case of a digital signature scheme, the public key defines the verification function.

Public Key Certificate - The public key information of an entity signed by the certification authority and thereby rendered unforgeable.

Redundancy - Any information that is known and can be checked.

Response - A message returned by the ICC to the terminal after the processing of a command message received by the ICC.

Secret Key - A key used with symmetric cryptographic techniques and usable only by a set of specified entities.

Script - A command or a string of commands transmitted by the issuer to the terminal for the purpose of being sent serially to the ICC as commands.

Signal Amplitude - The difference between the high and low voltages of a signal.

Signal Perturbations - Any abnormal conditions occurring on a signal such as undershoot/overshoot, electrical noise, ripple, spikes, crosstalk, etc.

State H - Voltage high on a signal line. May indicate a logic one or logic zero depending on the logic convention used with the ICC.

State L - Voltage low on a signal line. May indicate a logic one or logic zero depending on the logic convention used with the ICC.

Symmetric Cryptographic Technique - A cryptographic technique that uses the same secret key for both the originator's and recipient's transformation. Without knowledge of the secret key, it is computationally infeasible to compute either the originator's or the recipient's transformation.

T=0 - Character-oriented asynchronous half duplex transmission protocol.

T=1 - Block-oriented asynchronous half duplex transmission protocol.

Template - Value field of a constructed data object, defined to give a logical grouping of data objects.

Terminal - The device used in conjunction with the ICC at the point of transaction to perform a financial transaction. It incorporates the interface device and may also include other components and interfaces such as host communications.

Warm Reset - The reset that occurs when the reset (RST) signal is applied to the ICC while the clock (CLK) and supply voltage (VCC) lines are maintained in their active state.

4. Abbreviations and Notations

The following abbreviations and notations are used in this specification.

AAC	Application Authentication Cryptogram
AAR	Application Authorisation Referral
AC	Application Cryptogram
ACK	Acknowledgment
ADF	Application Definition File
AEF	Application Elementary File
AFL	Application File Locator
AID	Application Identifier
an	Alphanumeric
ans	Alphanumeric Special
APDU	Application Protocol Data Unit
ARPC	Authorisation Response Cryptogram
ARQC	Authorisation Request Cryptogram
ASN	Abstract Syntax Notation
ATC	Application Transaction Counter
ATR	Answer to Reset
b	Binary
BER	Basic Encoding Rules
BGT	Block Guardtime
BWI	Block Waiting Time Integer
BWT	Block Waiting Time
C	Celsius or Centigrade
C-APDU	Command APDU

CBC	Cipher Block Chaining
CDOL	Card Risk Management Data Object List
C_{IN}	Input Capacitance
CLA	Class Byte of the Command Message
CLK	Clock
cn	Compressed Numeric
C-TPDU	Command TPDU
CVM	Cardholder Verification Method
CWI	Character Waiting Time Integer
CWT	Character Waiting Time
DAD	Destination Node Address
DC	Direct Current
DDF	Directory Definition File
DDOL	Dynamic Data Authentication Data Object List
DES	Data Encryption Standard
DF	Dedicated File
DIR	Directory
DIS	Draft International Standard
ECB	Electronic Code Book
EDC	Error Detection Code
EF	Elementary File
etu	Elementary Time Unit
FCI	File Control Information
f	Frequency
FIPS	Federal Information Processing Standard

GND	Ground
hex.	Hexadecimal
HHMM	Hours, Minutes
HHMMSS	Hours, Minutes, Seconds
I-block	Information Block
IC	Integrated Circuit
ICC	Integrated Circuit Card
IEC	International Electrotechnical Commission
IFD	Interface Device
IFS	Information Field Size
IFSC	Information Field Size for the ICC
IFSD	Information Field Size for the Terminal
IFSI	Information Field Size Integer
I_{IH}	High Level Input Current
I_{IL}	Low Level Input Current
INF	Information Field
INS	Instruction Byte of Command Message
I/O	Input/Output
I_{OH}	High Level Output Current
I_{OL}	Low Level Output Current
ISO	International Organisation for Standardisation
K_M	Master Key
K_S	Session Key
kΩ	Kilohm
Lc	Exact Length of Data Sent by the TAL in a Case 3 or 4 Command

lcm	Least Common Multiple
L_{DD}	Length of the ICC Dynamic Data
Le	Maximum Length of Data Expected by the TAL in Response to a Case 2 or 4 Command
Licc	Exact Length of Data Available (or Remaining) in the ICC to be Returned in Response to the Case 2 or 4 Command Received by the ICC
LEN	Length
Lr	Length of Response Data Field
LRC	Longitudinal Redundancy Check
M	Mandatory
µm	Micrometre
mA	Milliampere
MAC	Message Authentication Code
max.	Maximum
MF	Master File
MHz	Megahertz
min.	Minimum
mm	Millimetre
mΩ	Milliohm
m/s	Meters per Second
µA	Microampere
µs	Microsecond
N	Newton
n	Numeric
NAD	Node Address

NAK	Negative Acknowledgment
nAs	Nanoampere-second
N_{CA}	Length of the Certification Authority Public Key Modulus
N_I	Length of the Issuer Public Key Modulus
N_{IC}	Length of the ICC Public Key Modulus
N_{PE}	Length of the ICC PIN Encipherment Public Key Modulus
ns	Nanosecond
O	Optional
P1	Parameter 1
P2	Parameter 2
P3	Parameter 3
PAN	Primary Account Number
P_{CA}	Certification Authority Public Key
PCB	Protocol Control Byte
PDOL	Processing Options Data Object List
pF	Picofarad
P_I	Issuer Public Key
P_{IC}	ICC Public Key
PIN	Personal Identification Number
PSA	Payment System Application
PSE	Payment System Environment
PTS	Protocol Type Selection
R-APDU	Response APDU
R-block	Receive Ready Block
RFU	Reserved for Future Use

RID	Registered Application Provider Identifier
RSA	Rivest, Shamir, Adleman Algorithm
RST	Reset
R-TPDU	Response TPDU
SAD	Source Node Address
S-block	Supervisory Block
S_{CA}	Certification Authority Private Key
S_I	Issuer Private Key
S_{IC}	ICC Private Key
SFI	Short File Identifier
SHA	Secure Hash Algorithm
SW1	Status Word One
SW2	Status Word Two
TAL	Terminal Application Layer
TC	Transaction Certificate
TCK	Check Character
TDOL	Transaction Certificate Data Object List
t_F	Fall Time Between 90% and 10% of Signal Amplitude
TLV	Tag Length Value
TPDU	Transport Protocol Data Unit
t_R	Rise Time Between 10% and 90% of Signal Amplitude
TTL	Terminal Transport Layer
TVR	Terminal Verification Results
V	Volt
var.	Variable

V_{CC}	Voltage Measured on VCC Contact
VCC	Supply Voltage
V_{IH}	High Level Input Voltage
V_{IL}	Low Level Input Voltage
V_{OH}	High Level Output Voltage
V_{OL}	Low Level Output Voltage
VPP	Programming Voltage
WI	Waiting Time Integer
WTX	Waiting Time Extension
YYMMDD	Year, Month, Day

The following notations apply:

'0' to '9' and 'A' to 'F'	16 hexadecimal digits
#	Number
[...]	Optional part
A := B	A is assigned the value of B
A = B	Value of A is equal to the value of B
$A \equiv B \bmod n$	Integers A and B are congruent modulo the integer n, that is, there exists an integer d such that $$(A - B) = dn$$
A mod n	The reduction of the integer A modulo the integer n, that is, the unique integer $0 \leq r < n$ for which there exists an integer d such that $$A = dn + r$$
abs(n)	Absolute value of an integer n defined as n if $n \geq 0$, and as $-n$ if $n < 0$

$Y := ALG(K)[X]$	Encipherment of a 64-bit data block X with a 64-bit block cipher as specified in Annex E1 using a secret key K
$X = ALG^{-1}(K)[Y]$	Decipherment of a 64-bit data block Y with a 64-bit block cipher as specified in Annex E1 using a secret key K
$Y := Sign(S_K)[X]$	The signing of a data block X with an asymmetric reversible algorithm as specified in Annex E2, using the private key S_K
$X = Recover(P_K)[Y]$	The recovery of the data block X with an asymmetric reversible algorithm as specified in Annex E2, using the public key P_K
$C := (A \mid\mid B)$	The concatenation of an n-bit number A and an m-bit number B, which is defined as $C = 2^m A + B$.
$H := Hash[MSG]$	Hashing of a message MSG of arbitrary length using an 80-bit hash function
lcm(a, b)	Least common multiple of two integers a and b
$\lvert n \rvert$	Length of an integer n in bits
$(X \mid n)$	The Jacobi symbol of an integer X with respect to an integer n = pq consisting of the product of two primes p and q, and which is defined as follows. Define $$J := (X^{(p-1)/2} \bmod p)(X^{(q-1)/2} \bmod q)$$ If $J = 1$ or $J = (pq - p - q + 1)$, then $(X \mid n) := 1$. Otherwise, $(X \mid n) := -1$. Note that the Jacobi symbol can efficiently be computed without the prime factors of n (for example, see Informative Reference [5] in Annex G).
xx	Any value

THIS PAGE LEFT INTENTIONALLY BLANK

Part I

Electromechanical Characteristics, Logical Interface, and Transmission Protocols

1. Electromechanical Interface

This section covers the electrical and mechanical characteristics of the ICC and the terminal. ICC and terminal specifications differ to allow a safety margin to prevent damage to the ICC.

The ICC characteristics defined herein are based on the ISO/IEC 7816 series of standards with some small variations.

1.1 Mechanical Characteristics of the ICC

This section describes the physical characteristics, contact assignment, and mechanical strength of the ICC.

1.1.1 Physical Characteristics

Except as otherwise specified herein, the ICC shall comply with the physical characteristics for ICCs as defined in ISO/IEC DIS 7816-1. The ICC shall also comply with the additional characteristics defined in ISO/IEC DIS 7816-1 as related to ultra-violet light, X-rays, surface profile of the contacts, mechanical strength, electromagnetic characteristics, and static electricity and shall continue to function correctly electrically under the conditions defined therein.

1.1.1.1 Module Height

The highest point on the IC module surface shall not be greater than 0.05mm above the plane of the card surface.

The lowest point on the IC module surface shall not be greater than 0.10mm below the plane of the card surface.

1.1.2 Dimensions and Location of Contacts

The dimensions and location of the contacts shall be as shown in Figure I-1:

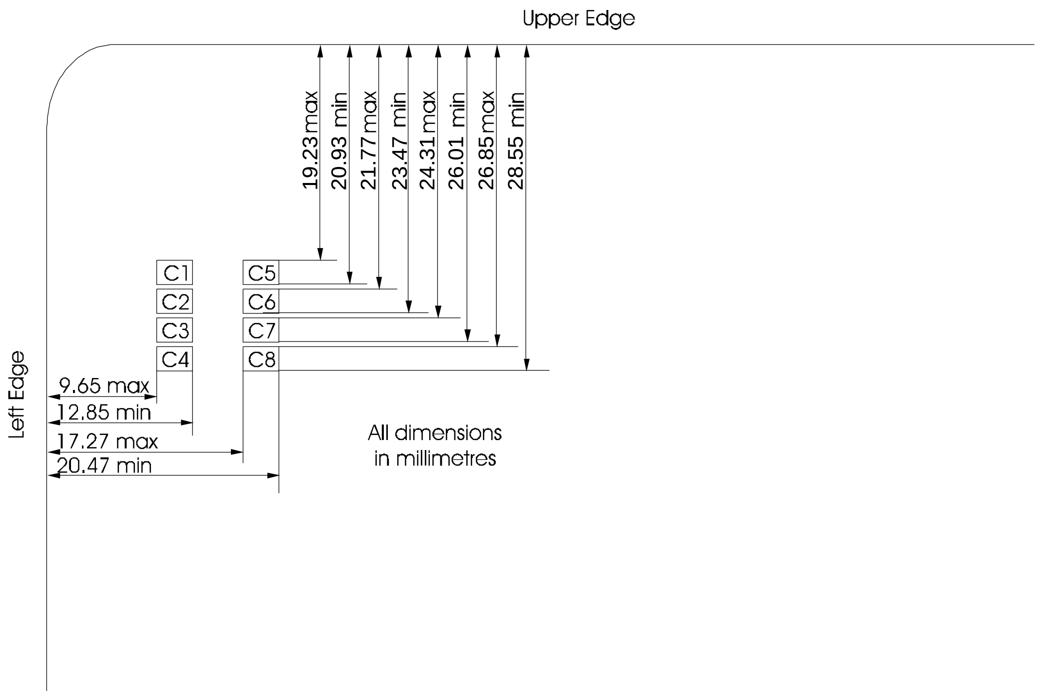

Figure I-1 - ICC Contact Location and Dimensions

It is recommended that the metallised contact areas be larger than the minimum specified wherever possible.

The layout of the contacts relative to embossing and/or magnetic stripe shall be as shown in Figure I-2:

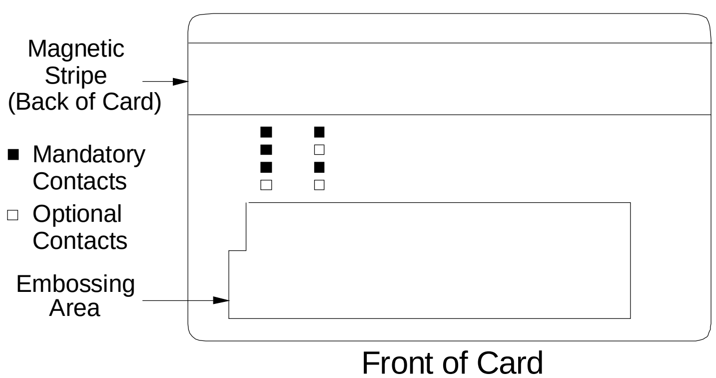

Figure I-2 - Layout of Contacts

1.1.3 Contact Assignment

The assignment of the ICC contacts shall be as defined in ISO/IEC DIS 7816-2 and is shown in Table I-1:

C1	Supply voltage (VCC)	C5	Ground (GND)
C2	Reset (RST)	C6	Not used[1]
C3	Clock (CLK)	C7	Input/output (I/O)

Table I-1 - ICC Contact Assignment

C4 and C8 are not used and need not be physically present. C6 is not used and need not be physically present; if present, it shall be electrically isolated[2] from the integrated circuit (IC) itself and other contacts on the ICC.

1.2 Electrical Characteristics of the ICC

This section describes the electrical characteristics of the signals as measured at the ICC contacts.

1.2.1 Measurement Conventions

All measurements are made at the point of contact between the ICC and the interface device (IFD) contacts and are defined with respect to the GND contact over an ambient temperature range 0° C to 50° C.

All currents flowing into the ICC are considered positive.

Note: The temperature range limits are dictated primarily by the thermal characteristics of polyvinyl chloride (that is used for the majority of cards that are embossed) rather than by constraints imposed by the characteristics of the IC.

1.2.2 Input/Output (I/O)

This contact is used as an input (reception mode) to receive data from the terminal or as an output (transmission mode) to transmit data to the terminal. During operation, the ICC and the terminal shall not both be in transmit mode. In the event that this condition occurs, the state (voltage level) of the I/O contact is indeterminate and no damage shall occur to the ICC.

1.2.2.1 Reception Mode

When in reception mode, and with the supply voltage (VCC) in the range specified in section I-1.2.6, the ICC shall correctly interpret signals from the terminal having the characteristics shown in Table I-2:

---

[1] Defined in ISO/IEC 7816 as programming voltage (VPP).

[2] Electrically isolated means that the resistance measured between C6 and any other contact shall be ≥10MΩ with an applied voltage of 5V DC.

Symbol	Minimum	Maximum	Unit
V_{IH}	0.7 x V_{CC}	V_{CC}	V
V_{IL}	0	0.8	V
t_R and t_F	-	1.0	µs

Table I-2 - Electrical Characteristics of I/O for ICC Reception

Note: The ICC shall not be damaged by signal perturbations on the I/O line in the range –0.3 V to V_{CC} + 0.3 V.

1.2.2.2 Transmission Mode

When in transmission mode, the ICC shall send data to the terminal with the characteristics shown in Table I-3:

Symbol	Conditions	Minimum	Maximum	Unit
V_{OH}	–20 µA < I_{OH} < 0, V_{CC} = min.	0.7 x V_{CC}	V_{CC}	V
V_{OL}	0 < I_{OL} < 1 mA, V_{CC} = min.	0	0.4	V
t_R and t_F	C_{IN} (terminal) = 30 pF max.	-	1.0	µs

Table I-3 - Electrical Characteristics of I/O for ICC Transmission

Unless transmitting, the ICC shall set its I/O line driver to reception mode. There is no requirement for the ICC to have any current source capability from I/O.

1.2.3 Programming Voltage (VPP)

The ICC shall not require VPP (see note in section I-1.3.3).

1.2.4 Clock (CLK)

With VCC in the range specified in section I-1.2.6, the ICC shall operate correctly with a CLK signal having the characteristics shown in Table I-4:

Symbol	Conditions	Minimum	Maximum	Unit
V_{IH}		V_{CC} – 0.7	V_{CC}	V
V_{IL}		0	0.5	V
t_R and t_F	V_{CC} = min. to max.	-	9% of clock period	

Table I-4 - Electrical Characteristics of CLK to ICC

Note: The ICC shall not be damaged by signal perturbations on the CLK line in the range –0.3 V to V_{CC} + 0.3 V.

The ICC shall operate correctly with a CLK duty cycle of between 44% and 56% of the period during stable operation.

The ICC shall operate correctly with a CLK frequency in the range 1 MHz to 5 MHz.

Note: Frequency shall be maintained by the terminal to within ± 1% of that used during the answer to reset throughout the card session.

1.2.5 Reset (RST)

With VCC in the range specified in section I-1.2.6, the ICC shall correctly interpret a RST signal having the characteristics shown in Table I-5:

Symbol	Conditions	Minimum	Maximum	Unit
V_{IH}		$V_{CC} - 0.7$	V_{CC}	V
V_{IL}		0	0.6	V
t_R and t_F	V_{CC} = min. to max.	-	1.0	µs

Table I-5 - Electrical Characteristics of RST to ICC

Note: The ICC shall not be damaged by signal perturbations on the RST line in the range –0.3 V to V_{CC} + 0.3 V.

The ICC shall answer to reset asynchronously using active low reset.

1.2.6 Supply Voltage (VCC)

The ICC shall operate correctly with a supply voltage V_{CC} of 5 V ± 0.5 V DC and have a maximum current requirement of 50 mA when operating at any frequency within the range specified in section I-1.2.4.

Note: It is strongly recommended that the current consumption of ICCs is maintained at as low a value as possible, since the maximum current consumption allowable for the ICC may be reduced in future versions of this specification. Issuers of ICCs bearing multisector applications should ensure that the IC used has a current requirement compatible with all terminals (from all sectors) in which the ICC might be used.

1.2.7 Contact Resistance

The contact resistance as measured across a pair of clean ICC and clean nominal IFD contacts shall be less than 500 mΩ throughout the design life of an ICC (see ISO/IEC 10373 for test method).

Note: A nominal IFD contact may be taken as a minimum of 1.25 µm of gold over 5.00 µm of nickel.

1.3 Mechanical Characteristics of the Terminal

This section describes the mechanical characteristics of the terminal interface device.

1.3.1 Interface Device

The IFD into which the ICC is inserted shall be capable of accepting ICCs having the following characteristics:

- Physical characteristics compliant with ISO/IEC DIS 7816-1
- Contacts on the front, in the position compliant with Figure 2 of ISO/IEC DIS 7816-2
- Embossing compliant with ISO/IEC 7811-1 and 3

The IFD contacts shall be located such that if an ICC having contacts with the dimensions and locations specified in Figure I-3 is inserted into the IFD, correct connection of all contacts shall be made.

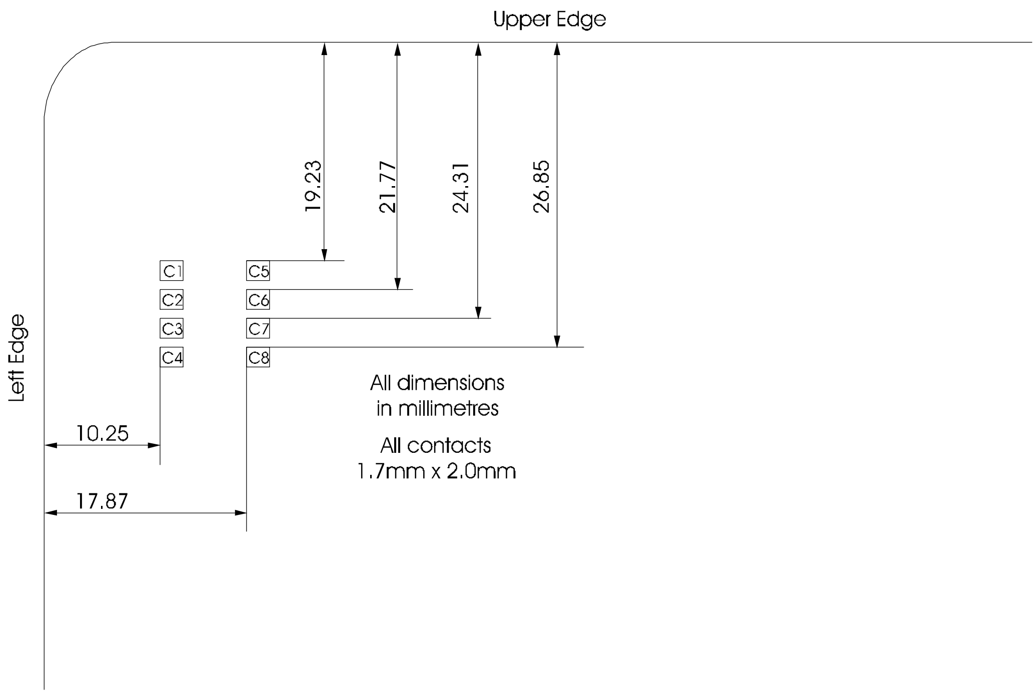

Figure I-3 - Terminal Contact Location and Dimensions

Location guides and clamps (if used) shall cause no damage to ICCs, particularly in the areas of the magnetic stripe, signature panel, embossing, and hologram.

Note: As a general principle, an ICC should be accessible to the cardholder at all times. Where the ICC is drawn into the IFD, a mechanism should exist to return the ICC to the cardholder in the event of a failure (for example, loss of power).

1.3.2 Contact Forces

The force exerted by any one IFD contact on the corresponding ICC contact shall be in the range 0.2 N to 0.6 N.

1.3.3 Contact Assignment

The assignment of the IFD contacts shall be as shown in Table I-6:

C1	VCC	C5	GND
C2	RST	C6	Not used[3]
C3	CLK	C7	I/O

Table I-6 - IFD Contact Assignment

C4 and C8 are not used and need not be physically present. C6 shall be electrically isolated[4].

Note: If connected in existing terminals, C6 shall be maintained at a potential between GND and 1.05 x V_{CC} throughout the card session. Keeping C6 isolated in new terminals facilitates its use for other purposes if so defined in future versions of this specification.

1.4 Electrical Characteristics of the Terminal

This section describes the electrical characteristics of the signals as measured at the IFD contacts.

1.4.1 Measurement Conventions

All measurements are made at the point of contact between the ICC and the IFD contacts and are defined with respect to GND contact over an ambient temperature range 0° C to 50° C.

All currents flowing out of the terminal are considered positive.

1.4.2 Input/Output (I/O)

This contact is used as an output (transmission mode) to transmit data to the ICC or as an input (reception mode) to receive data from the ICC. During operation, the terminal and the ICC shall not both be in transmit mode. In the event that this

[3] Defined in ISO/IEC 7816 as programming voltage (VPP).

[4] Electrically isolated means that the resistance measured between C6 and any other contact shall be ≥10MΩ with an applied voltage of 5V DC.

condition occurs, the state (voltage level) of the contact is indeterminate and no damage shall occur to the terminal.

When both the terminal and the ICC are in reception mode, the contact shall be in the high state. To achieve this, the terminal shall incorporate a pull-up resistor to VCC, or other device. The terminal shall not pull I/O high unless VCC is powered and stable within the tolerances specified in section I-1.4.6. See the contact activation sequence specified in section I-2.1.2.

The terminal shall limit the current flowing into or out of the I/O contact to ±15 mA at all times.

1.4.2.1 Transmission Mode

When in transmission mode, the terminal shall send data to the ICC with the characteristics shown in Table I-7:

Symbol	Conditions	Minimum	Maximum	Unit
V_{OH}	$0 < I_{OH} < 20$ µA, V_{CC} = min.	0.8 x V_{CC}	V_{CC}	V
V_{OL}	- 0.5 mA < I_{OL} < 0, V_{CC} = min.	0	0.4	V
t_R and t_F	$C_{IN(ICC)}$ = 30 pF max.	-	0.8	µs
Signal perturbations	Signal low	- 0.25	0.4	V
	Signal high	0.8 x V_{CC}	V_{CC} + 0.25	V

Table I-7 - Electrical Characteristics of I/O for Terminal Transmission

Unless transmitting, the terminal shall set its I/O line driver to reception mode.

1.4.2.2 Reception Mode

When in reception mode, the terminal shall correctly interpret signals from the ICC having the characteristics shown in Table I-8:

Symbol	Minimum	Maximum	Unit
V_{IH}	0.6 x V_{CC}	V_{CC}	V
V_{IL}	0	0.5	V
t_R and t_F	-	1.2	µs

Table I-8 - Electrical Characteristics of I/O for Terminal Reception

1.4.3 Programming Voltage (VPP)

The terminal shall not generate a VPP (see section I-1.3.3).

1.4.4 Clock (CLK)

The terminal shall generate a CLK signal having the characteristics shown in Table I-9:

Symbol	Conditions	Minimum	Maximum	Unit
V_{OH}	$0 < I_{OH} < 50$ µA, V_{CC} = min.	$V_{CC} - 0.5$	V_{CC}	V
V_{OL}	- 50 µA $< I_{OL} < 0$, V_{CC} = min.	0	0.4	V
t_R and t_F	$C_{IN(ICC)}$ = 30 pF max.	-	8% of clock period	
Signal perturbations	Signal low	- 0.25	0.4	V
	Signal high	$V_{CC} - 0.5$	$V_{CC} + 0.25$	V

Table I-9 - Electrical Characteristics of CLK from Terminal

Duty cycle shall be between 45% and 55% of the period during stable operation.

Frequency shall be in the range 1 MHz to 5 MHz and shall not change by more than ± 1% throughout a card session (see section I-2).

1.4.5 Reset (RST)

The terminal shall generate a RST signal having the characteristics shown in Table I-10:

Symbol	Conditions	Minimum	Maximum	Unit
V_{OH}	$0 < I_{OH} < 50$ µA, V_{CC} = min.	V_{CC} - 0.5	V_{CC}	V
V_{OL}	- 50 µA $< I_{OL} < 0$, V_{CC} = min.	0	0.4	V
T_R and t_F	$C_{IN(ICC)}$ = 30 pF max.	-	0.8	µs
Signal perturbations	Signal low	- 0.25	0.4	V
	Signal high	$V_{CC} - 0.5$	$V_{CC} + 0.25$	V

Table I-10 - Electrical Characteristics of RST from Terminal

1.4.6 Supply Voltage (VCC)

The terminal shall generate a V_{CC} of 5 V ± 0.4 V DC and shall be capable of delivering steady state output current in the range 0 to 55 mA whilst maintaining V_{CC} within these tolerances. The terminal shall contain protection circuitry to prevent damage occurring to it in the event of fault conditions such as a short circuit to GND or VCC. This supply shall be protected from transients and surges caused by internal operation of the terminal and from external interference introduced via power leads, communications links, etc. V_{CC} shall never be negative with respect to ground.

During normal operation of an ICC, current pulses cause voltage transients on VCC as measured at the ICC contacts. The power supply shall be able to counteract transients in the current consumption of the ICC up to a maximum charge of 40 nAs with no more than 400 ns duration and a maximum amplitude of 100 mA, ensuring that V_{CC} remains within the range specified.

Note: Terminals may be designed to be capable of delivering more than 55 mA if required, but it is recommended that terminals limit the steady state current that can be delivered to a maximum of 200 mA.

1.4.7 Contact Resistance

The contact resistance as measured across a pair of clean IFD and clean nominal ICC contacts shall be less than 500 mΩ throughout the design life of a terminal (see ISO/IEC DIS 7816-1 for test method).

Note: A nominal ICC contact may be taken as 1.25 µm of gold over 5.00 µm of nickel.

1.4.8 Short Circuit Resilience

The terminal shall be capable of sustaining a short circuit of any duration between any or all contacts without suffering damage or malfunction, for example, if a metal plate or an ICC with a metallic surface is inserted.

1.4.9 Powering and Depowering of Terminal with ICC in Place

If the terminal is powered on or off with an ICC in place no spurious signals or power perturbations shall appear at the interface contacts. Contact activation and deactivation sequences and timings, as described in sections I-2.1.2 and I-2.1.5 respectively shall be respected.

2. Card Session

This section describes all stages involved in a card session from insertion of the ICC into the IFD through the execution of the transaction to the removal of the ICC from the IFD.

2.1 Normal Card Session

This section describes the processes involved in the execution of a normal transaction.

2.1.1 Stages of a Card Session

A card session is comprised of the following stages:

1. Insertion of the ICC into the IFD and connection and activation of the contacts.

2. Reset of the ICC and establishment of communication between the terminal and the ICC.

3. Execution of the transaction(s).

4. Deactivation of the contacts and removal of the ICC.

2.1.2 ICC Insertion and Contact Activation Sequence

On insertion of the ICC into the IFD, the terminal shall ensure that all signal contacts are in state L with values of V_{OL} as defined in section I-1.4 and that V_{CC} is 0.4 V or less before any contacts are physically made. The IFD shall be able to detect when the ICC is seated to within ±0.5 mm of the nominally correct position[5] in the direction of insertion/withdrawal. When the IFD detects that the ICC is seated within this tolerance, and when all contacts have been physically made, the contacts shall be activated as follows (see Figure I-4):

- RST shall be maintained by the terminal in state L throughout the activation sequence.

- Following establishment of the physical contacts but prior to activation of I/O or CLK, VCC shall be powered.

- Following verification by the terminal that V_{CC} is stable and within the limits defined in section I-1.4.6, the terminal shall set its I/O line driver to reception mode and shall provide CLK with a suitable and stable clock as defined in section I-1.4.4. The I/O line driver in the terminal may be set to reception mode prior to

[5] The 'nominally correct position' is when the centres of the IFD contacts are exactly over the centres of the ICC contacts located as specified in ISO/IEC DIS 7816-2.

application of the clock but shall be set to reception mode no later than 200 clock cycles after application of the clock.

Note: The terminal may verify the state of Vcc by measurement, by waiting sufficient time for it to stabilise according to the design of the terminal, or otherwise. The state of the I/O line after the terminal has set its I/O line driver to reception mode is dependent upon the state of the I/O line driver in the ICC (see section I-2.1.3.1).

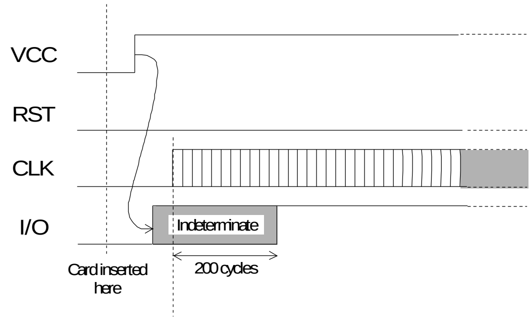

Figure I-4 - Contact Activation Sequence

2.1.3 ICC Reset

The ICC shall answer to reset asynchronously using active low reset.

The means of transportation of the answer to reset (ATR) are described in section I-3 and its contents are described in sections I-4.2 and I-4.3.

2.1.3.1 Cold Reset

Following activation of the contacts according to section I-2.1.2, the terminal shall initiate a cold reset and obtain an ATR from the ICC as follows (see Figure I-5):

- The terminal shall apply CLK at a notional time T0.
- Within a maximum of 200 clock cycles following T0, the ICC shall set its I/O line driver to reception mode. Since the terminal shall also have set its I/O line driver to reception mode within this period, the I/O line is guaranteed to be in state H no later than 200 clock cycles following time T0.
- The terminal shall maintain RST in state L through time T0 and for a period of between 40,000 and 45,000 clock cycles following time T0 to time T1, when it shall set RST to state H.
- The answer to reset on I/O from the ICC shall begin between 400 and 40,000 clock cycles after time T1 (time *t1* in Figure I-5).
- If the answer to reset from the ICC does not begin within this time, the terminal shall initiate the deactivation sequence described in section I-2.1.5 (hereafter referred to as the ‘deactivation sequence’) within 50 ms.

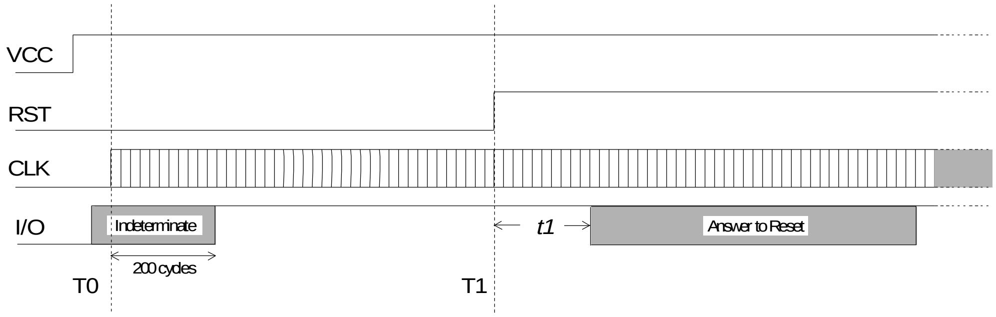

Figure I-5 - Cold Reset Sequence

2.1.3.2 Warm Reset

If the ATR received following a cold reset as described in section I-2.1.3.1 does not conform to the specification in section I-4, the terminal shall initiate a warm reset and obtain an ATR from the ICC as follows (see Figure I-6):

- A warm reset shall start at a notional time T0’, at which time the terminal shall set RST to state L.

- The terminal shall maintain VCC and CLK stable and within the limits defined in sections I-1.4.4 and I-1.4.6 throughout the warm reset sequence.

- Within a maximum of 200 clock cycles following T0’, the ICC and terminal shall set their I/O line drivers to reception mode. The I/O line therefore is guaranteed to be in state H no later than 200 clock cycles following time T0’.

- The terminal shall maintain RST in state L from time T0’ for a period of between 40,000 and 45,000 clock cycles following time T0’ to time T1’, when it shall set RST to state H.

- The answer to reset on I/O from the ICC shall begin between 400 and 40,000 clock cycles after time T1’ (time *t1’* in Figure I-6).

- If the answer to reset from the ICC does not begin within this time, the terminal shall initiate the deactivation sequence within 50 ms.

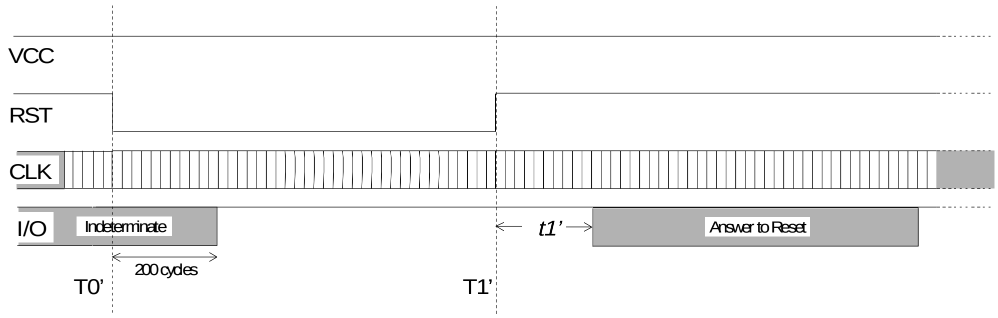

Figure I-6 - Warm Reset Sequence

2.1.4 Execution of a Transaction

Selection of the application in the ICC and the subsequent exchange of information between the ICC and the terminal necessary to perform a transaction are described in Part III of this specification, and in the *ICC Application Specification for Payment Systems.*

2.1.5 Contact Deactivation Sequence

As the final step in the card session, upon normal or abnormal termination of the transaction (including withdrawal of the ICC from the IFD during a card session), the terminal shall deactivate the IFD contacts as follows (see Figure I-7):

- The terminal shall initiate the deactivation sequence by setting RST to state L.
- Following the setting of RST to state L but prior to depowering VCC, the terminal shall set CLK and I/O to state L.
- Following the setting of RST, CLK, and I/O to state L but prior to galvanic disconnection of the IFD contacts, the terminal shall depower VCC. V_{CC} shall be 0.4 V or less prior to galvanic disconnection of the IFD contacts.
- The deactivation sequence shall be completed within 100 ms. This period is measured from the time that RST is set to state L to the time that V_{CC} reaches 0.4 V or less.

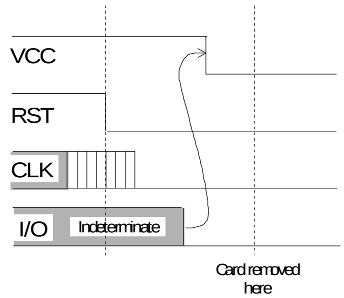

Figure I-7 - Contact Deactivation Sequence

2.2 Abnormal Termination of Transaction Process

If an ICC is prematurely removed from a terminal during execution of a transaction at speeds of up to 1 m/s, the terminal shall be capable of sensing the movement of the ICC relative to the IFD contacts, and of deactivating all IFD contacts in the manner described in section I-2.1.5 before the relative movement exceeds 1 mm. No electrical or mechanical damage shall be caused to the ICC under these conditions.

Note: For 'sliding carriage' type IFDs, it may be possible for the terminal to sense the movement of the ICC/IFD contact sub-assembly relative to the main body of the IFD. In this event, it is not mandatory to be able to sense the movement of the ICC relative to the IFD contacts, but deactivation of the contacts shall be complete before any electrical contact is broken between the ICC and IFD.

3. Physical Transportation of Characters

During the transaction process, data is passed bidirectionally between the terminal and the ICC over the I/O line in an asynchronous half duplex manner. A clock signal is provided to the ICC by the terminal, and this shall be used to control the timing of this exchange. The mechanism of exchanging bits and characters is described below. It applies during the answer to reset and is also used by both transmission protocols as described in section I-5.

3.1 Bit Duration

The bit duration used on the I/O line is defined as an elementary time unit (etu). A linear relationship exists between the etu on the I/O line and CLK frequency (*f*).

During the answer to reset, the bit duration is known as the initial etu, and is given by the following equation:

$$\text{initial etu} = \frac{372}{f} \text{ seconds, where } f \text{ is in Hertz}$$

Following the answer to reset (and establishment of the global parameters F and D, see section I-4), the bit duration is known as the current etu, and is given by the following equation:

$$\text{current etu} = \frac{\text{F}}{\text{D}f} \text{ seconds, where } f \text{ is in Hertz}$$

Note: For the basic answer(s) to reset described in this specification, only values of F = 372 and D = 1 are supported. Thus the initial and current etus are the same and are given by $\frac{372}{f}$. In the following sections of this specification where etu is referred to, it is current etu that is meant unless otherwise specified.

Throughout the card session, *f* shall be in the range 1 MHz to 5 MHz.

3.2 Character Frame

Data is passed over the I/O line in a character frame as described below. The convention used is specified in the initial character (TS) transmitted by the ICC in the ATR (see section I-4.3.1).

Prior to transmission of a character, the I/O line shall be in state H.

A character consists of 10 consecutive bits (see Figure I-8):

- 1 start bit in state L

- 8 bits, which comprise the data byte
- 1 even parity checking bit

The start bit is detected by the receiving end by periodically sampling the I/O line. The sampling time shall be less than or equal to 0.2 etu.

The number of logic ones in a character shall be even. The 8 bits of data and the parity bit itself are included in this check but not the start bit.

The time origin is fixed as midway between the last observation of state H and the first observation of state L. The existence of a start bit shall be verified within 0.7 etu. Subsequent bits shall be received at intervals of (n + 0.5 ± 0.2) etu (n being the rank of the bit). The start bit is bit 1.

Within a character, the time from the leading edge of the start bit to the trailing edge of the nth bit is (n ± 0.2) etu.

The interval between the leading edges of the start bits of two consecutive characters is comprised of the character duration (10 ± 0.2) etu, plus a guardtime. Under error free transmission, during the guardtime both the ICC and the terminal shall be in reception mode (I/O line in state H). For T=0 only, if the ICC or terminal as receiver detects a parity error in a character just received, it shall set I/O to state L to indicate the error to the sender (see section I-5.2.3)

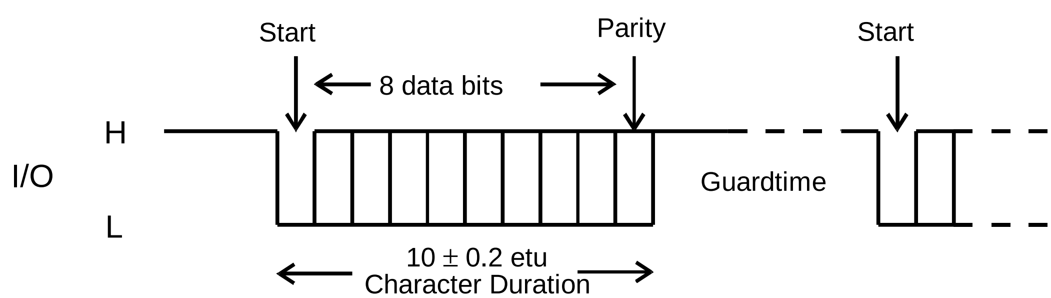

Figure I-8 - Character Frame

At the terminal transport layer (TTL), data shall always be passed over the I/O line most significant (m.s.) byte first. The order of bits within a byte (that is, whether the least significant (l.s.) or m.s. bit is transferred first) is specified in character TS returned in the answer to reset (see section I-4.3).

4. Answer to Reset

After being reset by the terminal as described in section I-2.1.3, the ICC answers with a string of bytes known as the ATR. These bytes convey information to the terminal that defines certain characteristics of the communication to be established between the ICC and the terminal. The method of transporting these bytes, and their meaning, is described below.

Note: In sections I-4 and I-5, the m.s. bit of a character refers to bit b8 and the l.s. bit refers to bit b1. A value in inverted commas is coded in hexadecimal notation, for example, '3F'.

4.1 Physical Transportation of Characters Returned at Answer to Reset

This section describes the structure and timing of the characters returned at the answer to reset.

The bit duration is defined in section I-3.1, and the character frame is defined in section I-3.2.

During the answer to reset, the minimum interval between the leading edges of the start bits of two consecutive characters shall be 12 initial etus, and the maximum interval between the leading edges of the start bits of two consecutive characters shall be 9600 initial etus.

The ICC shall transmit all the characters to be returned during an answer to reset (warm or cold) within 19,200 initial etus[6]. This time is measured between the leading edge of the start bit of the first character (TS) and 12 initial etus after the leading edge of the start bit of the last character.

4.2 Characters Returned by ICC at Answer to Reset

The number and coding of the characters returned by the ICC at the answer to reset varies depending upon the transmission protocol(s) and the values of the transmission control parameters supported. This section describes two basic answers to reset, one for ICCs supporting T=0 only and the other for ICCs supporting T=1 only. It defines the characters to be returned and the allowable ranges of values for the transmission control parameters. ICCs returning one of the two answers to reset described here are assured correct operation and interoperability in terminals conforming to this specification.

For proprietary reasons ICCs may optionally support more than one transmission protocol, but one of the supported protocols shall be T=0 or T=1. The first offered protocol shall be T=0 or T=1, and the terminal shall continue the card session using

[6] The maximum time allowed for the answer to reset varies according to clock frequency, since the period represented by an etu is frequency dependent (see section 3.1).

the first offered protocol unless for proprietary reasons it supports a mechanism for selecting an alternative protocol offered by the ICC. Support for such a mechanism is not required by, and is beyond the scope of these specifications.

Note: This specification does not support ICCs having both T=0 and T=1 protocols present at the same time. This can only be achieved by proprietary means beyond the scope of this specification.

Also for proprietary reasons ICCs may optionally support other values of the transmission control parameters at the issuer's discretion. However, such support is considered outside the scope of this specification and such ICCs may be rejected at terminals conforming to this specification, which need not have the corresponding additional proprietary functionality required to support the ICC.

The characters returned by the ICC at the answer to reset for the two basic answers to reset are shown in Tables I-11 and I-12. The characters are shown in the order in which they are sent by the ICC, that is, TS first.

If protocol type T=0 only is supported (character-oriented asynchronous transmission protocol), the characters returned shall be as shown in Table I-11:

Character	Value	Remarks
TS	'3B' or '3F'	Indicates direct or inverse convention
T0	'6x'	TB1 and TC1 present; x indicates the number of historical bytes present
TB1	'00'	VPP not required
TC1	'00' to 'FF'	Indicates the amount of extra guardtime required. Value 'FF' has a special meaning (see section I-4.3.3.3)

Table I-11 - Basic ATR for T=0 Only

If protocol type T=1 only is supported (block-oriented asynchronous transmission protocol), the characters returned shall be as shown in Table I-12:

Character	Value	Remarks
TS	'3B' or '3F'	Indicates direct or inverse convention
T0	'Ex'	TB1 to TD1 present; x indicates the number of historical bytes present
TB1	'00'	VPP not required
TC1	'00' to 'FF'	Indicates amount of extra guardtime required. Value 'FF' has special meaning - see section I-4.3.3.3
TD1	'81'	TA2 to TC2 absent; TD2 present; T=1 to be used
TD2	'31'	TA3 and TB3 present; TC3 and TD3 absent; T=1 to be used
TA3	'10' to 'FE'	Returns IFSI, which indicates initial value for information field size for the ICC and IFSC of 16-254 bytes
TB3	m.s. nibble '0' to '4;' l.s. nibble '0' to '5'	BWI = 0 to 4 CWI = 0 to 5
TCK	See section I-4.3.4	Check character

Table I-12 - Basic ATR for T=1 Only

4.3 Character Definitions

This section provides detailed descriptions of the characters that may be returned at the answer to reset. The presence or absence of a character, and the allowable range of values it may take (if present) if it is to conform to one of the basic ATRs is indicated by 'basic response' in the description of each character. The description of a basic response (even though indicated by 'shall') is not intended to preclude the use of other values of the characters, nor the omission/inclusion of a character at the issuer's discretion. For example, the ICC may return additional characters if it supports more than one transmission protocol (see section I-5). However, only ICCs returning a basic ATR, or an ATR supported by the minimum required terminal functionality described below, are guaranteed to be supported correctly in interchange.

Terminals conforming to this specification are only required (as a minimum) to support the basic ATRs described here together with any additional requirements specified in 'terminal behaviour'. Terminals may thus reject an ATR not supported by this required functionality. However, terminals may, *in addition*, be capable of correctly interpreting an ATR that does not conform to this specification but that is returned by an ICC for proprietary (for example, national) use. Such terminal functionality is not mandatory and is beyond the scope of this specification. As a general principle, a terminal should accept a non basic ATR if it is able to function correctly with the ATR that was returned.

The maximum number of characters returned in the answer to reset (including the historical bytes but not including TS) shall be 32.

Terminals shall be capable of checking the parity of characters returned in the answer to reset, but not necessarily as they are received.

In the following character descriptions, if it is indicated that a terminal shall:

- reject an ATR, it means that the terminal shall issue a warm reset if it is rejecting a cold ATR, or terminate the card session by deactivating the ICC contacts if it rejecting a warm ATR

- reject an ICC, it means that the terminal shall terminate the card session by deactivating the ICC contacts

- accept an ATR, it means that the terminal shall accept the ATR, but *only* if the requirements specified in this section for all other characters are also met.

Each character description is structured in the following way:

- title
- explanation of usage as described in ISO/IEC 7816-3
- EMV basic response. This response should always be used in a warm ATR to ensure interoperability
- required terminal behaviour in the event that a terminal receives characters outside the range allowed by EMV

4.3.1 TS - Initial Character

TS performs two functions. It provides a known bit pattern to the terminal to facilitate bit synchronisation, and it indicates the logic convention to be used for the interpretation of the subsequent characters.

Using inverse logic convention, a low state L on the I/O line is equivalent to a logic one, and the m.s. bit of the data byte is the first bit sent after the start bit. Using direct logic convention, a high state H on the I/O line is equivalent to a logic one, and the l.s. bit of the data byte is the first bit sent after the start bit. The first four bits LHHL are used for bit synchronisation.

Basic responses: The ICC shall return an ATR containing TS having one of two values:

- (H)LHHLLLLLLH - inverse convention, value ‘3F’

- (H)LHHLHHHLLH - direct convention, value ‘3B’

Terminal behaviour: The terminal shall be capable of supporting both inverse and direct convention and shall accept an ATR containing TS having a value of either '3B' or '3F'. An ICC returning an ATR containing TS having any other value shall be rejected.

Note: It is strongly recommended that a value of '3B' is returned by the ICC since a value of '3F' may not be supported in future versions of this specification.

4.3.2 T0 - Format Character

T0 is comprised of two parts. The m.s. nibble (b5-b8) is used to indicate whether the subsequent characters TA1 to TD1 are present. Bits b5-b8 are set to the logic one state to indicate the presence of TA1 to TD1, respectively. The l.s. nibble (b1-b4) indicates the number of optional historical bytes present (0 to 15). (See Table I-13 for the basic response coding of character T0.)

Basic responses: The ATR shall contain T0 = '6x' if T=0 only is to be used, indicating that characters TB1 and TC1 are present. The ATR shall contain T0 = 'Ex' if T=1 only is to be used, indicating that characters TB1 to TD1 are present. The value of 'x' represents the number of optional historical bytes to be returned.

Terminal behaviour: The terminal shall accept an ATR containing T0 of any value provided that the value returned correctly indicates and is consistent with the interface characters TA1 to TD1 and historical bytes actually returned

	b8	b7	b6	b5	b4	b3	b2	b1
T=0 only	0	1	1	0	x	x	x	x
T=1 only	1	1	1	0	x	x	x	x

Table I-13 - Basic Response Coding of Character T0

4.3.3 TA1 to TC3 - Interface Characters

TA1 to TC3 convey information that shall be used during exchanges between the terminal and the ICC subsequent to the answer to reset. They indicate the values of the transmission control parameters F, D, I, P, and N, and the IFSC, block waiting time integer (BWI), and character waiting time integer (CWI) applicable to T=1 as defined in ISO/IEC 7816-3. The information contained in TA1, TB1, TC1, TA2, and TB2 shall apply to all subsequent exchanges irrespective of the protocol type to be used.

4.3.3.1 TA1

TA1 conveys the values of FI and DI where:

- FI is used to determine the value of F, the clock rate conversion factor, which may be used to modify the frequency of the clock provided by the terminal subsequent to the answer to reset

- DI is used to determine the value of D, the bit rate adjustment factor, which may be used to adjust the bit duration used subsequent to the answer to reset

See section I-3.1 for calculation of the bit duration subsequent to the answer to reset (current etu).

Default values of FI = 1 and DI = 1 indicating values of F = 372 and D = 1, respectively, shall be used during the answer to reset.

Basic response: The ATR shall not contain TA1 and thus the default values of F = 372 and D = 1 shall continue be used during all subsequent exchanges.

Terminal behaviour: If TA1 is present in the ATR (indicated by b5 of T0 set to '1') and TA2 is returned with b5 = '0' (specific mode, parameters defined by the interface bytes), the terminal shall:

- Accept the ATR if the value of TA1 is '11', and immediately implement the values of F and D indicated.
- Reject the ATR if the value of TA1 is not '11', unless it is able to support and immediately implement the conditions indicated. Support for values of TA1 other than '11' is not required by this specification.

If TA1 is present in the ATR (indicated by b5 of T0 set to '1') and TA2 is not returned (negotiable mode), the terminal shall only accept the ATR if TA1 = '11' and shall continue using the default values of D = 1 and F = 372 during all subsequent exchanges.

If TA1 is absent from the ATR, the default values of D = 1 and F = 372 shall be used during all subsequent exchanges.

Note: It is strongly recommended that new terminals shall be capable of correctly interpreting the l.s. nibble of TA1 if it is returned (that codes the bit rate adjustment factor D), and of correctly implementing values of D of 1, 2, or 4. Future versions of this specification may support other values of D to improve data transfer rates between the TTL and ICC and a protocol type selection (PTS) mechanism for selecting the value to be used.

4.3.3.2 TB1

TB1 conveys the values of PI1 and II where:

- PI1 is specified in bits b1 to b5 and is used to determine the value of the programming voltage P required by the ICC. PI1 = 0 indicates that VPP is not connected in the ICC.
- II is specified in bits b6 and b7 and is used to determine the maximum programming current I required by the ICC. This parameter is not used if PI1 = 0.

- Bit 8 is not used and shall be set to logic zero.

Basic response: The ATR shall contain TB1 = ‘00’, indicating that VPP is not connected in the ICC.

Terminal behaviour: In response to a cold reset, the terminal shall accept only an ATR containing TB1 = ‘00’. In response to a warm reset the terminal shall accept an ATR containing TB1 of any value (provided that b6 of T0 is set to 1) or not containing TB1 (provided that b6 of T0 is set to 0) and shall continue the card session as though TB1 = ‘00’ had been returned. V_{PP} shall never be generated.

Note: Existing terminals may maintain V_{PP} in the idle state (see section I-1.3.3).

The basic response coding of character TB1 is shown in Table I-14:

b8	b7	b6	b5	b4	b3	b2	b1
0	0	0	0	0	0	0	0

Table I-14 - Basic Response Coding of Character TB1

4.3.3.3 TC1

TC1 conveys the value of N, where N is used to indicate the extra guardtime that shall be added to the minimum duration between the leading edges of the start bits of two consecutive characters for subsequent exchanges from the terminal to the ICC. N is binary coded over bits b1-b8 of TC1, and its value represents the number of etus to be added as extra guardtime. It may take any value between 0 and 255. N = 255 has a special meaning and indicates that the minimum delay between the start leading edges of two consecutive characters shall be reduced to 12 etus if T=0 is to be used, or 11 etus if T=1 is to be used.

Note: TC1 applies only to the timing between two consecutive characters sent from the terminal to the ICC. It does not apply to the timing between consecutive characters sent from the ICC to the terminal, nor does it apply to the timing between two characters sent in opposite directions. See sections I-5.2.2.1 and I-5.2.4.2.2.

If the value of TC1 is in the range ‘00’ to ‘FE’, between 0 and 254 etus of extra guardtime shall be added to the minimum character to character duration, which for subsequent transmissions shall be between 12 and 266 etus.

If the value of TC1 = ‘FF’ the minimum character to character duration for subsequent transmissions shall be 12 etus if T=0 is to be used, or 11 etus if T=1 is to be used.

Basic response: The ATR shall contain TC1 having a value in the range ‘00’ to ‘FF’.

Terminal behaviour: The terminal shall accept an ATR not containing TC1 (provided that b7 of T0 is set to 0), but if it accepts such an ATR it shall continue the card session as though TC1 = '00' had been returned.

The basic response coding of character TC1 is shown in Table I-15:

b8	b7	b6	b5	b4	b3	b2	b1
x	x	x	x	x	x	x	x

Table I-15 - Basic Response Coding of Character TC1

Note: It is strongly recommended that the value of TC1 be set to the minimum acceptable for the ICC. Large values of TC1 lead to very slow communication between the terminal and the ICC, and thus lengthy transaction times.

4.3.3.4 TD1

TD1 indicates whether any further interface bytes are to be transmitted and information concerning the protocol type(s) where:

- The m.s. nibble is used to indicate whether the characters TA2 to TD2 are present. These bits (b5-b8) are set to the logic one state to indicate the presence of TA2 to TD2 respectively.

- The l.s. nibble provides information concerning the protocol type(s) to be used for subsequent exchanges.

Basic responses: The ATR shall not contain TD1 if T=0 only is to be used, and protocol type T=0 shall be used as a default for all subsequent transmissions. The ATR shall contain TD1 = '81' if T=1 only is to be used, indicating that TD2 is present and that protocol type T=1 shall be used for all subsequent transmissions.

Terminal behaviour: The terminal shall accept an ATR containing TD1 with the m.s. nibble having any value (provided that the value returned correctly indicates and is consistent with the interface characters TA2 to TD2 actually returned), and the l.s. nibble having a value of '0' or '1'. The terminal shall reject an ATR containing other values of TD1.

The basic response coding of character TD1 is shown in Table I-16:

	b8	b7	b6	b5	b4	b3	b2	b1
T=1	1	0	0	0	0	0	0	1

Table I-16 - Basic Response Coding of Character TD1

4.3.3.5 TA2

The presence or absence of TA2 indicates whether the ICC is operating in specific mode or negotiable mode respectively.

Basic response: The ATR shall not contain TA2; the absence of TA2 indicates the negotiable mode of operation.

Terminal behaviour: The terminal shall accept an ATR containing TA2 provided that b5 = 0, and that it is able to support the exact conditions indicated by the interface parameters returned by the ICC in the answer to reset and immediately uses those conditions. Otherwise, the terminal shall reject an ATR containing TA2.

4.3.3.6 TB2

TB2 conveys PI2 that is used to determine the value of programming voltage P required by the ICC. When present it overrides the value indicated by PI1 returned in TB1.

Basic response: The ATR shall not contain TB2.

Terminal behaviour: The terminal shall reject an ATR containing TB2.

Note: Existing terminals may maintain V_{PP} in the idle state (see section I-1.3.3).

4.3.3.7 TC2

TC2 is specific to protocol type T=0 and conveys the work waiting time integer (WI) that is used to determine the maximum interval between the start leading edge of any character sent by the ICC and the start leading edge of the previous character sent either by the ICC or the terminal (the work waiting time). The work waiting time is given by 960 x D x WI.

Basic response: The ATR shall not contain TC2 and a default value of WI = 10 shall be used during subsequent communication.

Terminal behaviour: The terminal shall accept an ATR containing TC2 = ‘0A’. It shall reject an ATR containing TC2 = ‘00’, or TC2 > ‘0A’. It shall reject an ATR containing TC2 having a value in the range ‘01’ to ‘09’ unless it is able to support it.

4.3.3.8 TD2

TD2 indicates whether any further interface bytes are to be transmitted and the protocol type to be used for subsequent transmissions, where:

- The m.s. nibble is used to indicate whether the characters TA3 to TD3 are present. These bits (b5-b8) are set to the logic one state to indicate the presence of TA3 to TD3, respectively.

- The l.s. nibble indicates the protocol type to be used for subsequent exchanges. It shall take the value '1' as T=1 is to be used.

Basic responses: The ATR shall not contain TD2 if T=0 is to be used, and the protocol type T=0 shall be used as a default for all subsequent transmissions. The ATR shall contain TD2 = '31' if T=1 is to be used, indicating that TA3 and TB3 are present and that protocol type T=1 shall be used for all subsequent transmissions.

Terminal behaviour: The terminal shall accept an ATR containing TD2 with the m.s. nibble having any value (provided that the value returned correctly indicates and is consistent with the interface characters TA3 to TD3 actually returned), and the l.s. nibble having a value of '1' (or 'E' if the l.s. nibble of TD1 is '0'). The terminal shall reject an ATR containing other values of TD2.

The basic response coding of character TD2 is shown in Table I-17:

	b8	b7	b6	b5	b4	b3	b2	b1
T=1	0	0	1	1	0	0	0	1

Table I-17 - Basic Response Coding of Character TD2

4.3.3.9 TA3

TA3 (if T=1 is indicated in TD2) returns the information field size integer for the ICC (IFSI), which determines the IFSC, and specifies the maximum length of the information field (INF) of blocks that can be received by the card. It represents the length of IFSC in bytes and may take any value between '01' and 'FE'. Values of '00' and 'FF' are reserved for future use.

Basic response: The ATR shall contain TA3 having a value in the range '10' to 'FE' if T=1 is to be used indicating an initial IFSC in the range 16 to 254 bytes.

Terminal behaviour: The terminal shall accept an ATR not containing TA3 (provided that b5 of TD2 is set to 0), but if it accepts such an ATR it shall continue the card session using a value of '20' for TA3. The terminal shall reject an ATR containing TA3 having a value in the range '00' to '0F' or a value of 'FF'.

The basic response coding of character TA3 is shown in Table I-18:

	b8	b7	b6	b5	b4	b3	b2	b1
T=1	x	x	x	x	x	x	x	x
	'00' to '0F' and 'FF' not allowed							

Table I-18 - Basic Response Coding of Character TA3

4.3.3.10 TB3

TB3 (if T=1 is indicated in TD2) indicates the values of the CWI and the BWI used to compute the CWT and BWT respectively. TB3 is comprised of two parts. The l.s. nibble (b1-b4) is used to indicate the value of CWI, whilst the m.s. nibble (b5-b8) is used to indicate the value of BWI.

Basic response: The ATR shall contain TB3 having the l.s. nibble in the range ‘0’ to ‘5’, and the m.s. nibble in the range ‘0’ to ‘4’ if T=1 is to be used, indicating values of 0 to 5 for CWI and 0 to 4 for BWI.

The basic response coding of character TB3 is shown in Table I-19:

	b8	b7	b6	b5	b4	b3	b2	b1
T=1	0	x	x	x	0	y	y	y
	xxx is in the range 000 to 100 yyy is in the range 000 to 101							

Table I-19 - Basic Response Coding of Character TB3

Terminal behaviour: The terminal shall reject an ATR not containing TB3, or containing a TB3 indicating BWI greater than 4 and/or CWI greater than 5, or having a value such that $2^{CWI} < (N + 1)$. It shall accept an ATR containing a TB3 having any other value.

Note: N is the extra guardtime indicated in TC1. If TC1=255, the value of N shall be taken as -1.

4.3.3.11 TC3

TC3 (if T=1 is indicated in TD2) indicates the type of block error detection code to be used. The type of code to be used is indicated in b1, and b2 to b8 are not used.

Basic response: The ATR shall not contain TC3, thus indicating longitudinal redundancy check (LRC) as the error code to be used.

Terminal behaviour: The terminal shall accept an ATR containing TC3 = ‘00’. It shall reject an ATR containing TC3 having any other value.

4.3.4 TCK - Check Character

TCK has a value that allows the integrity of the data sent in the ATR to be checked. The value of TCK is such that the exclusive-OR’ing of all bytes from T0 to TCK inclusive is null.

Basic responses: The ATR shall not contain TCK if T=0 only is to be used. In all other cases TCK shall be returned in the ATR.

Terminal behaviour: The terminal shall accept an ICC returning an ATR not containing TCK if T=0 only is indicated. In all other cases, the terminal shall reject an ICC returning an ATR not containing TCK or containing an incorrect TCK, and shall be able to evaluate TCK when appropriately returned. If TCK is incorrect, the terminal shall initiate the deactivation sequence within 4,800 initial etus following the leading edge of the start bit of TCK. For either protocol, the terminal may continue the card session as soon as the last character indicated by the bit map characters T0 and/or TDi has been received. The terminal shall wait at least the guardtime applicable to the protocol to be used (16 etus for T=0, BGT for T=1) before transmitting.

Note: If TCK is not returned, bullet 3 of section I-4.4 applies.

4.4 Terminal Behaviour during Answer to Reset

Following activation of the ICC contacts as described in section I-2.1.2 the terminal shall initiate a cold reset as described in section I-2.1.3.1. Subsequently the following shall apply:

- If the terminal rejects a cold ATR as described in section I-4.3, it shall not immediately abort the card session but shall initiate a warm reset within 4,800 initial etus (reaction time) measured from the leading edge of the start bit of the last received character of the cold ATR to the time that RST is set low.

- If the terminal rejects a warm ATR as described in section I-4.3, it shall initiate the deactivation sequence within 4,800 initial etus measured from the leading edge of the start bit of the last received character of the warm ATR.

- If during the answer to either a cold reset or warm reset the time between two consecutive characters exceeds 9,600 initial etus as defined in section I-4.1, the terminal shall abort the card session by initiating the deactivation sequence within 14,400 initial etus (4,800 + 9,600 initial etus) following the leading edge of the start bit of the last received character.

- If the answer to a cold or warm reset is not complete within 19,200 initial etus as defined in section I-4.1, the terminal shall abort the card session by initiating the deactivation sequence within 24,000 initial etus (4,800 + 19,200 initial etus) following the leading edge of the start bit of the TS character.

- If the terminal detects a parity error in a character returned during ATR, it shall abort the card session by initiating the deactivation sequence within 4,800 initial etus measured from the leading edge of the start bit of the last received character of the ATR.

- Upon receipt of a valid cold or warm reset complying with the timings described above, the terminal shall proceed with the card session using the returned parameters.

4.5 Answer to Reset - Flow at the Terminal

Figure I-9 illustrates an example of the process of an ICC returning an ATR to the terminal and the checks performed by the terminal to ensure conformance to section I-4.

Note 1: 'Case' is a process variable used to indicate whether a cold or warm reset is operative. Case = 1 for a cold reset, and Case = 2 for a warm reset.

Note 2: If the process aborts at this point, the ICC may be acceptable in this terminal by business agreement. The terminal should be primed to accept the ICC prior to insertion. Any subsequent processing is proprietary and beyond the scope of this specification.

Note 3: If the process aborts at this point, reset may be retried after removing the ICC from the terminal and taking corrective action as required. An appropriate message should be displayed on the terminal.

Note 4: A proprietary card session beyond the scope of this specification may be started at this point.

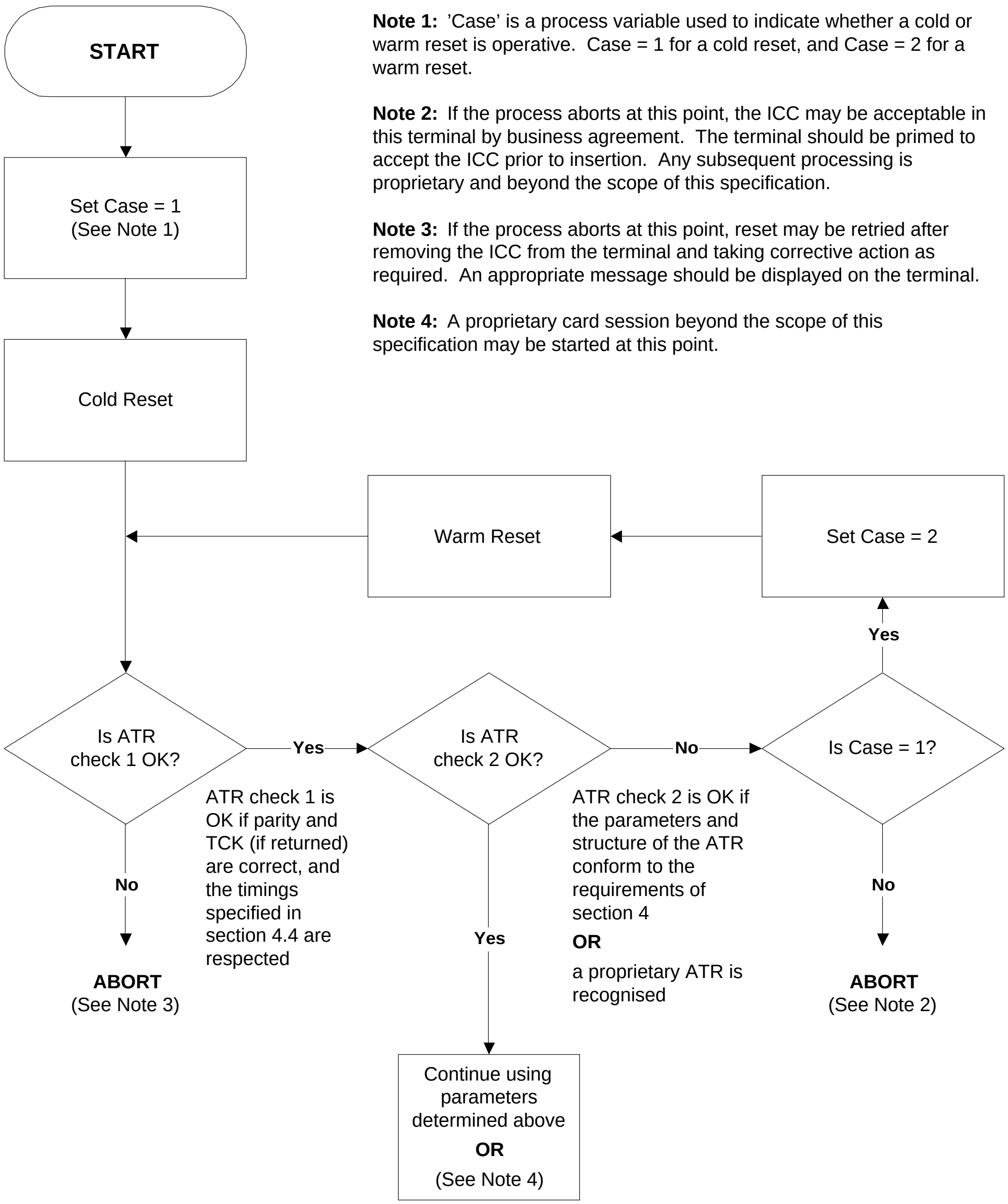

Figure I-9 - ATR - Example Flow at the Terminal

5. Transmission Protocols

This section defines the structure and processing of commands initiated by the terminal for transmission control and for specific control in asynchronous half duplex transmission protocols.

Two types of protocol are defined, character protocol (T=0) and block protocol (T=1). ICCs shall support either protocol T=0 or protocol T=1. Terminals shall support both protocol T=0 and T=1. The protocol to be used for subsequent communication between the ICC and terminal is indicated in TD1, and shall be T=0 or T=1. If TD1 is absent in the ATR, T=0 is assumed. The protocol indicated by the ICC applies immediately after the answer to reset, as there is no PTS procedure. Other parameters provided in the ATR and relevant to a specific protocol are defined in the respective parts of this section.

The protocols are defined according to the following layering model:

- Physical layer, which describes the exchanges of bits and is common to both protocols.

- Data link layer, which includes the following sub-definitions:

  - Character frame, defining the exchanges of characters common to both protocols.

  - Character protocol T=0, defining the exchange of characters specific to T=0.

  - Error detection and correction specific to T=0.

  - Block protocol T=1, defining the exchanges of blocks specific to T=1.

  - Error detection and correction specific to T=1.

- Transport layer, which defines the transmission of application-oriented messages specific to each protocol.

- Application layer, which defines the exchange of messages according to an application protocol that is common to both transmission protocols.

5.1 Physical Layer

Both protocols T=0 and T=1 use the physical layer and character frame as defined in section I-3.

5.2 Data Link Layer

This section describes the timing, specific options, and error handling for protocols T=0 and T=1.

5.2.1 Character Frame

The character frame as defined in section I-3.2 applies to all messages exchanged between the ICC and the terminal.

5.2.2 Character Protocol T=0

5.2.2.1 Specific Options - Character Timing for T=0

The minimum interval between the leading edges of the start bits of two consecutive characters sent by the terminal to the ICC shall be between 12 and 266 etus as indicated by the value of TC1 returned at the answer to reset (see sections I-4.2 and I-4.3).

The minimum interval between the leading edges of the start bits of two consecutive characters sent by the ICC to the terminal shall be 12 etus

The maximum interval between the start leading edge of any character sent by the ICC and the start leading edge of the previous character sent either by the ICC or the terminal (the Work Waiting Time) shall not exceed 960 x D x WI etus. (D and WI are returned in TA1 and TC2 ,respectively.) If the Work Waiting Time is exceeded, the terminal shall initiate the deactivation sequence within 960 etus.

The minimum interval between the leading edges of the start bits of two consecutive characters sent in opposite directions shall be 16 etus.

Note: The minimum interval between the leading edges of the start bits of two consecutive characters sent by the terminal to the ICC is always governed by the value of TC1, and may be less than the minimum interval of 16 etus allowed between two characters sent in opposite directions.

5.2.2.2 Command Header

A command is always initiated by the terminal application layer (TAL) which sends an instruction via the TTL to the ICC in the form of a five byte header called the command header. The command header is comprised of five consecutive bytes, CLA, INS, P1, P2, and P3, where:

- CLA is the command class.
- INS is the instruction code.
- P1 and P2 contain additional instruction specific parameters.

- P3 indicates either the length of data to be sent with the command to the ICC, or the maximum length of data expected in the response from the ICC, depending on the coding of INS.

These bytes together with any data to be sent with the command constitute the command transport protocol data unit (C-TPDU) for T=0. The mapping of the command application protocol data unit (C-APDU) onto the C-TPDU is described in section I-5.3.

The TTL transmits the five-byte header to the ICC and waits for a procedure byte.

5.2.2.3 Command Processing

Following reception of a command header by the ICC, the ICC shall return a procedure byte or status bytes SW1 SW2 (hereafter referred to as ‘status’) to the TTL. Both the TTL and ICC shall know implicitly at any point during exchange of commands and data between the TTL and the ICC what the direction of data flow is and whether it is the TTL or the ICC that is driving the I/O line.

5.2.2.3.1 Procedure Byte

The procedure byte indicates to the TTL what action it shall take next. The coding of the byte and the action that shall be taken by the TTL is shown in Table I-20.

Procedure Byte Value	Action
Equal to INS byte	All remaining data bytes shall be transferred by the TTL, or the TTL shall be ready to receive all remaining data bytes from the ICC
Equal to complement of INS byte ($\overline{\text{INS}}$)	The next data byte shall be transferred by the TTL, or the TTL shall be ready to receive the next data byte from the ICC
‘60’	The TTL shall provide additional work waiting time as defined in this section
‘61’	The TTL shall wait for a second procedure byte then send a GET RESPONSE command header to the ICC with a maximum length of ‘xx’, where ‘xx’ is the value of the second procedure byte
‘6C’	The TTL shall wait for a second procedure byte then immediately resend the previous command header to the ICC using a length of ‘xx’, where ‘xx’ is the value of the second procedure byte

Table I-20 - Terminal Response to Procedure Byte

In all cases, after the action has taken place the TTL shall wait for a further procedure byte or status.

5.2.2.3.2 Status Bytes

The status bytes indicate to the TTL that command processing by the ICC is complete. The meaning of the status bytes is related to the command being processed and is defined in Part II of these specifications.

First Status Byte Value	Action
‘6x’ or ‘9x’ (except ‘60’, ‘61’ and ‘6C’) - status byte SW1	TTL shall wait for a further status byte (status byte SW2)

Table I-21 - Status Byte Coding

Following receipt of the second status byte, the TTL shall return the status bytes (together with any appropriate data - see section I-5.3.1) to the TAL in the response APDU (R-APDU) and await a further C-APDU.

5.2.2.3.3 Error Handling

When awaiting a procedure byte or status byte, if the byte returned by the ICC has a value other than specified in sections I-5.2.2.3.1 and I-5.2.2.3.2, the terminal shall initiate the deactivation sequence within 9,600 etus following the leading edge of the start bit of the (invalid) byte received.

5.2.2.4 Transportation of C-APDUs

A C-APDU containing only command data to be sent to the ICC, or only expecting data in response from the ICC (cases 2 and 3 in section I-5.4), is mapped without change onto a T=0 C-TPDU. A C-APDU that contains and expects no data, or a C-APDU that requires data transmission to and from the ICC (cases 1 and 4 in section I-5.4) is translated according to the rules defined in section I-5.3 for transportation by a C-TPDU for T=0.

5.2.3 Error Detection and Correction for T=0

This procedure is mandatory for T=0 but does not apply during the answer to reset.

If a character is not received correctly or is received correctly but with incorrect parity, the receiver shall indicate an error by setting the I/O line to state L at time (10.5 ± 0.2) etus following the leading edge of the start bit of the character for a minimum of 1 etu and a maximum of 2 etus.

The transmitter shall test the I/O line (11 ± 0.2) etus after the leading edge of the start bit of a character was sent, and assumes that the character was correctly received if the I/O line is in state H.

If the transmitter detects an error, it shall repeat the disputed character after a delay of at least 2 etus following detection of the error. The transmitter shall repeat the same disputed character a maximum of three more times.

If the last repetition is unsuccessful, the terminal shall initiate the deactivation sequence within 960 etus following reception of the leading edge of the start bit of the invalid character (if it is the receiver), or within 960 etus following detection of the signaling of the parity error by the ICC (if it is the transmitter).

5.2.4 Block Protocol T=1

The protocol consists of blocks transmitted between the TAL and the ICC to convey command and R-APDUs and transmission control information (for example, acknowledgment).

The data link layer block frame structure, specific options of the protocol, and protocol operations (including error handling) are defined below.

5.2.4.1 Block Frame Structure

The character frame as defined in section I-3.2 applies.

The block is structured as follows (see Table I-22):

- Mandatory prologue field
- Optional information field
- Mandatory epilogue field

Prologue Field			Information Field	Epilogue Field
Node Address (NAD)	Protocol Control Byte (PCB)	Length (LEN)	APDU or Control Information (INF)	Error Detection Code (EDC)
1 byte	1 byte	1 byte	0-254 bytes	1 byte

Table I-22 - Structure of a Block

5.2.4.1.1 Prologue Field

The prologue field consists of three mandatory bytes:

- Node address to identify source and intended destination of the block and to provide VPP state control
- Protocol control byte to control data transmission
- Length of the optional information field

5.2.4.1.1.1 Node Address

Bits b1-b3 of NAD indicate the source node address (SAD) of the block, whilst bits b5 -b7 indicate the intended destination node address (DAD) of the block. Bits b4 and b8[7] are unused and shall be set to 0.

These specifications do not support node addressing. All blocks transmitted by either the terminal or ICC in support of EMV defined applications shall have the NAD byte set to ‘00’.

If during the card session the terminal or ICC receives a block with a NAD ≠ ‘00’, it may treat the block as invalid. In this event, it shall apply the error detection and correction techniques described in section I-5.2.5.

Note: Support for node addressing is not required by this specification and interoperability can only be guaranteed if the requirements of this section are met. However, terminals and ICCs may for proprietary reasons support node addressing. An ICC that optionally supports node addressing should work correctly with terminals/applications compliant with these specifications. A terminal that optionally supports node addressing must set NAD = ‘00’ when working with ICCs/applications compliant with these specifications.

5.2.4.1.1.2 Protocol Control Byte

The PCB codes the type of block. There are three types of blocks defined as follows:

- Information block (I-block) used to convey APDUs.
- Receive-ready block (R-block) used to convey acknowledgments (ACK or NAK).
- Supervisory block (S-block) used to exchange control information.

The coding of the PCB depends on its type and is defined in Tables I-23 to I-25.

b8	0
b7	Sequence number
b6	Chaining (more data)
b5-b1	Reserved for future use (RFU)

Table I-23 - Coding of the PCB of an I-block

[7] Defined in ISO/IEC 7816 as VPP control. A value of 0 indicates that VPP shall be maintained in the idle state.

b8	1
b7	0
b6	0
b5	Sequence number
b4-b1	0 = Error free 1 = EDC and/or parity error 2 = Other error(s) Other values RFU

Table I-24 - Coding of the PCB of an R-block

b8	1
b7	1
b6	0 = Request 1 = Response
b5-b1	0 = Resynchronisation request 1 = Information field size request 2 = Abort request 3 = Extension of BWT request 4 = VPP error[8] Other values RFU

Table I-25 - Coding of the PCB of a S-block

5.2.4.1.1.3 Length

The LEN codes the length of the INF part of the block; it may range from 0 to 254 depending on the type of block.

Note: This specification does not support I-blocks with LEN = 0.

5.2.4.1.2 Information Field

The INF is conditional. When present in an I-block, it conveys application data. When present in a S-block, it conveys control information. An R-block shall not contain an INF.

5.2.4.1.3 Epilogue Field

The Epilogue Field contains the EDC of the transmitted block. A block is invalid when a parity error and/or an EDC error occurs. This specification only supports the LRC as EDC. The LRC is one byte in length and is calculated as the exclusive-OR of all the bytes starting with the NAD and including the last byte of INF, if present.

Note: TCi (i > 2), which indicates the type of error detection code to be used, is not returned by the ICC in the ATR. The normal default of the LRC is thus used for the EDC.

---

[8] Not used by ICCs and terminals conforming to this specification.

5.2.4.1.4 Block Numbering

I-blocks are numbered using a modulo-2 number coded on one bit. The numbering system is maintained independently at the ICC and the terminal as senders. The value of the number starts with zero for the first I-block sent after the answer to reset by a sender and is incremented by one after sending each I-block. The number is reset to zero by the sender after resynchronisation.

R-blocks are numbered using a modulo-2 number coded on one bit. A R-block may be used to acknowledge a chained I-block or to request retransmission of an invalid block, and always carries the sequence number of the next I-block its sender expects to receive.

A S-block carries no number.

5.2.4.2 Specific Options

This section defines the information field sizes and timings to be used with protocol type T=1.

5.2.4.2.1 Information Field Sizes

The IFSC is the maximum length of the information field of blocks that can be received by the ICC, and is defined as follows. At the answer to reset the IFSI is returned by the ICC in TA3 indicating the size of the IFSC that can be accommodated by the ICC. IFSI may take values in the range ‘10’ to ‘FE’ that code values for IFSC in the range 16 to 254 bytes. The maximum block size that can be received by the ICC is therefore (IFSC + 3 + 1) bytes including the prologue and epilogue fields. The size established during the answer to reset shall be used throughout the rest of the card session or until a new value is negotiated by the ICC by sending a S(IFS request) block to the terminal.

The information field size for the terminal (IFSD) is the maximum length of the information field of blocks that can be received by the terminal. The initial size immediately following the answer to reset shall be 254 bytes, and this size shall be used throughout the rest of the card session. The first block sent by the terminal following the answer to reset shall be an S(IFS request) with IFSD = 254.

5.2.4.2.2 Timing for T=1

The minimum interval between the leading edges of the start bits of two consecutive characters sent by the terminal to the ICC shall be between 11 and 266 etus as indicated by the value of TC1 returned at the answer to reset (see sections I-4.2 and I-4.3).

The minimum interval between the leading edges of the start bits of two consecutive characters sent by the ICC to the terminal shall be 11 etus

The maximum interval between the leading edges of the start bits of two consecutive characters in the same block (the CWT shall not exceed (2^{CWI} + 11) etus. The CWI shall have a value of 0 to 5 as described in section I-4.3.3.6, and thus CWT lies in the range 12 to 43 etus.

The maximum interval between the leading edge of the start bit of the last character that gave the right to send to the ICC and the leading edge of the start bit of the first character sent by the ICC (the BWT) shall not exceed {(2^{BWI} x 960) + 11} etus. The BWI shall have a value in the range 0 to 4 as described in section I-4.3.3.6, and thus BWT lies in the range 971 to 15,371 etus.

The minimum interval between the leading edges of the start bits of two consecutive characters sent in opposite directions (the block guard time, BGT) shall be 22 etus.

Note: In general, for values of FI and DI other than 1, BWT is calculated using the formula:

$$\mathrm{BWT} = \left(\left(2^{\mathrm{BWI}} \times 960 \times \frac{372\mathrm{D}}{\mathrm{F}}\right) + 11\right) \mathrm{etu}$$

5.2.4.3 Error Free Operation

The protocol rules for error free operation are as follows:

- The first block transmitted after the answer to reset shall be sent by the terminal to the ICC and shall be a S-block.

- If the terminal wishes to change the size of the IFSD it shall send a S(IFS request) block to the ICC. The PCB of the S(IFS request) block shall have the value ‘C1’ indicating a request to change the IFSD. The INF field shall contain a byte the value of which indicates the size in bytes of the requested new IFSD. The ICC shall return a S(IFS response) block to the terminal acknowledging the change to the size of the IFSD. The PCB of the S(IFS response) block sent in response shall have the value ‘E1’, and the INF field shall have the same value as the INF field of the block requesting the change.

  Note: Terminals complying with this specification shall set IFSD to 254 in the first block sent following the answer to reset, and should not send any further S(IFS request) blocks during the card session.

- If the ICC wishes to change the size of the IFSC from the initial value indicated at the answer to reset, it shall send a S(IFS request) block to the terminal. The PCB of the S(IFS request) block shall have the value ‘C1’ indicating a request to change the IFSC. The INF field shall contain a byte the value of which indicates the size in bytes of the requested new IFSC. This byte shall have a value in the range ‘10’ to ‘FE’. The terminal shall return a S(IFS response) block to the ICC acknowledging the change to the size of the IFSC. The PCB of the S(IFS response block sent in response shall have the value ‘E1’, and the INF field shall have the same value as the INF field of the block requesting the change.

- The SAD and DAD in the first block transmitted by the terminal shall be set to zero indicating that node addressing is not to be used.

  Note: If for proprietary reasons the terminal supports node addressing, the SAD and DAD in the first block transmitted by the terminal should be set to the values to be used throughout the rest of the card session.

- During the card session, only blocks as defined in this section shall be exchanged. The half duplex block protocol consists of transmitting blocks alternately by the terminal and the ICC. When the sender has transmitted a complete block, the sender switches to the receiving state.

- When the receiver has received the number of characters in accordance with the value of LEN and the EDC, the receiver gains the right to send.

- The ICC shall acknowledge an I-block transmitted by the terminal. The acknowledgment is indicated in the sequence number of the I-block, or the R-block if chaining is in use (except the last block of the chain), that the ICC returns to the terminal.

- An I-block is considered by the sender to be acknowledged when the sequence number of the I-block received in response differs from the sequence number of the previously received I-block. If no I-block was previously received, the sequence number of the I-block sent in response shall be 0.

- During chaining, an I-block is considered by the sender to be acknowledged when the sequence number of the R-block sent in response differs from the sequence number of the I-block being acknowledged and it has bits b4-b1 set to '0' (indicating an error free block).

- If the ICC requires more than the BWT to process the previously received I-block, it sends a waiting time extension request S(WTX request) block, where the INF contains the one-byte binary integer multiplier of the BWT value requested. The terminal shall acknowledge by sending a waiting time extension response S(WTX response) block with the same value in the INF. The time allocated (which is the time requested in the S(WTX request) block, and which replaces BWT for this instance only) starts at the leading edge of the last character of the S(WTX response) block. After the ICC responds, BWT is again used as the time allowed for the ICC to process the I-block.

- S-blocks are only used in pairs. A S(request) block is always followed by a S(response) block.

When synchronisation as outlined above is lost, the procedure described in section I-5.2.5 shall apply.

5.2.4.4 Chaining

When the sender has to transmit data of length greater than IFSC or IFSD bytes, it shall divide it into several consecutive I-blocks. The transmission of these multiple I-blocks is achieved using the chaining function described below.

The chaining of I-blocks is controlled by b6 of the PCB. The coding of b6 is as follows:

- b6 = 0 Last block of the chain
- b6 = 1 Subsequent block follows

Any I-block with b6 = 1 shall be acknowledged by an R-block according to section I-5.2.4.1.

The last block of a chain sent by the terminal shall be acknowledged by either an I-block if correctly received, or an R-block if incorrectly received. The last block of a chain sent by the ICC shall be acknowledged by an R-block if incorrectly received; if correctly received, the terminal will only transmit further I-blocks if another command is to be processed.

5.2.4.4.1 Rules for Chaining

The TTL shall support chaining for both transmitted and received blocks. The ICC may optionally chain blocks sent to the terminal. Chaining is only possible in one direction at a time. The following rules for chaining apply:

- When the terminal is the receiver, the terminal shall accept a sequence of chained I-blocks sent from the ICC of length ≤ IFSD bytes per block.
- When the ICC is the receiver, the ICC shall accept a sequence of chained I-blocks sent from the terminal all having length LEN = IFSC except the last block, whose length may be in the range 1 to IFSC bytes inclusive.
- When the ICC is the receiver, the ICC shall reject an I-block sent by the terminal of length > IFSC using an R-block with bits b4-b1 of the PCB having a value of ‘2’ (see Table I-24).
- If the ICC as sender chains blocks sent to the terminal it shall send I-blocks of length ≤IFSD bytes per block
- When the terminal is the sender, all I-blocks of a chain sent to the ICC shall have LEN = IFSC bytes except the last, which shall have a length in the range 1 to IFSC bytes inclusive.

5.2.4.4.2 Construction of Chained Blocks

C-APDUs are transported from the TTL to the ICC in the INF field of I-blocks (see section I-5.3.2). If a C-APDU is too large to fit in one block, it is chained over several as illustrated in the following example.

Block(1)	CLA INS P1 P2	Lc	Data		Data
Block(2)	Data		Data		Data
Block(n)	Data		Data	Le	

The data and status returned by the ICC may optionally be chained over several I-blocks as follows.

Block(1)	Data	Data		Data
Block(2)	Data	Data		Data
Block(n)	Data	Data	SW1 SW2	

Note: The above examples are for a case 4 command and show only the INF fields of the chained blocks. Each block also has a prologue and epilogue field. All chained blocks shall contain an INF field having a length in the range 1 to IFSD bytes if the ICC is the sender, or IFSC bytes during chaining and 1 to IFSC bytes in the last block of the chain if the terminal is the sender.

5.2.5 Error Detection and Correction for T=1

The following errors shall be detected by the TTL:

- Transmission error including parity error, EDC error, and BWT time-out.

  Note: If a parity error is detected, character repetition shall not be implemented when using T=1.

- Loss of synchronisation (under run or overrun of the number of characters).
- Protocol error (infringement of the rules of the protocol).
- Abort request for a chain of blocks.

Error recovery is attempted in the following manner.

The TTL shall attempt error recovery by trying the following techniques in the order shown.

1. Retransmission of blocks
2. Deactivation of the ICC contacts

The ICC shall attempt error recovery by trying retransmission of blocks.

If a block is retransmitted, the retransmitted block shall be identical to the originally transmitted block.

Note: In some terminals the TTL may not be solely responsible for error handling. Where 'TTL' is used it includes any functionality present in the terminal as applicable.

5.2.5.1 Protocol Rules for Error Handling

The following rules apply for error handling and correction. In each case where a R-block is sent, the l.s. nibble shall be set to '1' for EDC/parity errors or '2' for other errors (including protocol errors) as defined in Table I-24.

- If the first block received by the ICC after the answer to reset is invalid, it shall return an R-block to the TTL with b5 = 0 and NAD = 0.

- If there is no response from the ICC to any block sent by the TTL within time BWT (or WTX if waiting time extension has been negotiated), or the time between the start bits of the leading edges of two consecutive characters sent by the ICC exceeds CWT, the terminal shall terminate the card session by initiating the deactivation sequence within 4,800 etus.

- If an invalid block is received in response to an I-block, the sender shall transmit a R-block with b5 set to the number of the next I-block it had expected to receive.

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

- If a S(... response) block is not received in response to a S(... request) block, the sender shall retransmit the S(... request) block.

- If an invalid block is received in response to a S(... response) block, the sender shall transmit a R-block with b5 set to the number of the next expected I-block.

- If the TTL has sent a block a maximum of three times in succession, or the ICC has sent a block a maximum of twice in succession without obtaining a valid response, the TTL shall terminate the card session by deactivating the ICC contacts. If the terminal is the receiver, it shall initiate the deactivation sequence within 9,600 etus following reception of the leading edge of the start bit of the last character of the repeated block. If the terminal is the sender, it shall initiate the deactivation sequence within 9,600 etus following reception of the leading edge of the start bit of the last character of the R-block requesting retransmission. The error coding bits (b4 – b1) of R-blocks shall be ignored.

  Note: Resynchronisation is not required by this specification. If for proprietar y reasons the terminal supports resynchronisation, it may attempt this by sending a S(RESYNCH request) to try to obtain a valid response (a S(RESYNCH response)) from the ICC before terminating the card session.

- If the ICC or terminal as receiver detects an overrun of data the block shall be treated as invalid. The 2nd, 3rd, 4th, and 6th bullets apply.

  Note: The receiver may detect an overrun of data by waiting CWT to ensure no further characters are received, or by checking the I/O line prior to transmitting. An overrun of data may go undetected by the receiver. Waiting CWT to detect overrun is not required by these specifications, and may adversely affect transmission speed.

- The ICC shall send a S(IFS request) block a maximum of three times in succession in an attempt to obtain a valid S(IFS response) from the TTL. After three unsuccessful attempts, it shall remain in receive mode.

- A S(ABORT request) shall not be sent by the terminal. If the terminal receives a S(ABORT request) from the ICC, it shall terminate the card session by initiating the deactivation sequence within 9,600 etus following reception of the leading edge of the start bit of the last character of the S(ABORT request) block.

  Note: Transaction abortion is not required by this specification. If an ICC or terminal supports abortion for proprietary reasons it may issue a S(ABORT request), but note that it will receive an invalid response if the receiver does not also support abortion. In this event the card session will be terminated according to the rules above. If a terminal optionally supporting abortion receives a S(ABORT request) from an ICC it may return a S(ABORT response) rather than terminating the card session.

5.3 Terminal Transport Layer (TTL)

This section describes the mechanism by which command and response APDUs are transported between the terminal and the ICC. APDUs are command or response messages, and since both command and response messages may contain data the TTL shall be capable of managing the four cases defined in section I-5.4. The construction of C-APDUs and R-APDUs are described in sections I-5.4.1 and I-5.4.2, respectively.

The C-APDU is passed from the TAL to the TTL where it is mapped in a manner appropriate to the transmission protocol to be used before being sent to the ICC. Following processing of the command by the ICC, data (if present) and status are returned by the ICC to the TTL, which maps it onto the R-APDU.

5.3.1 Transport of APDUs by T=0

This section describes the mapping of C-APDUs and R-APDUs, the mechanism for exchange of data between the TTL and the ICC, and the use of the GET RESPONSE command for retrieval of data from the ICC when case 2 or 4 commands are used.

5.3.1.1 Mapping of C-APDUs and R-APDUs and Data Exchange

The mapping of the C-APDU onto the T=0 command header is dependent upon the case of the command. The mapping of the data (if present) and status returned by the ICC onto the R-APDU is dependent upon the length of the data returned and the meaning of the status bytes.

Procedure bytes ‘61xx’ and ‘6Cxx’ are returned by the ICC to control exchanges between the TTL and the ICC, and should never be returned to the TAL. Command processing in the ICC is not complete if it has returned procedure bytes ‘61xx’ or ‘6Cxx’.

Note: For proprietary reasons, the TTL may in addition be capable of accepting data from the ICC without using the ’61’ and ‘6C’ procedure bytes. Such functionality is not required and is beyond the scope of these specifications.

Normal status on completion of processing a command is indicated if the ICC returns status bytes SW1 SW2 = ‘9000’ to the TTL. The TTL shall discontinue processing of a command on receipt of any other status (but not on receipt of procedure bytes ‘61xx’ and ‘6Cxx’) from the ICC. (For case 4 commands only, immediately following successful transmission of command data to the ICC, the TTL shall continue processing the command if warning status bytes (‘62xx’ or ‘63xx’) or application related status bytes (‘9xxx’ except ‘9000’) are received.)

The following descriptions of the mapping of data and status returned by the ICC onto the R-APDU are for information, and apply only after the ICC has completed processing of the command, successfully or otherwise, and all data (if present) has been returned by the ICC under the control of ‘61xx’ and ‘6Cxx’ procedure bytes. Detailed use of the INS, $\overline{\text{INS}}$, and ‘60’ procedure bytes is not described.

The status returned by the ICC shall relate to the most recently received command; where a GET RESPONSE command is used to complete the processing of a case 2 or case 4 command, any status returned by the ICC after receipt of the GET RESPONSE command shall relate to GET RESPONSE command, not to the case 2 or case 4 command which it completes.

5.3.1.1.1 Case 1

The C-APDU header is mapped onto the first four bytes of the T=0 command header, and P3 of the T=0 command header is set to ‘00’.

The flow of the exchange is as follows:

1. The TTL shall send the T=0 command header to the ICC.

2. On receipt of the command header the ICC, under normal or abnormal processing, shall return status to the TTL.

   (The ICC shall analyse the T=0 command header to determine whether it is processing a case 1 command or a case 2 command requesting all data up to the maximum length available.)

3. On receipt of status from the ICC, the TTL shall discontinue processing of the command.

See Annex A, section A1, for details of the exchanges between the TTL and the ICC.

The status returned to the TTL from the ICC after completion of processing of the command is mapped onto the mandatory trailer of the R-APDU without change.

5.3.1.1.2 Case 2

The C-APDU header is mapped onto the first four bytes of the T=0 command header, and length byte ‘Le’ from the conditional body of the C-APDU is mapped onto P3 of the T=0 command header. All case 2 commands issued according to this specification shall have Le set to ‘00’; see Part II of this specification.

The flow of the exchange is as follows:

1. The TTL shall send the T=0 command header to the ICC.

2. On receipt of the command header the ICC:

   (a) under normal processing shall return data and status to the TTL. The ICC shall use procedure bytes ‘6Cxx’ (and if required, procedure bytes ‘61xx’) to control the return of data.

   OR

   (b) under abnormal processing shall return status only to the TTL.

3. On receipt of the data (if present) and status from the ICC, the TTL shall discontinue processing the command.

See Annex A, section A2, for details of the exchanges between the TTL and the ICC, including use of the ‘61xx’ and ‘6Cxx’ procedure bytes.

The data (if present) and status returned to the TTL from the ICC after completion of processing of the command, or the status returned by the ICC that caused the TTL to discontinue processing of the command, are mapped onto the R-APDU as follows:

The data returned (if present) is mapped onto the conditional body of the R-APDU. If no data is returned, the conditional body of the R-APDU is left empty.

The status returned is mapped onto the mandatory trailer of the R-APDU without change.

5.3.1.1.3 Case 3

The C-APDU header is mapped onto the first four bytes of the T=0 command header, and length byte ‘Lc’ from the conditional body of the C-APDU is mapped onto P3 of the T=0 command header.

The flow of the exchange is as follows:

1. The TTL shall send the T=0 command header to the ICC.

2. On receipt of the command header, if the ICC:

   (a) returns a procedure byte, the TTL shall send the data portion of the conditional body of the C-APDU to the ICC under the control of procedure bytes returned by the ICC

   OR

   (b) returns status, the TTL shall discontinue processing of the command.

3. If processing was not discontinued in step 2(b), the ICC shall return status following receipt of the conditional body of the C-APDU and completion of processing the command.

4. On receipt of status from the ICC, the TTL shall discontinue processing the command.

See Annex A, section A3, for details of the exchanges between the TTL and the ICC.

The status returned to the TTL from the ICC after completion of processing of the command, or the status returned by the ICC that caused the TTL to discontinue processing of the command, is mapped onto the R-APDU without change.

5.3.1.1.4 Case 4

The C-APDU header is mapped onto the first four bytes of the T=0 command header, and length byte ‘Lc’ from the conditional body of the C-APDU is mapped onto P3 of the T=0 command header. All case 4 commands issued according to this specification shall have Le set to ‘00’; see Part II of this specification.

The flow of the exchange is as follows:

1. The TTL shall send the T=0 command header to the ICC.

2. On receipt of the command header, if the ICC:

   (a) returns a procedure byte, the TTL shall send the data portion of the conditional body of the C-APDU to the ICC under the control of procedure bytes returned by the ICC

   OR

   (b) returns status, the TTL shall discontinue processing of the command.

3. If processing was not discontinued in step 2(b), following receipt of the conditional body of the C-APDU, the ICC:

   (a) under normal processing, shall return procedure bytes ‘61xx’ to the TTL requesting the TTL to issue a GET RESPONSE command to retrieve the data from the ICC

OR

(b) under abnormal processing, shall return status only to the TTL.

4. On receipt of the procedure bytes or status returned in step 3, if the ICC:

   (a) returned ‘61xx’ procedure bytes as in step 3(a), the TTL shall send a GET RESPONSE command header to the ICC with P3 set to a value less than or equal to the value contained in the ‘xx’ byte of ‘61xx’ procedure bytes

   OR

   (b) returned status as in step 3(b) that indicates a warning (‘62xx’ or ‘63xx’), or which is application related (‘9xxx’ but not ‘9000’), the TTL shall send a GET RESPONSE command with Le=‘00’

   OR

   (c) returned status as in step 3(b) other than that described in step 4(b), the TTL shall discontinue processing of the command.

5. If processing was not discontinued in step 4(c), the GET RESPONSE command shall be processed according to the rules for case 2 commands in section I-5.3.1.1.2.

See Annex A, section A4, for details of the exchanges between the TTL and the ICC, including use of the ‘61xx’ and ‘6Cxx’ procedure bytes.

The data (if present) and status returned to the TTL from the ICC after completion of processing of the command, or the status returned by the ICC that caused the TTL to discontinue processing of the command, are mapped onto the R-APDU as follows:

The data returned (if present) is mapped onto the conditional body of the R-APDU. If no data is returned, the conditional body of the R-APDU is left empty.

The first status returned during processing of the entire case 4 command, including the GET RESPONSE command if used, is mapped onto the mandatory trailer of the R-APDU without change.

5.3.1.2 Use of Procedure Bytes ‘61xx’ and ‘6Cxx’

The ICC returns procedure bytes ‘61xx ‘and ‘6Cxx’ to the TTL to indicate to it the manner in which it should retrieve the data requested by the command currently being processed. These procedure bytes are only used when processing case 2 and 4 commands using T=0.

Procedure bytes ‘61xx’ instruct the TTL to issue a GET RESPONSE command to the ICC. P3 of the GET RESPONSE command header is set to ≤ ‘xx’.

Procedure bytes ‘6Cxx’ instruct the TTL to immediately resend the previous command header setting P3 = ‘xx’.

Usage of these procedure bytes during error free processing with case 2 and 4 commands is as follows. In the case of an error, the ICC may return status indicating error or warning conditions instead of the ‘61xx’ or ‘6Cxx’ response.

5.3.1.2.1 Case 2 Commands

1. If the ICC receives a case 2 command header and Le = ‘00’ or Le > Licc, it shall return

   (a) procedure bytes ‘6C Licc’ instructing the TTL to immediately resend the command header with P3 = Licc

   OR

   (b) status indicating a warning or error condition (but not SW1 SW2 = ‘90 00’)

2. If the ICC receives a case 2 command header and Le = Licc, it shall return

   (a) data of length Le (= Licc) under the control of the INS, $\overline{\text{INS}}$, or ‘60’ procedure bytes followed by the associated status

   OR

   (b) procedure bytes ‘61xx’ instructing the TTL to issue a GET RESPONSE command with a maximum length of ‘xx’

   OR

   (c) status indicating a warning or error condition (but not SW1 SW2 = ‘90 00’)

3. If the ICC receives a case 2 command header and Le < Licc it shall return

   (a) data of length Le under the control of the INS, $\overline{\text{INS}}$, or ‘60’ procedure bytes followed by procedure bytes ‘61xx’ instructing the TTL to issue a GET RESPONSE command with a maximum length of ‘xx’

   OR

   (b) procedure bytes ‘6C Licc’ instructing the TTL to immediately resend the command header with P3 = Licc

   OR

   (c) status indicating a warning or error condition (but not SW1 SW2 = ‘90 00’)

3(b) above is not valid response by the ICC to a GET RESPONSE command.

5.3.1.2.2 Case 4 Commands

1. If the ICC receives a case 4 command, after processing the data sent with the C-APDU, it shall return

   (a) procedure bytes ‘61 xx’ instructing the TTL to issue a GET RESPONSE command with a maximum length of ‘xx’

   OR

   (b) status indicating a warning or error condition (but not SW1 SW2 = ’90 00’)

The GET RESPONSE command so issued is then treated as described in section I-5.3.1.2.1 for case 2 commands.

5.3.1.3 GET RESPONSE Command

The GET RESPONSE command is issued by the TTL to obtain available data from the ICC when processing case 2 and 4 commands. It is employed only when the T=0 protocol type is in use.

The structure of the command message is shown in Table I-26:

CLA	‘00’
INS	‘C0’
P1	‘00’
P2	‘00’
Le	Maximum length of data expected

Table I-26 - Structure of Command Message

Following normal processing, the ICC returns status bytes SW1 SW2 = ‘9000’ and Licc bytes of data.

In the event that an error condition occurs, the coding of the error status bytes (SW1 SW2) is shown in Table I-27:

SW1	SW2	Meaning
‘62’	‘81’	Part of returned data may be corrupted
‘67’	‘00’	Length field incorrect
‘6A’	‘86’	P1 P2 ≠ ‘00’
‘6F’	‘00’	No precise diagnosis

Table I-27 - GET RESPONSE Error Conditions

5.3.2 Transportation of APDUs by T=1

The C-APDU is sent from the TAL to the TTL. The TTL maps the C-APDU onto the INF field of an I-block without change, and sends the I-block to the ICC.

Response data (if present) and status is returned from the ICC to the TTL in the INF field of an I-block. If the ICC returns status indicating normal processing (‘61xx’), a warning (‘62xx’ or ‘63xx’), which is application related (‘9xxx’), or is ‘9000’, it shall also return data (if available) associated with processing of the command. No data shall be returned with any other status. The contents of the INF field of the I-block are mapped onto the R-APDU without change and returned to the TAL.

Note: C-APDUs and response data/status may be chained over the INF fields of multiple blocks if required.

5.4 Application Layer

The application protocol consists of an ordered set of exchanges between the TAL and the TTL. Application protocols are defined in subsequent parts of this specification.

Each step in an application layer exchange consists of a command-response pair, where the TAL sends a command to the ICC via the TTL, and the ICC processes it and sends a response via the TTL to the TAL. Each specific command has a specific response. An APDU is defined as a command message or a response message.

Both command and response messages may contain data. Thus, four cases shall be managed by the transmission protocols via the TTL, as shown in Table I-28:

Case	Command Data	Response Data
1	Absent	Absent
2	Absent	Present
3	Present	Absent
4	Present	Present

Table I-28 - Definition of Cases for Data in APDUs

Note: When secure messaging is used only case 3 and case 4 commands exist since data (as a minimum, the MAC) is always sent to the ICC. When using secure messaging, case 1 commands will become case 3, and case 2 commands will become case 4.

5.4.1 C-APDU

The C-APDU consists of a mandatory header of four consecutive bytes denoted CLA, INS, P1, and P2, followed by a conditional body of variable length.

These mandatory header bytes are defined as follows:

- CLA: Instruction class; may take any value except ‘FF’.
- INS: Instruction code within the instruction class. The INS is only valid if the l.s. bit is 0, and the m.s. nibble is neither ‘6’ nor ‘9’.
- P1, P2: Reference bytes completing the INS.

Note: The full coding of the headers for each command is covered in Part II of this specification.

The conditional body consists of a string of bytes defined as follows:

- 1 byte, denoted by Lc, defining the number of data bytes to be sent in the C-APDU. The value of Lc may range from 1 to 255.
- String of bytes sent as the data field of the C-APDU, the number of bytes sent being as defined by Lc.
- 1 byte, denoted by Le, indicating the maximum number of data bytes expected in the R-APDU. The value of Le may range from 0 to 255; if Le = 0, the maximum number of bytes expected in the response is 256.

Note: The full coding of the data field of the conditional body for each command is covered in Part II of this specification.

Four cases of C-APDU structure are possible as defined in Table I-29:

Case	Structure
1	CLA INS P1 P2
2	CLA INS P1 P2 Le
3	CLA INS P1 P2 Lc Data
4	CLA INS P1 P2 Lc Data Le

Table I-29 - Cases of C-APDUs

5.4.2 R-APDU

The R-APDU is a string of bytes consisting of a conditional body followed by a mandatory trailer of two bytes denoted SW1 SW2.

The conditional body is a string of data bytes with a maximum length as defined by Le in the C-APDU.

The mandatory trailer indicates the status of the ICC after processing the command.

The coding of SW1 SW2 is defined in Part II of this specification.

THIS PAGE LEFT INTENTIONALLY BLANK

Part II

Data Elements and Commands

1. Data Elements and Files

An application in the ICC includes a set of items of information. These items of information may be accessible to the terminal after a successful application selection (see Part III of this specification).

An item of information is called a data element. A data element is the smallest piece of information that may be identified by a name, a description of logical content, a format, and a coding.

1.1 Data Elements Associated with Financial Transaction Interchange

The data element directory defined in Annex B, Table B-1 includes those data elements that may be used for financial transaction interchange. Data elements not specified in Annex B, Table B-1, are outside the scope of these specifications.

1.2 Data Objects

A data object consists of a tag, a length, and a value. A tag uniquely identifies a data object within the environment of an application. The length is the length of the value field of the data object. The value of a data object may consist either of a data element or of one or more data objects. When a data object encapsulates a data element, it is called a primitive data object. When a data object encapsulates one or more data objects, it is called a constructed data object. Specific tags are assigned to the constructed data objects with a specific meaning in the environment of an application according to this specification. The value field of such constructed data objects is a context-specific template. Rules for the coding of context-specific data objects and templates are given in Annex C.

Table B-1 in Annex B describes the mapping of data elements onto data objects and the mapping of data objects into templates when applicable.

Records are templates containing one or more primitive and/or constructed data objects.

The mapping of data objects into records is left to the discretion of the issuer and the manner in which data elements are to be used is described in the *ICC Application Specification for Payment Systems*.

Annex C defines the tags that are reserved by this specification for EMV, the payment systems and issuers. All ICC applications conforming to this specification shall comply with this coding and allocation scheme in accordance with ISO/IEC 7816-6.

1.2.1 Classes of Data Objects

Identification and coding of different classes of data objects are defined in Annex C. The tag definitions specified in that annex are according to ISO/IEC 8825 and ISO/IEC 7816 series and apply to applications conforming to this specification.

1.3 Files

The data objects contained in data files accessible from the ICC are stored in records. The file structure and referencing method depend on the purpose of the file. Structures and referencing methods are described in the following sections. The layout of the data files accessible from the ICC is left to the discretion of the issuer except for the directory files described in the following section.

1.3.1 File Structure

The file organisation applying to this specification is deduced from and complies with the basic organisations as defined in ISO/IEC 7816-4.

This part describes the file structure of the applications conforming to this specification, named Payment System Applications (PSA). Other applications conforming to ISO/IEC 7816-4 but not necessarily conforming to this specification may also be present in the ICC. They may be managed by the commands defined in this specification.

The path to the set of PSA in the ICC is enabled by explicitly selecting the Payment System Environment (PSE), when present. A successful selection of the PSE gives access to the directory structure as described in section II-1.3.1.4. The application selection process is described in Part III of this specification.

When the PSE is present, the PSA related files can be seen from the terminal as a tree structure accessible through a directory structure. Every branch of the tree is an application definition file (ADF) or a directory definition file (DDF). An ADF is the entry point to one or more application elementary files (AEFs). An ADF and its related data files are seen as being on the same branch of the tree. A DDF is an entry point to other ADFs or DDFs.

1.3.1.1 Application Definition Files

The tree structure of ADFs:

- Enables the attachment of data files to an application.
- Ensures the separation between applications.
- Allows access to the logical structure of an application by its selection.

An ADF is seen from the terminal as a file containing only data objects encapsulated in its file control information (FCI) as shown in Table II-31.

1.3.1.2 Application Elementary Files

An AEF in the range 1-10, contains one or more primitive Basic Encoding Rules - Tag Length Value (BER-TLV) data objects grouped into constructed BER-TLV data objects (records) according to Annex C. After selecting the application, an AEF in the range 1-10 is only referred to by its short file identifier (SFI) as described in section II-1.3.2.

A data file referred to in this specification consists of a sequence of records addressed by record number. The data files referred to by SFIs in the range 1-10 contain only data not interpreted by the card, that is, data that is not used by the card in its internal processes. This file structure is defined as linear. It can be either linear fixed or linear variable according to ISO/IEC 7816-4. The choice is left to the issuer and does not impact the reading of the file according to this specification.

1.3.1.3 Mapping of Files Onto ISO/IEC 7816-4 File Structure

The following mapping onto ISO/IEC 7816-4 applies:

- A dedicated file (DF) as defined in ISO/IEC 7816-4, containing a FCI is mapped onto an ADF or a DDF. It may give access to elementary files and DFs. The DF at the highest level of the card is the master file (MF).

- An elementary file (EF) as defined in ISO/IEC 7816-4, containing a set of records is mapped onto the AEF. An EF is never used as an entry point to another file.

If DFs are embedded, retrieval of the attached EF is transparent to this specification.

1.3.1.4 Directory Structure

When the PSE is present, the ICC shall maintain a directory structure for the list of applications within the PSE that the issuer wants to be selected by a directory. In that case, the directory structure consists of a payment system directory file (DIR file) and optional additional directories introduced by directory definition files (DDF) as described in this section.

The directory structure allows for the retrieval of an application using its Application Identifier (AID) or the retrieval of a group of applications using the first n-bytes of their AID as DDF name.

The presence of the DIR file shall be coded in the response message to the selection of the PSE (see the SELECT command).

The DIR file is an AEF (in other words, an EF with a record structure according to this specification) including the following data objects according to ISO/IEC 7816-5:

- One or more Application Templates (tag ‘61’) as described in Part III of this specification.

- Other data objects may be present within a Directory Discretionary Template (tag ‘73’). The data objects contained in this template are outside the scope of this specification.

Directories other than the payment system directory are optional within an ICC, but there is no defined limit to the number of such directories that may exist. Each such directory is located by a directory SFI data object contained in the FCI of each DDF.

1.3.2 File Referencing

A file may be referred to by a name or a SFI depending on its type.

1.3.2.1 Referencing by Name

Any ADF or DDF in the card is referenced by its DF name. A DF name for an ADF corresponds to the AID. Each DF name shall be unique within a given card.

1.3.2.2 Referencing by SFI

SFIs are used for the selection of AEFs. Any AEF within a given application is referenced by a SFI coded on 5 bits in the range 1 to 30. The coding of the SFI is described in every command that uses it.

The structure of a SFI is according to Table II-1:

Value	Meaning
1-10	Governed by this specification
11-20	Payment system specific
21-30	Issuer specific

Table II-1 - Structure of SFI

A SFI shall be unique within an application. The coding of SFIs within the range 1 to 10 is used to address AEFs governed by this specification.

1.4 Rules for Using a Data Object List (DOL)

In several instances, the terminal is asked to build a flexible list of data elements to be passed to the card under the card’s direction. To minimise processing within the ICC, such a list is generally not TLV encoded but is a single constructed field built by concatenating several data elements together. Since the elements of the constructed field are not TLV encoded, it is imperative that the ICC knows the format of this field when the data is received. This is achieved by including a Data Object List (DOL) in the ICC, specifying the format of the data to be included in the constructed field. DOLs currently used in this specification include the PDOL used with the GET PROCESSING OPTIONS command, CDOL1 and CDOL2 used with the GENERATE AC command, the TDOL used to generate a TC Hash Value, and the DDOL used with the INTERNAL AUTHENTICATE command. This section describes the rules for constructing a field using a DOL supplied by the card.

A DOL is a concatenated list of entries, with each entry representing a single data element to be included in the constructed field. The format of each entry is a one- or two-byte tag identifying the desired data object, followed by a one-byte length, representing the number of bytes the field shall occupy in the command data. Only tags representing primitive data objects defined in Annex B shall be used in DOLs.

The terminal shall complete the following steps to build the constructed field:

1. Read the DOL from the ICC.

2. Concatenate together all data elements listed in the DOL. The following rules apply to the creation of this concatenation:

   a) If the tag of any data object identified in the DOL is unknown to the terminal or represents a constructed data object, the terminal shall provide a data element with the length specified and a value of all hexadecimal zeroes.

   b) If a data object is in the list and is meaningful to the terminal but represents optional static data that is absent from the ICC, the portion of the command field representing the data object shall be filled with hexadecimal zeroes.

   c) If the length specified in the DOL entry is less than the length of the actual data object, the leftmost bytes of the data element shall be truncated if the data object has numeric (n) format, or the rightmost bytes of the data shall be truncated for any other format. If the length specified is greater than the actual data, the actual data shall be padded with leading hexadecimal zeroes if the data has numeric format, with hexadecimal FF's if the data has compressed numeric (cn) format, or with trailing hexadecimal zeroes for any other format.

   d) If a data object is in the list and is meaningful to the terminal but represents data that is not applicable to the current transaction, the portion of the command field representing the data object shall be filled with hexadecimal zeroes.

The completed list of data elements shall be concatenated in the sequence in which the corresponding data objects appear in the DOL.

2. Commands for Financial Transaction

2.1 Message Structure

Messages are transported between the terminal and the card according to the transmission protocol selected at the ATR (see Part I of this specification). The terminal and the card shall also implement the physical, data link, and transport layers as defined in Part I.

To run an application, an additional layer called application protocol is implemented in the terminal. It includes steps consisting of sending a command to the card, processing it in the card, and sending back the ICC response to the command. All commands and responses referred to in this part and further parts of this specification, are defined at the application layer.

The command message sent from the application layer and the response message returned by the card to the application layer are called Application Protocol Data Units (APDU). A specific response corresponds to a specific command. These are referred to as APDU command-response pairs. In an APDU command-response pair, the command message and the response message may contain data.

This section describes the structure and coding of the APDU command-response pairs necessary to the application protocols defined in this specification. Post-issuance commands defined in this section are optional. All other commands shall be implemented as required by specific applications.

2.1.1 Command APDU Format

The command APDU consists of a mandatory header of four bytes followed by a conditional body of variable length, as shown in Figure II-1:

<table>
<tr><td>CLA</td><td>INS</td><td>P1</td><td>P2</td><td>Lc</td><td>Data</td><td>Le</td></tr>
<tr><td colspan="4">← Mandatory Header →</td><td colspan="3">← Conditional Body →</td></tr>
</table>

Figure II-1 - Command APDU Structure

The number of data bytes sent in the command APDU is denoted by Lc (length of command data field).

The maximum number of data bytes expected in the response APDU is denoted by Le (length of expected data). When Le is present and contains the value zero, the maximum number of data bytes (256) is requested. When required in a command message, Le shall always be set to ‘00’.

The content of a command APDU message is as shown in Table II-2:

Code	Description	Length
CLA	Class of instruction	1
INS	Instruction code	1
P1	Instruction parameter 1	1
P2	Instruction parameter 2	1
Lc	Number of bytes present in command data field	0 or 1
Data	String of data bytes sent in command (= Lc)	var.
Le	Maximum number of data bytes expected in data field of response	0 or 1

Table II-2 - Command APDU Content

The different cases of command APDU structure are described in Part I of this specification.

2.1.2 Response APDU Format

The response APDU format consists of a conditional body of variable length followed by a mandatory trailer of two bytes, as shown in Figure II-2:

Data	SW1	SW2
← Body →	← Trailer →	

Figure II-2 - Response APDU Structure

The data field of a response APDU is an object structured as defined in Annex C. The detailed coding of the objects is provided with the commands described in subsequent sub-clauses.

The number of data bytes received in the response APDU is denoted by Lr (length of response data field). Lr is not returned by the transport layer. The application layer may rely on the object oriented structure of the response message data field to calculate Lr if needed.

The trailer codes on two bytes the processing state of the command as returned by the transport layer.

The content of a response APDU message is as shown in Table II-3:

Code	Description	Length
Data	String of data bytes received in response (= Lr)	var.
SW1	Command processing status	1
SW2	Command processing qualifier	1

Table II-3 - Response APDU Content

2.1.3 Command-Response APDU Conventions

In an APDU command-response pair, both the command message and the response message may contain data, thus resulting in four cases, as shown in Table II-4:

Case	Command Data	Response Data
1	Absent	Absent
2	Absent	Present
3	Present	Absent
4	Present	Present

Table II-4 - Data Within an APDU Command-Response Pair

These four cases are handled by the transmission protocol selected at the ATR according to Part I of this specification.

2.2 Coding Conventions

This section defines the coding of the header and the body of the messages (command and response).

2.2.1 Coding of the Class Byte

The most significant nibble of the class byte codes the type of command as shown in Table II-5:

Hex.	Meaning
'0'	Inter-industry command
'8'	Proprietary to this specification
Any other value	Outside the scope of this specification

Table II-5 - Most Significant Nibble of the Class Byte

A command proprietary to this specification is introduced by the most significant nibble of the class byte set to '8', in other words, the structure of the command and response messages is according to ISO/IEC 7816-4, the coding of the messages is defined within the context of the PSE.

The least significant nibble of the class byte codes secure messaging and logical channel mechanisms, according to ISO/IEC 7816-4**.**

2.2.2 Coding of the Instruction Byte

The INS byte of a command is structured according to Part I of this specification. The coding of INS and its relationship to CLA as shown in Table II-6 applies:

CLA	INS	Meaning
‘8x’	‘1E’	APPLICATION BLOCK
‘8x’	‘18’	APPLICATION UNBLOCK
‘8x’	‘16’	CARD BLOCK
‘0x’	‘82’	EXTERNAL AUTHENTICATE
‘8x’	‘AE’	GENERATE APPLICATION CRYPTOGRAM
‘0x’	‘84’	GET CHALLENGE
‘8x’	‘CA’	GET DATA
‘8x’	‘A8’	GET PROCESSING OPTIONS
‘0x’	‘88’	INTERNAL AUTHENTICATE
‘8x’	‘24’	PERSONAL IDENTIFICATION NUMBER (PIN) CHANGE/UNBLOCK
‘0x’	‘B2’	READ RECORD
‘0x’	‘A4’	SELECT
‘0x’	‘20’	VERIFY
‘8x’	‘Dx’	RFU for the payment systems
‘8x’	‘Ex’	RFU for the payment systems
‘9x’	‘xx’	RFU for manufacturers for proprietary INS coding
‘Ex’	‘xx’	RFU for issuers for proprietary INS coding

Table II-6 - Coding of the Instruction Byte

Note: Additional INS codes may be assigned in the future in the PSE context using the class ‘8x’. It is strongly recommended not to define proprietary commands in the class ‘8x’ when they are to be used in the PSE context, so that collision is avoided.

2.2.3 Coding of Parameter Bytes

The parameter bytes P1 P2 may have any value. If not used, a parameter byte has the value ‘00’.

2.2.4 Coding of Data Field Bytes

When present, an APDU command message data field consists of a string of data elements.

When present, an APDU response data field consists of a data object or a string of data objects encapsulated in a template according to Annex C.

2.2.5 Coding of the Status Bytes

The status bytes SW1 SW2 are returned by the transport layer to the application layer in any response message and denote the processing state of the command. The coding of the status words is structured according to Figure II-3:

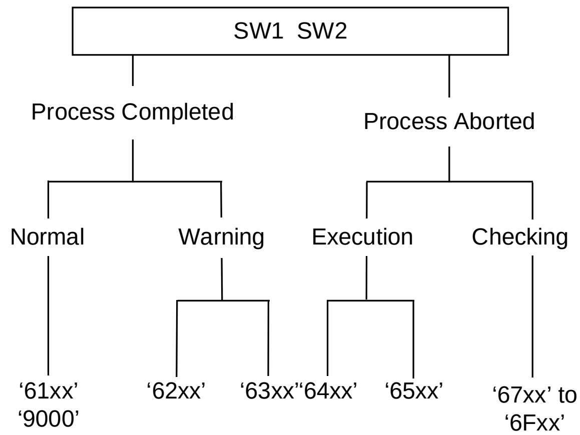

Figure II-3 - Structural Scheme of Status Bytes

Coding of SW1 SW2 is as shown in Table II-7 for those status bytes for which the *ICC Application Specification for Payment Systems* requires the card to return certain status bytes in response to a specific condition. The card may generate status bytes not listed in this table for error and warning conditions not specified in the *ICC Application Specification for Payment Systems*.

SW1	SW2	Meaning
		Normal processing
'90'	'00'	Process completed (any other value for SW2 is RFU)
		Warning processing
'62'	'83'	State of non-volatile memory unchanged; selected file invalidated
'63'	'00'	State of non-volatile memory changed; authentication failed
'63'	'Cx'	State of non-volatile memory changed; counter provided by 'x' (from 0-15)
		Checking errors
'69'	'83'	Command not allowed; authentication method blocked
'69'	'84'	Command not allowed; referenced data invalidated
'69'	'85'	Command not allowed; conditions of use not satisfied
'6A'	'81'	Wrong parameter(s) P1 P2; function not supported
'6A'	'82'	Wrong parameter(s) P1 P2; file not found
'6A'	'83'	Wrong parameter(s) P1 P2; record not found

Table II-7 - Coding of Status Bytes SW1 SW2

The following values of SW1 SW2 are described in Part I of this specification as they apply to the TPDU and are not returned to the APDU:

- '61xx': SW2 indicates the number of response bytes still available.
- '6Cxx': Wrong length Le, SW2 indicates the exact length.

SW1 = '6x' or '90' denotes a normal, warning, or error condition coded according to ISO/IEC 7816-4.

When using transmission protocol T=1, for all commands having Le = ‘00’ the successful execution of the command is coded by ’90 00’ or ’61 La’. However, for readability, both response codes are referenced throughout the text as ’90 00’ only.

Other values of SW1 returned by the ICC are not supported by Part II.

SW1	SW2	APPLICATION BLOCK	APPLICATION UNBLOCK	CARD BLOCK	EXTERNAL AUTHENTICATE	GENERATE APPLICATION CRYPTOGRAM	GET CHALLENGE	GET DATA	GET PROCESSING OPTIONS	INTERNAL AUTHENTICATE	PIN CHANGE/UNBLOCK	READ RECORD	SELECT	VERIFY
‘90’	‘00’	√	√	√	√	√	√	√	√	√	√	√	√	√
‘61’	‘La’					√		√	√	√		√	√	
‘62’	‘83’												√^{1}	
‘63’	‘00’				√^{2}									
‘63’	‘Cx’													√^{3}
‘69’	‘83’													√^{4}
‘69’	‘84’													√^{5}
‘69’	‘85’				√^{6}	√^{7}			√^{8}					
‘6A’	‘81’												√^{9}	
‘6A’	‘82’												√^{10}	
‘6A’	‘83’											√^{11}		

Table II-8 - Allocation of Status Bytes

The following convention is used in the table:

√ = Indicates normal processing

√^{x} = Allowed response code, for which a dedicated action shall be taken or which has a special meaning for an EMV compliant application. The meaning of the action is explained under the table

[grey box] = If this response code is returned by the card, the terminal will terminate the transaction

√^{1} = See section II-2.4.1.1 for appropriate processing.

√[2] = SW1 SW2 = ‘6300’ means ‘Authentication failed’ for the EXTERNAL AUTHENTICATE command.

√[3] = See section II-2.4.12.5 and section 7.5.1 of the *Integrated Circuit Card Application Specification for Payment Systems* for appropriate processing.

√[4] = See section II-2.4.12.5 and section 7.5.1 of the *Integrated Circuit Card Application Specification for Payment Systems* for appropriate processing.

√[5] = See section 7.5.1 of the *ICC Application Specification for Payment Systems* for appropriate processing.

√[6] = See section 7.9 of the *ICC Application Specification for Payment Systems* for appropriate processing.

√[7] = See section 8.3.2 of the *ICC Application Specification for Payment Systems*.

√[8] = See section 7.1 of the *ICC Application Specification for Payment Systems* and section I-2.2.1 of the *ICC Terminal Specification for Payment Systems* for appropriate processing.

√[9] = See section II-2.4.3.1 for appropriate processing.

√[10] = See section II-2.4.11.5 for appropriate processing.

√[11] = See section III-3.2 for appropriate processing.

If during the transaction processing as described in the *ICC Application Specification for Payment Systems*, the card returns another value for SW1 SW2 than specified in Table II-8, the transaction shall be terminated. An example is that the transaction would be terminated if the application reads records in a file that contains four records and the card returns SW1 SW2 = ‘6400’ in response to the READ RECORD command for record 5 instead of SW1 SW2 = ‘6A83’.

When using transmission protocol T=1, for all commands having Le = ‘00’, the successful execution of the command is coded by ‘90 00’ or ‘61 La’. However, for readability, both response codes are referenced throughout the text as ‘90 00’ only.

If during the processing of the GET DATA command, defined in section 2.4.6, the card returns an error condition, the terminal shall proceed as indicated in Section 7.6.3 of the *ICC Application Specification for Payment Systems* (for terminal velocity checking) or in section 2.2.4.1 of the *ICC Terminal Specification for Payment Systems* (for cardholder verification processing).

If during the processing of an issuer script command, as defined in section 7.10 of the *ICC Application Specification for Payment Systems*, the card returns a warning condition (SW1 SW2 = ‘62XX’ or ‘63xx’), the terminal shall continue with the next command from the Issuer Script (if any).

2.2.6 Coding of RFU Data

The coding of data (bits and bytes) indicated as RFU and marked as ‘x’ in the tables of the specifications shall be set to zero unless otherwise stated.

2.3 Logical Channels

A logical channel establishes and maintains the link between an application in the terminal and an application in the card.

A card may support more than one logical channel but only the basic logical channel is supported by this specification. This limits to one the number of concurrent applications according to this specification.

2.4 Commands

This section describes the following APDU command-response pairs:

- APPLICATION BLOCK (post-issuance command)
- APPLICATION UNBLOCK (post-issuance command)
- CARD BLOCK (post-issuance command)
- EXTERNAL AUTHENTICATE
- GENERATE APPLICATION CRYPTOGRAM
- GET CHALLENGE
- GET DATA
- GET PROCESSING OPTIONS
- INTERNAL AUTHENTICATE
- PIN CHANGE/UNBLOCK (post-issuance command)
- READ RECORD
- SELECT
- VERIFY

The post-issuance commands shall only be sent using script processing (see the *ICC Application Specification for Payment Systems*) and secure messaging as specified in Part IV of this specification.

2.4.1 APPLICATION BLOCK Command-Response APDUs

2.4.1.1 Definition and Scope

The APPLICATION BLOCK command is a post-issuance command that invalidates the currently selected application.

Following the successful completion of an APPLICATION BLOCK command:

- An invalidated application shall return the status bytes ‘Selected file invalidated’ (SW1 SW2 = ‘6283’) in response to a SELECT command.

- An invalidated application shall return only an AAC as AC in response to a GENERATE AC command.

2.4.1.2 Command Message

The APPLICATION BLOCK command message is coded according to Table II-9:

Code	Value
CLA	'8C' or '84'; coding according to the secure messaging specified in Part IV of this specification
INS	'1E'
P1	'00'; all other values are RFU
P2	'00'; all other values are RFU
Lc	Number of data bytes
Data	Message Authentication Code (MAC) data component; coding according to the secure messaging specified in Part IV of this specification
Le	Not present

Table II-9 - APPLICATION BLOCK Command Message

2.4.1.3 Data Field Sent in the Command Message

The data field of the command message contains the MAC data component coded according to the secure messaging format specified in Part IV of this specification.

2.4.1.4 Data Field Returned in the Response Message

The data field of the response message is not present.

2.4.1.5 Processing State Returned in the Response Message

A successful execution of the command is coded by '9000', independent whether the application was already invalidated or not.

2.4.2 APPLICATION UNBLOCK Command-Response APDUs

2.4.2.1 Definition and Scope

The APPLICATION UNBLOCK command is a post-issuance command that rehabilitates the currently selected application.

Following the successful completion of an APPLICATION UNBLOCK command, the restrictions imposed by the APPLICATION BLOCK command are removed.

2.4.2.2 Command Message

The APPLICATION UNBLOCK command message is coded according to Table II-10.

Code	Value
CLA	‘8C’ or ‘84’; coding according to the secure messaging specified in Part IV of this specification
INS	‘18’
P1	‘00’; all other values are RFU
P2	‘00’; all other values are RFU
Lc	Number of data bytes
Data	MAC data component; coding according to the secure messaging specified in Part IV of this specification
Le	Not present

Table II-10 - APPLICATION UNBLOCK Command Message

2.4.2.3 Data Field Sent in the Command Message

The data field of the command message contains the MAC data component coded according to the secure messaging format specified in Part IV of this specification.

2.4.2.4 Data Field Returned in the Response Message

The data field of the response message is not present.

2.4.2.5 Processing State Returned in the Response Message

A successful execution of the command is coded by ‘9000’, independent whether the application was invalidated or not.

2.4.3 CARD BLOCK Command-Response APDUs

2.4.3.1 Definition and Scope

The CARD BLOCK command is a post-issuance command that permanently disables all applications in the ICC.

The CARD BLOCK command shall disable all applications in the ICC, including applications that may be selected implicitly.

Following the successful completion of a CARD BLOCK command, all subsequent SELECT commands shall return the status bytes ‘Function not supported’ (SW1 SW2 = ‘6A81’) and perform no other action.

2.4.3.2 Command Message

The CARD BLOCK command message is coded according to Table II-11.

Code	Value
CLA	‘8C’ or ‘84’; coding according to the secure messaging specified in Part IV of this specification
INS	‘16’
P1	‘00’; all other values are RFU
P2	‘00’; all other values are RFU
Lc	Number of data bytes
Data	MAC data component; coding according to the secure messaging specified in Part IV of this specification
Le	Not present

Table II-11 - CARD BLOCK Command Message

2.4.3.3 Data Field Sent in the Command Message

The data field of the command message contains the MAC data component coded according to the secure messaging format specified in Part IV of this specification.

2.4.3.4 Data Field Returned in the Response Message

The data field of the response message is not present.

2.4.3.5 Processing State Returned in the Response Message

A successful execution of the command is coded by ‘9000’, independent of whether the card was already blocked or not.

2.4.4 EXTERNAL AUTHENTICATE Command-Response APDUs

2.4.4.1 Definition and Scope

The EXTERNAL AUTHENTICATE command asks the application in the ICC to verify a cryptogram.

The response from the ICC consists of returning the processing state of the command.

2.4.4.2 Command Message

The EXTERNAL AUTHENTICATE command message is coded according to Table II-12:

Code	Value
CLA	‘00’
INS	‘82’
P1	‘00’
P2	‘00’
Lc	8-16
Data	Issuer Authentication Data
Le	Not present

Table II-12 - EXTERNAL AUTHENTICATE Command Message

The reference of the algorithm (P1) of the EXTERNAL AUTHENTICATE command is coded ‘00’, which means that no information is given. The reference of the algorithm is known either before issuing the command or is provided in the data field.

2.4.4.3 Data Field Sent in the Command Message

The data field of the command message contains the value field of tag ‘91’ coded as follows:

- Mandatory first 8 bytes containing the cryptogram.
- Optional additional 1-8 bytes are proprietary.

2.4.4.4 Data Field Returned in the Response Message

The data field of the response message is not present.

2.4.4.5 Processing State Returned in the Response Message

A successful execution of the command is coded by ‘9000’.

2.4.5 GENERATE APPLICATION CRYPTOGRAM Command-Response APDUs

2.4.5.1 Definition and Scope

The GENERATE AC command sends transaction-related data to the ICC, which computes and returns a cryptogram. This cryptogram shall either be an Application Cryptogram (AC) as specified in this specification or a proprietary cryptogram. In both cases, the cryptogram shall be of a type specified in Table II-13 (for more details, see the *ICC Application Specification for Payment Systems*).

Type	Meaning
Application Authentication Cryptogram (AAC)	Transaction declined
Application Authorisation Referral (AAR)	Referral requested by the card
Authorisation Request Cryptogram (ARQC)	Online authorisation requested
Transaction Certificate (TC)	Transaction approved

Table II-13 - GENERATE AC Cryptogram Types

The cryptogram returned by the ICC may differ from that requested in the command message according to an internal process in the ICC (as described in the *ICC Application Specification for Payment Systems*).

2.4.5.2 Command Message

The GENERATE AC command message is coded according to Table II-14:

Code	Value
CLA	'80'
INS	'AE'
P1	Reference control parameter (see Table II-15)
P2	'00'
Lc	var.
Data	Transaction-related data
Le	'00'

Table II-14 - GENERATE AC Command Message

The reference control parameter of the GENERATE AC command is coded as shown in Table II-15:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0							AAC
0	1							TC
1	0							ARQC
1	1							RFU
		x	x	x	x	x	x	RFU

Table II-15 - GENERATE AC Reference Control Parameter

2.4.5.3 Data Field Sent in the Command Message

The content of the data field of the command message is coded according to the rules for the data object list as defined in Section II-1.4.

2.4.5.4 Data Field Returned in the Response Message

The data field of the response message consists of a BER-TLV coded data object. The coding of the data object shall be according to one of the following two formats.

- **Format 1:** The data object returned in the response message is a primitive data object with tag equal to '80'. The value field consists of the concatenation without delimiters (tag and length) of the value fields of the data objects specified in Table II-16:

Value	Presence
Cryptogram Information Data	M
Application Transaction Counter (ATC)	M
Application Cryptogram (AC)	M
Issuer Application Data	O

Table II-16 - Format 1 GENERATE AC Response Message Data Field

- **Format 2:** The data object returned in the response message is a constructed data object with tag equal to '77'. The value field may contain several BER-TLV coded objects, but shall always include the Cryptogram Information Data, the Application Transaction Counter and the cryptogram computed by the ICC (either an AC or a proprietary cryptogram). The utilization and interpretation of proprietary data objects which may be included in this response message are outside the scope of these specifications.

For both formats, the Cryptogram Information Data returned by the GENERATE AC response message is coded according to Table II-17:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0							AAC
0	1							TC
1	0							ARQC
1	1							AAR
		x	x					RFU
				0				No advice required
				1				Advice required
					x	x	x	Reason/advice/referral code
					0	0	0	No information given
					0	0	1	Service not allowed
					0	1	0	PIN Try Limit exceeded
					0	1	1	Issuer authentication failed
					x	x	x	Other values RFU

Table II-17 - Coding of Cryptogram Information Data

2.4.5.5 Processing State Returned in the Response Message

A successful execution of the command is coded by ‘9000’.

2.4.6 GET DATA Command-Response APDUs

2.4.6.1 Definition and Scope

The GET DATA command is used to retrieve a primitive data object not encapsulated in a record within the current application.

The usage of the GET DATA command in this specification is limited to the retrieval of the primitive data objects ATC (tag ‘9F36’), Last Online ATC Register (tag ‘9F13’), or PIN Try Counter (tag ‘9F17’) defined in Annex B, Table B-1 that are interpreted by the application in the ICC.

2.4.6.2 Command Message

The GET DATA command message is coded according to Table II-18:

Code	Value
CLA	‘80’
INS	‘CA’
P1 P2	‘9F36’, ‘9F13’, or ‘9F17’
Lc	Not present
Data	Not present
Le	‘00’

Table II-18 - GET DATA Command Message

2.4.6.3 Data Field Sent in the Command Message

The data field of the command message is not present.

2.4.6.4 Data Field Returned in the Response Message

The data field of the response message contains the primitive data object referred to in P1 P2 of the command message (in other words, including its tag and its length).

2.4.6.5 Processing State Returned in the Response Message

A successful execution of the command is coded by ‘9000’.

2.4.7 GET PROCESSING OPTIONS Command-Response APDUs

2.4.7.1 Definition and Scope

The GET PROCESSING OPTIONS command initiates the transaction within the ICC.

The response from the ICC consists of returning the Application Interchange Profile (AIP) and the Application File Locator (AFL).

2.4.7.2 Command Message

The GET PROCESSING OPTIONS command message is coded according to Table II-19:

Code	Value
CLA	'80'
INS	'A8'
P1	'00'; all other values are RFU
P2	'00'; all other values are RFU
Lc	var.
Data	Processing Options Data Object List (PDOL) related data
Le	'00'

Table II-19 - GET PROCESSING OPTIONS Command Message

2.4.7.3 Data Field Sent in the Command Message

The data field of the command message is a data object coded according to the Processing Options Data Object List (PDOL) provided by the ICC, as defined in section II-1.4, and is introduced by the tag '83'. When the data object list is not provided by the ICC, the length field of the template is set to zero. Otherwise, the length field of the template is the total length of the value fields of the data objects transmitted to the ICC.

2.4.7.4 Data Field Returned in the Response Message

The data field of the response message consists of a BER-TLV coded data object. The coding of the data object shall be according to one of the following two formats.

- **Format 1:** The data object returned in the response message is a primitive data object with tag equal to '80'. The value field consists of the concatenation without delimiters (tag and length) of the value fields of the Application Interchange Profile (AIP) and the Application File Locator (AFL). The coding of the data object returned in the response message is shown in Table II-20:

‘80’	Length	Application Interchange Profile	AFL

Table II-20 - Format 1 GET PROCESSING OPTIONS Response Message Data Field

- **Format 2:** The data object returned in the response message is a constructed data object with tag equal to ‘77’. The value field may contain several BER-TLV coded objects, but shall always include the AIP and the AFL. The utilization and interpretation of proprietary data objects which may be included in this response message are outside the scope of these specifications.

The AIP specifies the application functions that are supported by the application in the ICC and is coded according to the *ICC Application Specification for Payment Systems*.

The AFL consists of the list without delimiters of files and related records that shall be read according to the *ICC Application Specification for Payment Systems* for the currently selected application.

2.4.7.5 Processing State Returned in the Response Message

A successful execution of the command is coded by ‘9000’.

2.4.8 INTERNAL AUTHENTICATE Command-Response APDUs

2.4.8.1 Definition and Scope

The INTERNAL AUTHENTICATE command initiates the computation of the Signed Dynamic Application Data by the card using the challenge data sent from the IFD and data and a relevant private key stored in the card.

The response from the ICC consists of returning the Signed Dynamic Application Data to the terminal.

2.4.8.2 Command Message

The INTERNAL AUTHENTICATE command message is coded according to Table II-21:

Code	Value
CLA	‘00’
INS	‘88’
P1	‘00’
P2	‘00’
Lc	Length of authentication-related data
Data	Authentication-related data
Le	‘00’

Table II-21 - INTERNAL AUTHENTICATE Command Message

The reference of the algorithm (P1) of the INTERNAL AUTHENTICATE command is coded ‘00’, which means that no information is given. The reference of the algorithm is known either before issuing the command or is provided in the data field.

2.4.8.3 Data Field Sent in the Command Message

The data field of the command message contains the authentication-related data proprietary to an application. It is coded according to the Dynamic Data Authentication Data Object List (DDOL) as defined in Part IV of this specification.

2.4.8.4 Data Field Returned in the Response Message

The data field of the response message consists of a BER-TLV coded data object. The coding of the data object shall be according to one of the following two formats.

- **Format 1:** The data object returned in the response message is a primitive data object with tag equal to ‘80’. The value field consists of the value field of the Signed Dynamic Application Data as specified in Part IV of this specification.

- **Format 2:** The data object returned in the response message is a constructed data object with tag equal to ‘77’. The value field may contain several BER-TLV coded objects, but shall always include the Signed Dynamic Application Data as specified in Part IV of this specification. The utilization and interpretation of proprietary data objects which may be included in this response message are outside the scope of these specifications.

2.4.8.5 Processing State Returned in the Response Message

A successful execution of the command is coded by ‘9000’’.

2.4.9 PIN CHANGE/UNBLOCK Command-Response APDUs

2.4.9.1 Definition and Scope

The PIN CHANGE/UNBLOCK command is a post-issuance command. Its purpose is to provide the issuer the capability either to unblock the PIN or to simultaneously change and unblock the reference PIN.

Upon successful completion of the PIN CHANGE/UNBLOCK command, the card shall perform the following functions:

- The value of the PIN Try Counter shall be reset to the value of the PIN Try Limit.

- If requested, the value of the reference PIN shall be set to the new PIN value.

The PIN data transmitted in the command, if present, shall be enciphered for confidentiality.

Note: The reference PIN, which is stored within the card, is the one that is associated wi th the application, and which the card uses to the check the Transaction PIN Data transmitted within the VERIFY command.

2.4.9.2 Command Message

The PIN CHANGE/UNBLOCK command message is coded according to Table II-22.

Code	Value
CLA	'8C' or '84'; coding according to the secure messaging specified in Part IV of this specification
INS	'24'
P1	'00'
P2	'00', '01', or '02'
Lc	Number of data bytes
Data	Enciphered PIN data component, if present, and MAC data component; coding according to the secure messaging specified in Part IV of this specification
Le	Not present

Table II-22 - PIN CHANGE/UNBLOCK Command Message

P2: If P2 is equal to '00', the reference PIN is unblocked and the PIN Try Counter is reset to the PIN Try Limit. There is no PIN update, since the PIN CHANGE/UNBLOCK command does not contain a new PIN value.

Any other value of P2 allowing PIN unblocking and/or PIN changing is out of the scope of this specification and shall be described for each payment system individually.

2.4.9.3 Data Field Sent in the Command Message

The data field of the command message contains the PIN data component, if present, followed by the MAC data component coded according to the secure messaging format specified in Part IV of this specification.

2.4.9.4 Data Field Returned in the Response Message

The data field of the response message is not present.

2.4.9.5 Processing State Returned in the Response Message

A successful execution of the command is coded by '9000'.

2.4.10 READ RECORD Command-Response APDUs

2.4.10.1 Definition and Scope

The READ RECORD command reads a file record in a linear file.

The response from the ICC consists of returning the record.

2.4.10.2 Command Message

The READ RECORD command message is coded according to Table II-23:

Code	Value
CLA	‘00’
INS	‘B2’
P1	Record number
P2	Reference control parameter (see Table II-24)
Lc	Not present
Data	Not present
Le	‘00’

Table II-23 - READ RECORD Command Message

Table II-24 defines the reference control parameter of the command message:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	x	x	x				SFI
					1	0	0	P1 is a record number

Table II-24 - READ RECORD Command Reference Control Parameter

2.4.10.3 Data Field Sent in the Command Message

The data field of the command message is not present.

2.4.10.4 Data Field Returned in the Response Message

The data field of the response message of any successful READ RECORD command contains the record read. For SFIs in the range 1-10, the record is a BER-TLV constructed data object as defined in Annex C and coded as shown in Table II-25:

‘70’	Length	Record Template

Table II-25 - READ RECORD Response Message Data Field

The response message to READ RECORD for SFIs outside the range 1-10 is outside the scope of this specification.

2.4.10.5 Processing State Returned in the Response Message

A successful execution of the command is coded by ‘9000’.

2.4.11 SELECT Command-Response APDUs

2.4.11.1 Definition and Scope

The SELECT command is used to select the ICC PSE, DDF, or ADF corresponding to the submitted file name or AID. The selection of an application is described in Part III of this specification.

A successful execution of the command sets the path to the PSE, DDF, or ADF.

Subsequent commands apply to AEFs associated to the selected PSE, DDF, or ADF using SFIs.

The response from the ICC consists of returning the FCI.

2.4.11.2 Command Message

The SELECT command message is coded according to Table II-26:

Code	Value
CLA	‘00’
INS	‘A4’
P1	Reference control parameter (see Table II-27)
P2	Selection options (see Table II-28)
Lc	‘05’-‘10’
Data	File name
Le	‘00’

Table II-26 - SELECT Command Message

Table II-27 defines the reference control parameter of the SELECT command message:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0				
					1			Select by name
						0	0	

Table II-27 - SELECT Command Reference Control Parameter

Table II-27 defines the selection options P2 of the SELECT command message:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
						0	0	First or only occurrence
						1	0	Next occurrence

Table II-28 - SELECT Command Options Parameter

2.4.11.3 Data Field Sent in the Command Message

The data field of the command message contains the PSE name or the DF name or the AID to be selected.

2.4.11.4 Data Field Returned in the Response Message

The data field of the response message contains the FCI specific to the selected PSE, DDF, or ADF. The tags defined in Table II-29 to Table II-31 apply to this specification. Additional tags returned in the FCI that are not described in this specification shall be ignored.

Table II-29 defines the FCI returned by a successful selection of the PSE:

Tag	Value	Presence
'6F'	FCI Template	M
'84'	DF Name	M
'A5'	FCI Proprietary Template	M
'88'	SFI of the directory elementary file	M
'5F2D'	Language Preference	O
'9F11'	Issuer Code Table Index	O
'BF0C'	FCI Issuer Discretionary Data	O

Table II-29 - SELECT Response Message Data Field (FCI) of the PSE

Table II-30 defines the FCI returned by a successful selection of a DDF:

Tag	Value	Presence
'6F'	FCI Template	M
'84'	DF Name	M
'A5'	FCI Proprietary Template	M
'88'	SFI of the directory elementary file	M
'BF0C'	FCI Issuer Discretionary Data	O

Table II-30 - SELECT Response Message Data Field (FCI) of a DDF

Table II-31 defines the FCI returned by a successful selection of an ADF:

Tag	Value	Presence
‘6F’	FCI Template	M
‘84’	DF Name	M
‘A5’	FCI Proprietary Template	M
‘50’	Application Label	O
‘87’	Application Priority Indicator	O
‘9F38’	PDOL	O
‘5F2D’	Language Preference	O
‘9F11’	Issuer Code Table Index	O
‘9F12’	Application Preferred Name	O
‘BF0C’	FCI Issuer Discretionary Data	O

Table II-31 - SELECT Response Message Data Field (FCI) of an ADF

2.4.11.5 Processing State Returned in the Response Message

A successful execution of the command is coded by ‘9000’.

ICC support for the selection of a DF file using only a partial DF name is not mandatory. However, if the ICC does support partial name selection, it shall comply with the following:

If the SELECT command having the coding of b2 and b1 set to ‘10’ (next) is repeated with the same partial DF name by the terminal after a DF file has been successfully selected, the card shall select a different DF file matching the partial name, if such another DF file exists. Repeated issuing of the same command with no intervening application level commands shall retrieve all such files, but shall retrieve no file twice. After all matching DF files have been selected, repeating the same command again shall result in no file being selected, and the card shall respond with SW1 SW2 = ‘6A82’ (file not found).

2.4.12 VERIFY Command-Response APDUs

2.4.12.1 Definition and Scope

The VERIFY command initiates in the ICC the comparison of the Transaction PIN Data sent in the data field of the command with the reference PIN data associated with the application. The manner in which the comparison is performed is proprietary to the application in the ICC.

The VERIFY command applies when the Cardholder Verification Method (CVM) chosen from the CVM List is an offline PIN, as described in the *ICC Application Specification for Payment Systems*.

2.4.12.2 Command Message

The VERIFY command message is coded according to Table II-32:

Code	Value
CLA	‘00’
INS	‘20’
P1	‘00’
P2	Qualifier of the reference data (see Table II-33)
Lc	var.
Data	Transaction PIN Data
Le	Not present

Table II-32 - VERIFY Command Message

Table II-33 defines the qualifier of the reference data (P2):

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	0	0	0	As defined in ISO/IEC 7816-4
1	0	0	0	0	0	0	0	Plaintext PIN, format as defined below
1	0	0	0	0	x	x	x	RFU for this specification
1	0	0	0	1	0	0	0	Enciphered PIN, format as defined in section IV-4
1	0	0	0	1	0	x	x	RFU for this specification
1	0	0	0	1	1	x	x	RFU for the individual payment systems
1	0	0	1	x	x	x	x	RFU for the issuer

Table II-33 - VERIFY Command Qualifier of Reference Data (P2)

The processing of the VERIFY command in the ICC will be defined in combination with the CVM rules as specified in the *ICC Application Specification for Payment Systems*.

The plaintext offline PIN block shall be formatted as follows:

C	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

where

	Name	Value
C	Control field	Binary 2 (hex. 0010)
N	PIN length	4-bit binary number with permissible values of hex. 0100 to hex. 1100
P	PIN digit	4-bit field with permissible values of hex. 0000 to hex. 1001
P/F	PIN/filler	Determined by PIN length
F	Filler	4-bit binary number with value of hex. 1111

P2 = ‘00’ indicates that no particular qualifier is used. The processing of the VERIFY command in the ICC should know how to find the PIN data unambiguously.

2.4.12.3 Data Field Sent in the Command Message

The data field of the command message contains the value field of tag ‘99’.

2.4.12.4 Data Field Returned in the Response Message

The data field of the response message is not present.

2.4.12.5 Processing State Returned in the Response Message

A successful execution of the command is coded by ‘9000’.

When for the currently selected application the comparison between the Transaction PIN Data and the reference PIN data performed by the VERIFY command fails, the ICC shall return SW2 = ‘Cx’, where ‘x’ represents the number of retries still possible. When the card returns ‘C0’, no more retries are left, and the CVM shall be blocked. Any subsequent VERIFY command applied in the context of that application shall then fail with SW1 SW2 = ‘6983’.

2.4.13 GET CHALLENGE Command-Response APDUs

2.4.13.1 Definition and Scope

The GET CHALLENGE ommand is used to obtain an unpredictable number from the ICC for use in a security-related procedure.

The challenge shall be valid only for the next issued command.

2.4.13.2 Command Message

The GET CHALLENGE command message is coded according to Table II-32:

Code	Value
CLA	‘00’
INS	‘84’
P1	‘00’
P2	‘00’
Lc	Not present
Data	Not present
Le	’00’

Table II-34 - GET CHALLENGE Command Message

2.4.13.3 Data Field Sent in the Command Message

The data field of the command message is not present.

2.4.13.4 Data Field Returned in the Response Message

The data field of the response message contains an 8-byte unpredictable number generated by the ICC.

2.4.13.5 Processing State Returned in the Response Message

A successful execution of the command is coded by ‘9000’.

Part III

Application Selection

1. Overview of Application Selection

Application selection is the process performed immediately after the reset of the card and prior to the first application function.

This section describes the application selection process from the standpoint of both the card and the terminal. The logical structure of data and files within the card that are required for the process is specified, after which the terminal logic using the card structure is described.

The ICC and the terminal may support and use implicit selection, but it is not described here as it is not useful in an interchange environment.

The application selection process described in this section is the process by which the terminal uses data in the ICC according to protocols defined herein to determine the terminal program and the ICC application to be used in processing a transaction. The process is described in two steps:

1. Create a list of ICC applications that are supported by the terminal. (This list is referred to below using the name 'candidate list.') This process is described in section III-3.

2. Select the application to be run from the list generated above. This process is described in section III-4.

It is the intent of this part of the specification to describe the necessary information in the card and two terminal selection algorithms that yield the correct results. Other terminal selection algorithms that yield the same results are permitted in place of the selection algorithms described here.

A payment system application is comprised of the following:

- A set of files in the ICC providing data customised by the issuer.
- Data in the terminal provided by the acquirer or the merchant.
- An application protocol agreed upon by both the ICC and the terminal.

Applications are uniquely identified by AIDs conforming to ISO/IEC 7816-5 (see section III-2.13).

The techniques chosen by the payment systems and described herein are designed to meet the following key objectives:

- Ability to work with ICCs with a wide range of capabilities.

- Ability for terminals with a wide range of capabilities to work with all ICCs supporting payment system applications according to this specification.

- Conformance with ISO standards.

- Ability of ICCs to support multiple applications, not all of which need to be payment system applications.

- Ability for ICCs to provide multiple sets of applications to be supported by a single terminal program. (For example, a card may contain multiple credit/debit applications, each representing a different type or level of service or a different account).

- As far as possible, provide the capability for applications conforming with this specification to co-reside on cards with presently existing applications.

- Minimum overhead in storage and processing.

- Ability for the issuer to optimise the selection process.

The set of data that the ICC contains in support of a given application is defined by an ADF selected by the terminal using a SELECT command and an AFL returned by the ICC in response to a GET PROCESSING OPTIONS command.

2. Data in the ICC Used for Application Selection

2.1 Coding of Payment System Application Identifier

The structure of the AID is according to ISO/IEC 7816-5 and consists of two parts:

1. A Registered Application Provider Identifier (RID) of 5 bytes, unique to an application provider and assigned according to ISO/IEC 7816-5.

2. An optional field assigned by the application provider of up to 11 bytes. This field is known as a Proprietary Application Identifier Extension (PIX) and may contain any 0-11 byte value specified by the provider. The meaning of this field is defined only for the specific RID and need not be unique across different RIDs.

Additional ADFs defined under the control of other application providers may be present in the ICC but shall avoid duplicating the range of RIDs assigned to payment systems. Compliance with ISO/IEC 7816-5 will assure this avoidance.

2.2 Structure of the Payment Systems Environment

The Payment Systems Environment (PSE) begins with a Directory Definition File (DDF) given the name '1PAY.SYS.DDF01'. The presence of this DDF in the ICC is optional but, if present, shall comply with this specification. If it is present, this DDF is mapped onto a DF within the card, which may or may not be the MF. As with all DDFs, this DDF shall contain a Payment Systems Directory. The FCI of this DDF shall contain at least the information defined for all DDFs in Part II and, optionally, the Language Preference (tag '5F2D') and the Issuer Code Table Index (tag '9F11').

The Language Preference and Issuer Code Table Index are optional data objects that may occur in two places: the FCI of the PSE and the FCI of ADF files. If these data objects exist, they shall exist in both places, and shall have the identical values in all occurrences. The terminal may use the values from either location.[9]

The directory attached to this initial DDF contains entries for ADFs that are formatted according to this specification, although the applications defined by those ADFs may or may not conform to this specification. The directory may also contain entries for other payment system's DDFs, which shall conform to this specification.

The directory is not required to have entries for all DDFs and ADFs in the card, and following the chain of DDFs may not reveal all applications supported by the card. However, if the PSE exists, only applications that are revealed by following the chain of DDFs beginning with the initial directory can be assured of international interoperability.

See Annex D for examples of the internal logic structure of an ICC containing the PSE.

2.3 Coding of a Payment System's Directory

A Payment System's Directory (hereafter referred to as simply a directory) is a linear EF file identified by an SFI in the range 1 to 10. The SFI for the directory is contained in the FCI of the DDF to which the directory is attached. The directory is read using the READ RECORD command as defined in Part II of this specification. A record may have several entries, but a single entry shall always be encapsulated in a single record.

Records in the Payment Systems Directory conform to all other requirements in this specification. Each record is a constructed data object, and the value field is comprised of one or more directory entries as described below. Each record is formatted as shown in Table III-1:

[9] A terminal building a candidate list using the process described in section 3.2 will encounter the values specified in the FCI of the PSE and will not see the values specified in the FCI of the ADF until the application to be run has been chosen. A terminal building the candidate list using the process described in section 3.3 will encounter the values specified in the FCI of the ADFs. To ensure consistent interface to the cardholder, the values must be the same.

Tag '70'	Data Length (L)	Tag '61'	Length of directory entry 1	Directory entry 1 (ADF or DDF)	...	Tag '61'	Length of directory entry n	Directory entry n (ADF or DDF)

Table III-1 - PSE Directory Record Format

Each entry in a Payment Systems Directory is the value field of an Application Template (tag '61') and contains the information according to Table III-2 or Table III-3.

If any data objects that are not encapsulated in an Application Template (tag '61') appear in the directory record or any data objects other than those listed in Table III-2 or Table III-3 appear in a directory entry they shall be ignored.

Tag	Length	Value	Presence
'9D'	5-16	DDF Name	M
'73'	var.	Directory Discretionary Template	O[10]

Table III-2 - DDF Directory Entry Format

Tag	Length	Value	Presence
'4F'	5-16	ADF Name (AID)	M
'50'	1-16	Application Label	M
'9F12'	1-16	Application Preferred Name	O
'87'	1	Application Priority Indicator (see Table III-)	O
'73'	var.	Directory Discretionary Template	O[10]

Table III-3 - ADF Directory Entry Format

b8	b7-b5	b4-b1	Definition
1			Application cannot be selected without confirmation of cardholder
0			Application may be selected without confirmation of cardholder
	xxx		RFU
		0000	No priority assigned
		xxxx (except 0000)	Order in which the application is to be listed or selected, ranging from 1-15, with 1 being highest priority

Table III-4 - Format of Application Priority Indicator

[10] Other data objects not relevant to this specification may appear in this constructed data object.

2.4 Coding of Other Directories

Each directory in an ICC is contained by a separate DDF. DDFs and directories in the card are optional, but there is no defined limit to the number that may exist. Each directory is located by a Directory SFI data object which must be contained in the FCI of the DDF (see Part II for the SELECT command). The low order five bits of the Directory SFI contain the SFI to be used in READ RECORD commands for reading the directory. The SFI shall be valid for reading the directory when the DDF containing the directory is the current file selected.

All directories, including the initial directory, have the same format, as described in section III-2.3.

3. Building the Candidate List

The terminal shall maintain a list of applications supported by the terminal and their AIDs. This section describes two procedures for determining which of those applications is to be run. If the card contains no PSE, the procedure described in section 3.3 must be followed.

3.1 Matching Terminal Applications to ICC Applications

The terminal determines which applications in the ICC are supported by comparing the AIDs for applications in the terminal with AIDs for applications within the ICC. In some cases, the terminal supports the ICC application only if the AID in the terminal has the same length and value as the AID in the ICC. This case limits the ICC to at most one matching ADF.

In other cases, the terminal supports the ICC application if the AID in the ICC begins with the entire AID kept within the terminal. This allows the ICC to have multiple ADFs matching the terminal application by adding unique information to the AID used by each of the ADFs. If the card has only one ADF matching the terminal AID, it should identify that ADF with the exact AID known to the terminal. If the ICC has multiple ADFs supported by a single terminal AID, the following requirements must be met by the ICC:

- The ICC must support partial name selection as described in Part II of this specification (see SELECT command).

- All of the matching AIDs in the ICC must be distinguished by adding unique data to the PIX. None of the ICC AIDs shall be the same length as the AID in the terminal.

For each of the AIDs within the list of applications supported by the terminal, the terminal shall keep an indication of which matching criterion to use.

3.2 Using the Payment Systems Directories

If a terminal chooses to support the Payment System directory, it shall follow the procedure described in this section to determine the applications supported by the card. Figure III-1 is a flow diagram of the logic described here.

The steps the terminal takes to use the directories are as follows:

1. The terminal begins with an explicit selection of the Payment Systems Environment using a SELECT command as described in Part II and a file name of '1PAY.SYS.DDF01'. This establishes the payment systems environment and makes the initial directory accessible.

   If there is no PSE in the ICC, the ICC shall return '6A82' ('File not found') in response to the SELECT command for the PSE. In this case, the terminal shall use the list of applications method described in section 3.3.

   If the ICC returns SW1 SW2 other than '9000' or '6A82' to the SELECT command for the PSE, the terminal shall terminate the card session.

2. The terminal reads all the records in the directory beginning with record number 1 and continuing with successive records until the card returns SW1 SW2 = '6A83', which indicates that the record number requested does not exist. (The card must return '6A83' if the record number in the READ RECORD command is greater than the number of the last record in the file). If the card returns SW1 SW2 = '6A83' in response to a READ RECORD for record number 1, no directory entries exist, and step 6 (below) applies.

   For each record, the terminal begins with the first entry and processes each entry in turn as described in steps 3 through 5.

3. If the entry is for an ADF and the ADF name matches one of the applications supported by the terminal as defined in section 3.1, the application joins the 'candidate list' for final application selection.

4. If the entry is for a DDF the terminal selects the DDF indicated using the DDF name. Using the Directory SFI from the FCI of the selected DDF, the directory is read and processed according to steps 2 through 5, after which the terminal resumes processing the previously interrupted directory at the point of interruption.

5. When the terminal completes the list in the first directory, all ADFs that can be found by this procedure have been determined. The search and the candidate list are complete. The terminal continues processing as described in section III-4.

6. If no directory entries exist, the terminal may know other ways to find proprietary applications within the card (but this is outside the scope of this

specifications) or may apply the selection technique using a list of AIDs as described in section 3.2.

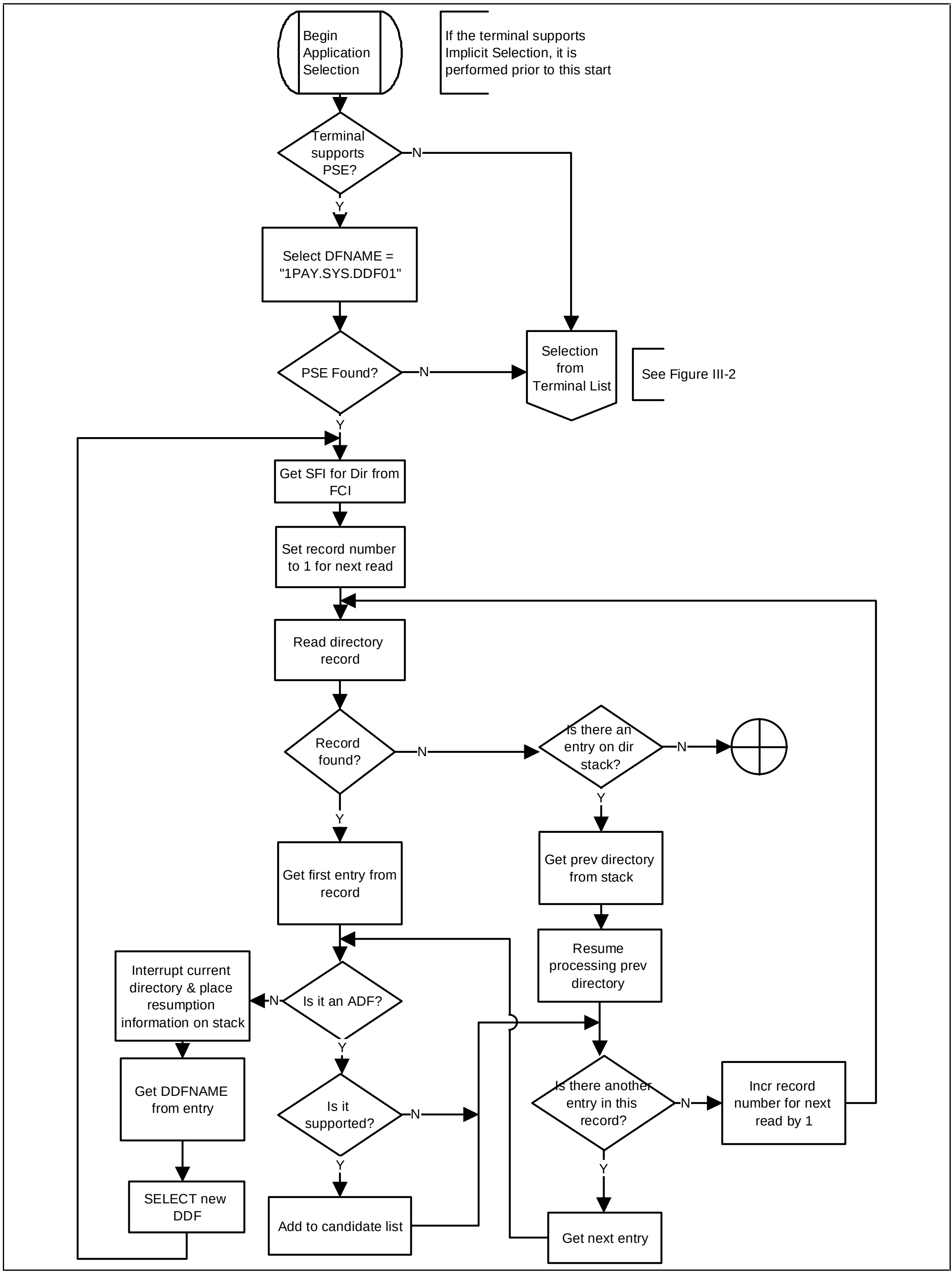

Figure III-1 - Terminal Logic Using Directories

3.3 Using a List of AIDs

If the number of applications supported by a terminal is small or the card contains no PSE, the terminal uses a list of applications that it supports to build the candidate list. This procedure is shown in Figure III-2. The terminal performs the following steps for each AID in its list:

1. The terminal issues a SELECT command using the AID in the terminal list as the file name.

2. If the SELECT command fails because the card is blocked or the command is not supported by the ICC (SW1 SW2 = '6A81'), the terminal terminates the card session.

   If the SELECT command is successful but the application is blocked (SW1 SW2 = '6283'), the terminal proceeds to step 3 without adding the DFNAME to the candidate list.

   If the SELECT command is successful (SW1 SW2 = '9000'), the terminal adds the DFNAME from the FCI of the selected file to the candidate list and proceeds to step 3.

   If the response from the ICC is anything else, the terminal proceeds back to step 1 using the next AID in the list. If there are no more AIDs in the list, the candidate list is complete, and the terminal proceeds as specified in section III-4.

3. The terminal checks for the possibility of multiple occurrences of the application in the card by comparing the AID in the terminal list to the DFNAME in the FCI of the selected ADF. Two possibilities exist:

   a. The terminal AID is the same as the DFNAME and they have the same length. In this case, there can be no other occurrences of the application in the ICC (see restriction in section 2.1). The terminal proceeds to step 1 using the next AID in its list.

   or

   b. The DFNAME is longer than the AID in the terminal, but they are identical up to and including the last character in the terminal AID. In this case, there may be multiple applications in the ICC matching this terminal AID. The terminal repeats the SELECT command using the same command data, but changes P2 in the command to '02' ('select next').

   If the ICC does not support the 'next' option on the SELECT command, the ICC shall return SW1 SW2 = '6A81' ('Wrong parameters P1-P2 - Function not supported'). In this case, the card cannot have multiple occurrences of the application, and the terminal proceeds to step 1 using the next AID in its list. If the ICC returns anything else, the terminal proceeds to step 2.

Once all applications supported by the terminal have been processed as specified above, the candidate list is complete. The terminal proceeds as specified in section III-4.

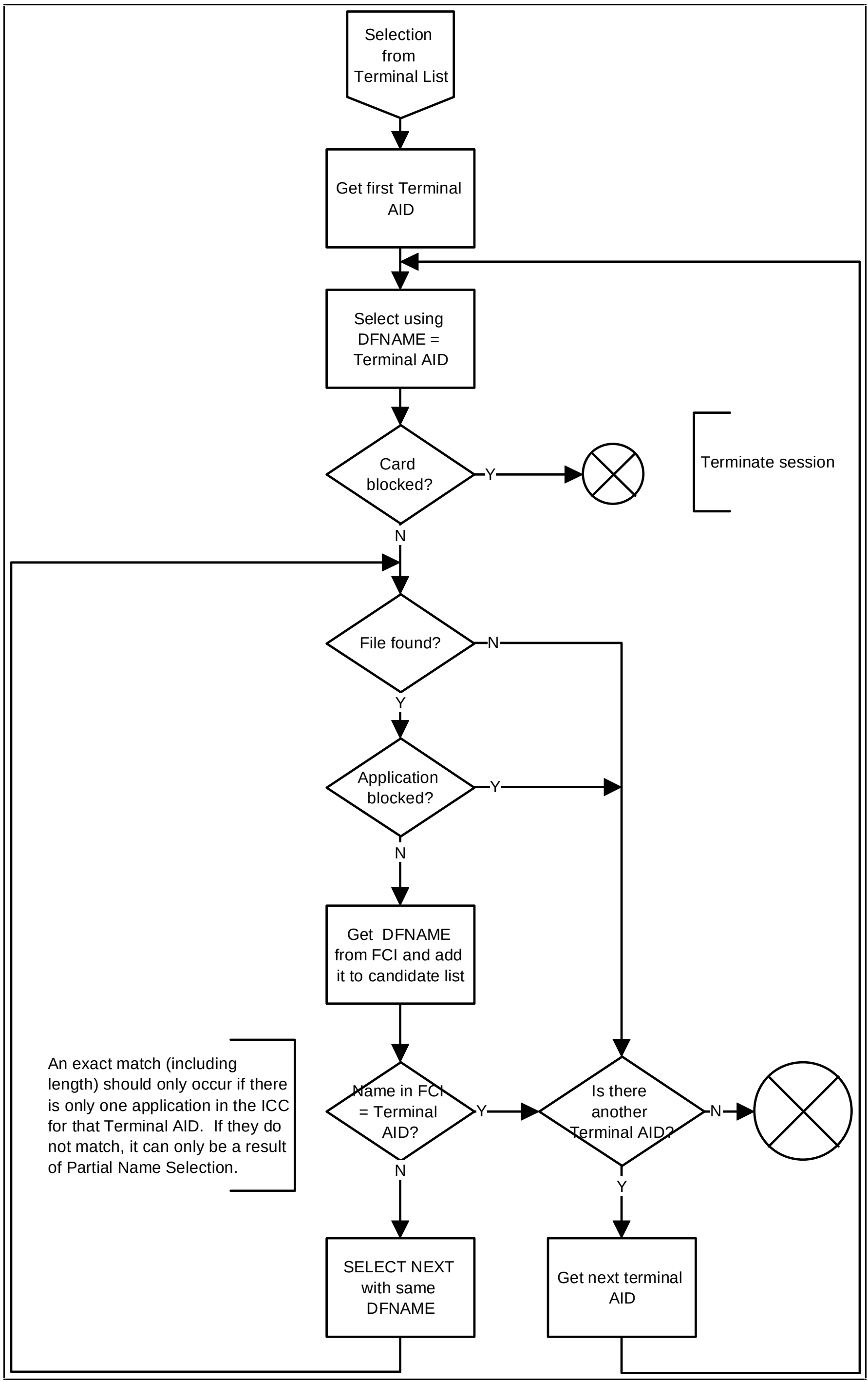

Figure III-2 - Using the List of Applications in the Terminal

3.4 Final Selection

Once the terminal determines the list of mutually supported applications, it proceeds as follows:

1. If there are no mutually supported applications, the transaction is terminated.

2. If there is only one mutually supported application, the terminal checks b8 of the Application Priority Indicator for that application. If b8 = '0', the terminal selects the application. If b8 = '1' and the terminal provides for confirmation by the cardholder, the terminal requests confirmation and selects the application if the cardholder approves. If the terminal does not provide for confirmation by the cardholder, or if the terminal requests confirmation and the cardholder does not approve, the terminal terminates the session.

3. If multiple applications are supported, the terminal may offer a selection to the cardholder as described in step 4, or make the selection itself as described in step 5. Step 4 is the preferred method.

4. If a list is presented to the cardholder, it shall be in priority sequence, with the highest priority application listed first. If there is no priority sequence specified in the card, the list should be in the order in which the applications were encountered in the card, unless the terminal has its own preferred order. The same applies where duplicate priorities are assigned to multiple applications or individual entries are missing the Application Priority Indicator; that is, in this case, the terminal may use its own preferred order or display the duplicate priority or nonprioritised applications in the order encountered in the card.

5. The terminal may select the application without cardholder assistance. In this case, the terminal shall select the highest priority application from the list of mutually supported applications, except that if the terminal does not provide for confirmation of the selected application, applications prohibiting such selection (b8 = '1' in the Application Priority Indicator) shall be excluded from possible selection.

Once the application to be run is determined by the terminal or by the cardholder, the application shall be selected. A SELECT command coded according to Part II shall be issued by the terminal for the application using the ADF Name field (if the directories were read) or the DFNAME field from the FCI (if the list method was used) found during the building of the candidate list. If the command returns other than '9000' in SW1 SW2, the application shall be removed from the candidate list, and processing shall resume at step 1. The terminal shall inform the cardholder of the action taken, if appropriate.

THIS PAGE LEFT INTENTIONALLY BLANK

Part IV

Security Aspects

1. Static Data Authentication

Static data authentication is performed by the terminal using a digital signature based on public key techniques to confirm the legitimacy of critical ICC-resident static data identified by the AFL. This detects unauthorised alteration of data after personalisation.

Static data authentication requires the existence of a certification authority, which is a highly secure cryptographic facility that 'signs' the issuer's public keys. Every terminal conforming to this specification shall contain the appropriate certification authority's public key(s) for every application recognised by the terminal. This specification permits multiple AIDs to share the same 'set' of certification authority public keys. The relationship between the data and the cryptographic keys is shown in Figure IV-1.

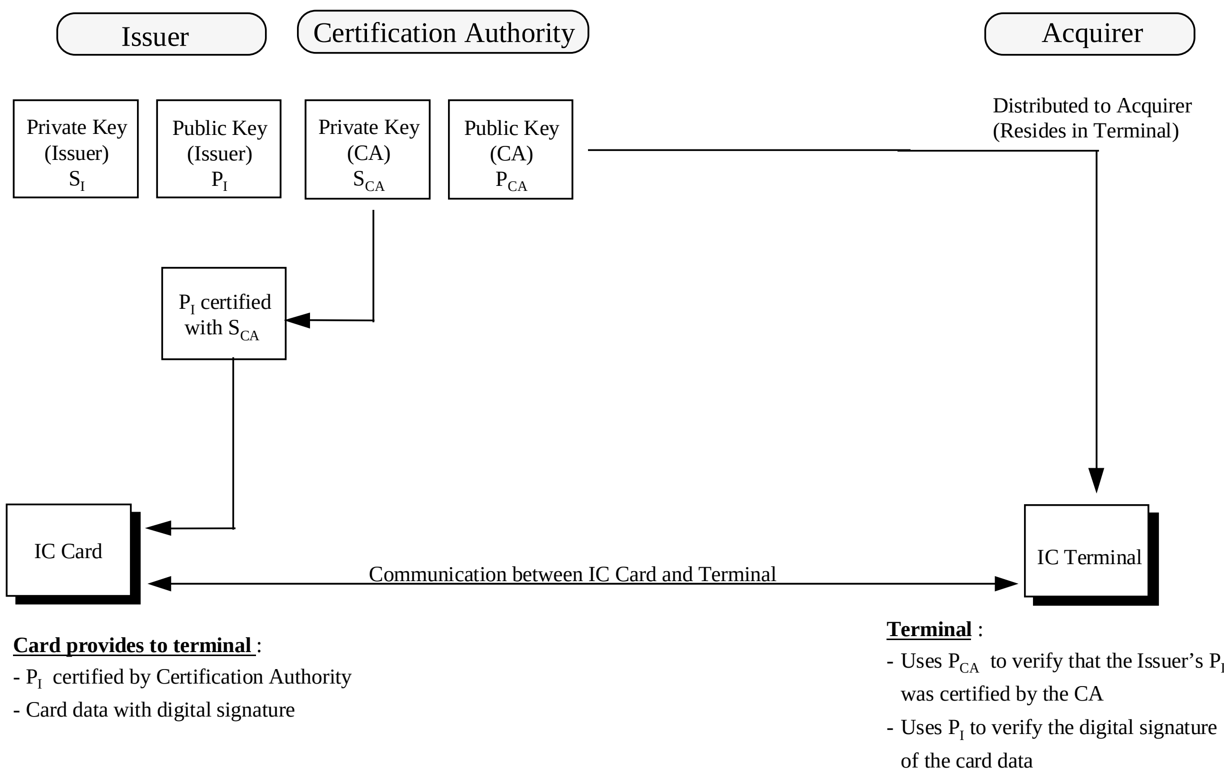

Figure IV-1 - Diagram of Static Data Authentication

ICCs that support static data authentication shall contain the following data elements:

- Certification Authority Public Key Index: This one-byte data element contains a binary number that indicates which of the application's certification authority public keys and its associated algorithm that reside in the terminal is to be used with this ICC.

- Issuer Public Key Certificate: This variable-length data element is provided by the appropriate certification authority to the card issuer. When the terminal verifies this data element, it authenticates the Issuer Public Key plus additional data as described in section IV-1.3.

- Signed Static Application Data: A variable-length data element generated by the issuer using the private key that corresponds to the public key authenticated in the Issuer Public Key Certificate. It is a digital signature covering critical ICC-resident static data elements, as described in section IV-1.4.

- Issuer Public Key Remainder: A variable length data element. Its presence in the ICC is optional. See section IV-1.1 for further explanation.

- Issuer Public Key Exponent: A variable length data element provided by the issuer. See section IV-1.1 for further explanation.

To support static data authentication, each terminal shall be able to store multiple certification authority public keys and shall associate with each such key the key-related information to be used with the key (so that terminals can in the future support multiple algorithms and allow an evolutionary transition from one to another). The terminal shall be able to locate any such key (and the key-related information) given the RID and Certification Authority Public Key Index as provided by the ICC.

Static data authentication shall use a reversible algorithm as specified in Annex E2.1 and Annex F2. Section IV-1.1 contains an overview of the keys and certificates involved in the static data authentication process, and sections IV-1.2 to IV-1.4 specify the three main steps in the process, namely

- Retrieval of the Certification Authority Public Key by the terminal.

- Retrieval of the Issuer Public Key by the terminal.

- Verification of the Signed Static Application Data by the terminal.

1.1 Keys and Certificates

To support static data authentication, an ICC shall contain the Signed Static Application Data, which is signed with the Issuer Private Key. The Issuer Public Key shall be stored on the ICC with a public key certificate.

The bit length of all moduli shall be a multiple of 8, the leftmost bit of its leftmost byte being 1. All lengths are given in bytes.

The signature scheme specified in Annex E2.1 is applied to the data specified in Table IV-1 using the Certification Authority Private Key S_{CA} in order to obtain the Issuer Public Key Certificate.

The public key pair of the certification authority has a public key modulus of N_{CA} bytes, where $N_{CA} \leq 248$. The Certification Authority Public Key Exponent shall be equal to 2, 3, or $2^{16} + 1$.

The signature scheme specified in Annex E2.1 is applied to the data specified in Table IV-2 using the Issuer Private Key S_I in order to obtain the Signed Static Application Data.

The public key pair of the issuer has an Issuer Public Key Modulus of N_I bytes, where $N_I < 248$ and $N_I < N_{CA}$. If $N_I > (N_{CA} - 36)$, the Issuer Public Key Modulus is split into two parts, namely one part consisting of the $N_{CA} - 36$ most significant bytes of the modulus (the Leftmost Digits of the Issuer Public Key) and a second part consisting of the remaining $N_I - (N_{CA} - 36)$ least significant bytes of the modulus (the Issuer Public Key Remainder). The Issuer Public Key Exponent shall be equal to 2, 3, or $2^{16} + 1$.

All the information necessary for static data authentication is specified in Table IV-3 and stored in the ICC. With the exception of the RID, which can be obtained from the AID, this information may be retrieved with the READ RECORD command. If any of this data is missing, static data authentication has failed.

Field Name	Length	Description	Format
Certificate Format	1	Hex. value ‘02’	b
Issuer Identification Number	4	Leftmost 3-8 digits from the Primary Account Number (PAN) (padded to the right with hex. ‘F’s)	cn 8
Certificate Expiration Date	2	MMYY after which this certificate is invalid	n 4
Certificate Serial Number	3	Binary number unique to this certificate assigned by the certification authority	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme[11]	b
Issuer Public Key Algorithm Indicator	1	Identifies the digital signature algorithm to be used with the Issuer Public Key[11]	b
Issuer Public Key Length	1	Identifies the length of the Issuer Public Key Modulus in bytes	b
Issuer Public Key Exponent Length	1	Identifies the length of the Issuer Public Key Exponent in bytes	b
Issuer Public Key or Leftmost Digits of the Issuer Public Key	$N_{CA} - 36$	If $N_I \le N_{CA} - 36$, this field consists of the full Issuer Public Key padded to the right with $N_{CA} - 36 - N_I$ bytes of value ‘BB’ If $N_I > N_{CA} - 36$, this field consists of the $N_{CA} - 36$ most significant bytes of the Issuer Public Key[12]	b
Issuer Public Key Remainder	0 or $N_I - N_{CA} + 36$	This field is only present if $N_I > N_{CA} - 36$ and consists of the $N_I - N_{CA} + 36$ least significant bytes of the Issuer Public Key.	b
Issuer Public Key Exponent	1 or 3	Issuer Public Key Exponent equal to 2, 3, or $2^{16} + 1$	b

Table IV-1 - Issuer Public Key Data to be Signed by the Certification Authority (i.e., input to the hash algorithm)

[11] See Annex F for specific values assigned to approved algorithms.

[12] As can be seen in Annex E2.1, $N_{CA} - 22$ bytes of the data signed are retrieved from the signature. Since the length of the first through the eighth data elements in Table IV-1 is 14 bytes, there are $N_{CA} - 22 - 14 = N_{CA} - 36$ bytes remaining in the signature to store the Issuer Public Key Modulus.

Field Name	Length	Description	Format
Signed Data Format	1	Hex. value '03'	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme[11]	b
Data Authentication Code	2	Issuer-assigned code	b
Pad Pattern	$N_I - 26$	Pad pattern consisting of $N_I - 26$ bytes of value 'BB'[13]	b
Static Data to be Authenticated	var.	Static data to be authenticated as specified in the *ICC Application Specification for Payment Systems*	-

Table IV-2 - Static Application Data to be Signed by the Issuer (i.e., input to the hash algorithm)

Tag	Length	Value	Format
-	5	Registered Application Provider Identifier (RID)	b
'8F'	1	Certification Authority Public Key Index	b
'90'	N_{CA}	Issuer Public Key Certificate	b
'92'	$N_I - N_{CA} + 36$	Issuer Public Key Remainder, if present	b
'9F32'	1 or 3	Issuer Public Key Exponent	b
'93'	N_I	Signed Static Application Data	b
-	var.	Static data to be authenticated as specified in the *ICC Application Specification for Payment Systems*	-

Table IV-3 - Data Objects Required for Static Data Authentication

1.2 Retrieval of the Certification Authority Public Key

The terminal reads the Certification Authority Public Key Index. Using this index and the RID, the terminal shall identify and retrieve the terminal-stored Certification Authority Public Key Modulus and Exponent and the associated key-related information, and the corresponding algorithm to be used. If the terminal does not have the key stored associated with this index and RID, static data authentication has failed.

[13] As can be seen in Annex E2.1, $N_I - 22$ bytes of the data signed are retrieved from the signature. Since the first through the third data elements in Table IV-2 total 4 bytes, there are $N_I - 22 - 4 = N_I - 26$ bytes left for the data to be stored in the signature.

1.3 Retrieval of the Issuer Public Key

1. If the Issuer Public Key Certificate has a length different from the length of the Certification Authority Public Key Modulus obtained in the previous section, static data authentication has failed.

2. In order to obtain the recovered data specified in Table IV-4, apply the recovery function specified in Annex E2.1 to the Issuer Public Key Certificate using the Certification Authority Public Key in conjunction the corresponding algorithm. If the Recovered Data Trailer is not equal to 'BC', static data authentication has failed.

Field Name	Length	Description	Format
Recovered Data Header	1	Hex. value '6A'	b
Certificate Format	1	Hex. value '02'	b
Issuer Identification Number	4	Leftmost 3-8 digits from the PAN (padded to the right with hex. 'F's)	cn 8
Certificate Expiration Date	2	MMYY after which this certificate is invalid	n 4
Certificate Serial Number	3	Binary number unique to this certificate assigned by the certification authority	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme[11]	b
Issuer Public Key Algorithm Indicator	1	Identifies the digital signature algorithm to be used with the Issuer Public Key[11]	b
Issuer Public Key Length	1	Identifies the length of the Issuer Public Key Modulus in bytes	b
Issuer Public Key Exponent Length	1	Identifies the length of the Issuer Public Key Exponent in bytes	b
Issuer Public Key or Leftmost Digits of the Issuer Public Key	$N_{CA} - 36$	If $N_I \leq N_{CA} - 36$, this field consists of the full Issuer Public Key padded to the right with $N_{CA} - 36 - N_I$ bytes of value 'BB' If $N_I > N_{CA} - 36$, this field consists of the $N_{CA} - 36$ most significant bytes of the Issuer Public Key[12]	b
Hash Result	20	Hash of the Issuer Public Key and its related information	b
Recovered Data Trailer	1	Hex. value 'BC'	b

Table IV-4 - Format of the Data Recovered from the Issuer Public Key Certificate

3. Check the Recovered Data Header. If it is not '6A', static data authentication has failed.

4. Check the Certificate Format. If it is not '02', static data authentication has failed.

5. Concatenate from left to right the second to the tenth data elements in Table IV-4 (that is, Certificate Format through Issuer Public Key or Leftmost Digits of the Issuer Public Key), followed by the Issuer Public Key Remainder (if present) and finally the Issuer Public Key Exponent.

6. Apply the indicated hash algorithm (derived from the Hash Algorithm Indicator) to the result of the concatenation of the previous step to produce the hash result.

7. Compare the calculated hash result from the previous step with the recovered Hash Result. If they are not the same, static data authentication has failed.

8. Verify that the Issuer Identification Number matches the leftmost 3-8 PAN digits (allowing for the possible padding of the Issuer Identification Number with hexadecimal 'F's). If not, static data authentication has failed.

9. Verify that the last day of the month specified in the Certificate Expiration Date is equal to or later than today's date. If the Certificate Expiration Date is earlier than today's date, the certificate has expired, in which case static data authentication has failed.

10. Verify that the concatenation of RID, Certification Authority Public Key Index, and Certificate Serial Number is valid. If not, static data authentication has failed[14].

11. If the Issuer Public Key Algorithm Indicator is not recognised, static data authentication has failed.

12. If all the checks above are correct, concatenate the Leftmost Digits of the Issuer Public Key and the Issuer Public Key Remainder (if present) to obtain the Issuer Public Key Modulus, and continue with the next steps for the verification of the Signed Static Application Data.

1.4 Verification of the Signed Static Application Data

1. If the Signed Static Application Data has a length different from the length of the Issuer Public Key Modulus, static data authentication has failed.

2. In order to obtain the Recovered Data specified in Table IV-5, apply the recovery function specified in Annex E2.1 on the Signed Static Application Data using the

[14] This step is optional and is to allow the revocation of the Issuer Public Key Certificate against a list that may be kept by the terminal.

Issuer Public Key in conjunction the corresponding algorithm. If the Recovered Data Trailer is not equal to ‘BC’, static data authentication has failed.

Field Name	Length	Description	Format
Recovered Data Header	1	Hex. value ‘6A’	b
Signed Data Format	1	Hex. value ‘03’	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme[11]	b
Data Authentication Code	2	Issuer-assigned code	b
Pad Pattern	$N_I - 26$	Pad pattern consisting of $N_I - 26$ bytes of value ‘BB’[13]	b
Hash Result	20	Hash of the Static Application Data to be authenticated	b
Recovered Data Trailer	1	Hex. value ‘BC’	b

Table IV-5 - Format of the Data Recovered from the Signed Static Application Data

3. Check the Recovered Data Header. If it is not ‘6A’, static data authentication has failed.

4. Check the Signed Data Format. If it is not ‘03’, static data authentication has failed.

5. Concatenate from left to right the second to the fifth data elements in Table IV-5 (that is, Signed Data Format through Pad Pattern), followed by the static data to be authenticated as specified in the *ICC Application Specification for Payment Systems*.

6. Apply the indicated hash algorithm (derived from the Hash Algorithm Indicator) to the result of the concatenation of the previous step to produce the hash result.

7. Compare the calculated hash result from the previous step with the recovered Hash Result. If they are not the same, static data authentication has failed.

If all of the above steps were executed successfully, static data authentication was successful.

2. Dynamic Data Authentication

Dynamic data authentication is performed by the terminal using a digital signature based on public key techniques to authenticate the ICC, and confirm the legitimacy of critical ICC-resident data identified by the ICC dynamic data and data received from the terminal identified by the Dynamic Data Authentication Data Object List (DDOL). This precludes the counterfeiting of any such card.

Dynamic data authentication requires the existence of a certification authority, a highly secure cryptographic facility that ‘signs’ the Issuer’s Public Keys. Every terminal conforming to this specification shall contain the appropriate certification authority's public key(s) for every application recognised by the terminal. This specification permits multiple AIDs to share the same ‘set’ of certification authority public keys. The relationship between the data and the cryptographic keys is shown in Figure IV-2.

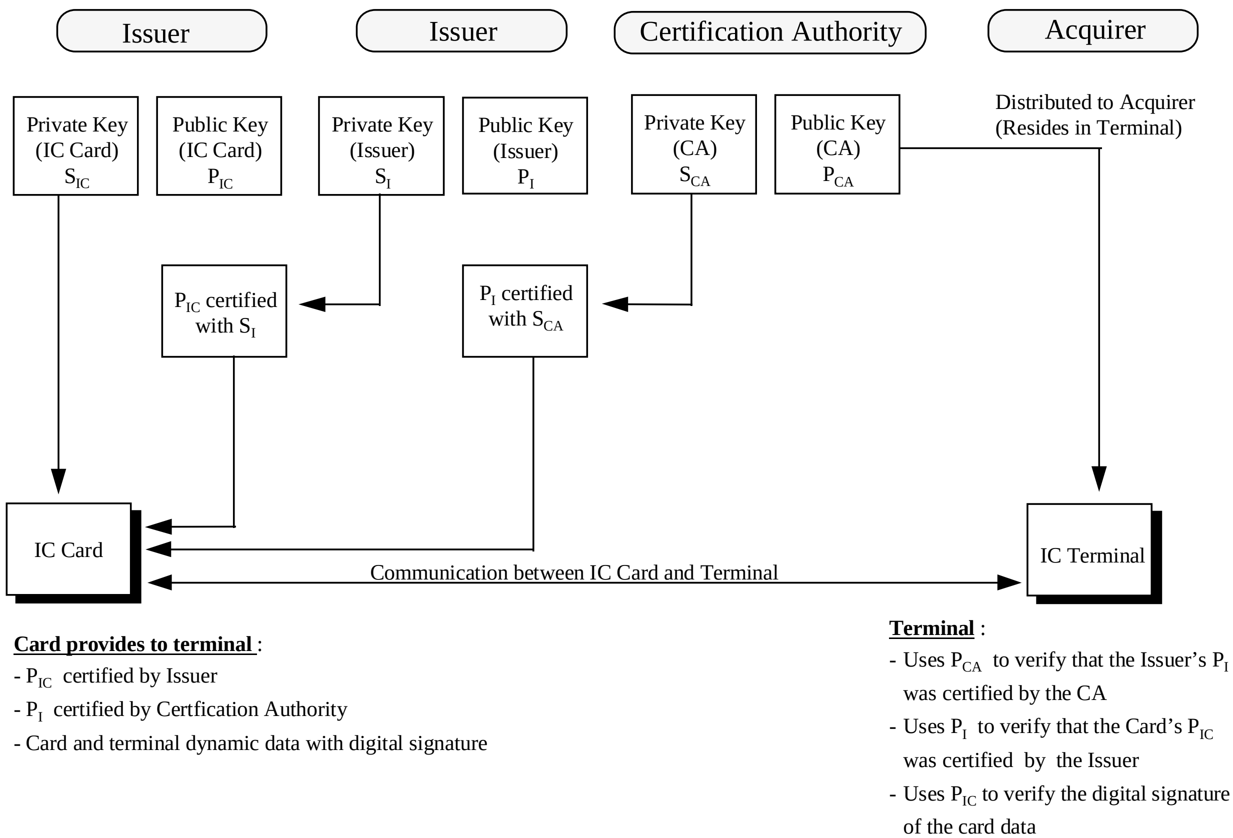

Figure IV-2 - Diagram of Dynamic Data Authentication

ICCs that support dynamic data authentication shall contain the following data elements:

- Certification Authority Public Key Index: This one-byte data element contains a binary number that indicates which of the application’s certification authority public keys and its associated algorithm that reside in the terminal is to be used with this ICC.

- Issuer Public Key Certificate: This variable-length data element is provided by the appropriate certification authority to the card issuer. When the terminal verifies this data element, it authenticates the Issuer Public Key plus additional data as described in section IV-2.3.

- ICC Public Key Certificate: This variable-length data element is provided by the issuer to the ICC. When the terminal verifies this data element, it authenticates the ICC Public Key plus additional data as described in section IV-2.4.

- Issuer Public Key Remainder: A variable-length data element. See section IV-2.1 for further explanation.

- Issuer Public Key Exponent: A variable-length data element provided by the issuer. See section IV-2.1 for further explanation.

- ICC Public Key Remainder: A variable-length data element. See section IV-2.1 for further explanation.

- ICC Public Key Exponent: A variable-length data element provided by the issuer. See section IV-2.1 for further explanation.

- ICC Private Key: An ICC internal variable-length data element used to generate the Signed Dynamic Application Data as described in section IV-2.5.

ICCs that support dynamic data authentication shall generate the following data element:

- Signed Dynamic Application Data: A variable-length data element generated by the ICC using the private key that corresponds to the public key authenticated in the ICC Public Key Certificate. It is a digital signature covering critical ICC-resident and terminal data elements, as described in section IV-2.5.

To support dynamic data authentication, each terminal shall be able to store multiple certification authority public keys and shall associate with each such key the key-related information to be used with the key (so that terminals can in the future support multiple algorithms and allow an evolutionary transition from one to another). The terminal shall be able to locate any such key (and key-related information) given the RID and Certification Authority Public Key Index as provided by the ICC.

Dynamic data authentication shall use a reversible algorithm as specified in Annex E2.1 and Annex F2. Section IV-2.1 contains an overview of the keys and certificates involved in the dynamic data authentication process, and sections IV-2.3 to IV-2.6 specify the five main steps in the process, namely

- Retrieval of the Certification Authority Public Key by the terminal.
- Retrieval of the Issuer Public Key by the terminal.
- Retrieval of the IC Public Key by the terminal.
- Dynamic signature generation by the ICC.
- Dynamic signature verification by the terminal.

2.1 Keys and Certificates

To support dynamic data authentication, an ICC shall own its own unique public key pair consisting of a private signature key and the corresponding public verification key. The ICC Public Key shall be stored on the ICC in a public key certificate.

More precisely, a three-layer public key certification scheme is used. Each ICC Public Key is certified by its issuer, and the certification authority certifies the Issuer Public Key. This implies that, for the verification of an ICC signature, the terminal first needs to verify two certificates in order to retrieve and authenticate the ICC Public Key, which is then employed to verify the ICC's dynamic signature.

The bit length of all moduli shall be a multiple of 8, the leftmost bit of its leftmost byte being 1. All lengths are given in bytes.

The signature scheme specified in Annex E2.1 is applied on the data in Table IV-6 and on the data in Table IV-7 using the Certification Authority Private Key S_{CA} and the Issuer Private Key S_I in order to obtain the Issuer Public Key Certificate and ICC Public Key Certificate, respectively.

The public key pair of the certification authority has a Certification Authority Public Key Modulus of N_{CA} bytes, where $N_{CA} \leq 248$. The Certification Authority Public Key Exponent shall be equal to 2, 3, or $2^{16} + 1$.

The public key pair of the issuer has a Public Key Modulus of N_I bytes, where $N_I <$ 248 and $N_I < N_{CA}$. If $N_I > (N_{CA} - 36)$, the Issuer Public Key Modulus is divided into two parts, one part consisting of the $N_{CA} - 36$ most significant bytes of the modulus (the Leftmost Digits of the Issuer Public Key) and a second part consisting of the remaining $N_I - (N_{CA} - 36)$ least significant bytes of the modulus (the Issuer Public Key Remainder). The Issuer Public Key Exponent shall be equal to 2, 3, or $2^{16} + 1$.

The public key pair of the ICC has an ICC Public Key Modulus of N_{IC} bytes, where $N_{IC} \leq 128$ and $N_{IC} < N_I$. If $N_{IC} > (N_I - 42)$, the ICC Public Key Modulus is divided into two parts, one part consisting of the $N_I - 42$ most significant bytes of the modulus (the Leftmost Digits of the ICC Public Key) and a second part consisting of the remaining $N_{IC} - (N_I - 42)$ least significant bytes of the modulus (the ICC Public Key Remainder). The ICC Public Key Exponent shall be equal to 2, 3, or $2^{16} + 1$.

To execute dynamic data authentication, the terminal shall first retrieve and authenticate the ICC Public Key (this process is called ICC Public Key authentication). All the information necessary for ICC Public Key authentication is specified in Table IV-8 and stored in the ICC. With the exception of the RID, which can be obtained from the AID, this information may be retrieved with the READ RECORD command. If any of this data is missing, dynamic data authentication has failed.

Field Name	Length	Description	Format
Certificate Format	1	Hex. value '02'	b
Issuer Identification Number	4	Leftmost 3-8 digits from the PAN (padded to the right with hex. 'F's)	cn 8
Certificate Expiration Date	2	MMYY after which this certificate is invalid	n 4
Certificate Serial Number	3	Binary number unique to this certificate assigned by the certification authority	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme[11]	b
Issuer Public Key Algorithm Indicator	1	Identifies the digital signature algorithm to be used with the Issuer Public Key [11]	b
Issuer Public Key Length	1	Identifies the length of the Issuer Public Key Modulus in bytes	b
Issuer Public Key Exponent Length	1	Identifies the length of the Issuer Public Key Exponent in bytes	b
Issuer Public Key or Leftmost Digits of the Issuer Public Key	$N_{CA} - 36$	If $N_I \leq N_{CA} - 36$, this field consists of the full Issuer Public Key padded to the right with $N_{CA} - 36 - N_I$ bytes of value 'BB' If $N_I > N_{CA} - 36$, this field consists of the $N_{CA} - 36$ most significant bytes of the Issuer Public Key [12]	b
Issuer Public Key Remainder	0 or $N_I - N_{CA} + 36$	This field is only present if $N_I > N_{CA} - 36$ and consists of the $N_I - N_{CA} + 36$ least significant bytes of the Issuer Public Key	b
Issuer Public Key Exponent	1 or 3	Issuer Public Key Exponent equal to 2, 3 or $2^{16} + 1$	b

Table IV-6 - Issuer Public Key Data to be Signed by the Certification Authority (i.e., input to the hash algorithm)

Field Name	Length	Description	Format
Certificate Format	1	Hex. value ‘04’	b
Application PAN	10	PAN (padded to the right with hex. ‘F’s)	cn 20
Certificate Expiration Date	2	MMYY after which this certificate is invalid	n 4
Certificate Serial Number	3	Binary number unique to this certificate assigned by the issuer	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme[11]	b
ICC Public Key Algorithm Indicator	1	Identifies the digital signature algorithm to be used with the ICC Public Key[11]	b
ICC Public Key Length	1	Identifies the length of the ICC Public Key Modulus in bytes	b
ICC Public Key Exponent Length	1	Identifies the length of the ICC Public Key Exponent in bytes	b
ICC Public Key or Leftmost Digits of the ICC Public Key	$N_I - 42$	If $N_{IC} \leq N_I - 42$, this field consists of the full ICC Public Key padded to the right with $N_I - 42 - N_{IC}$ bytes of value ‘BB’ If $N_{IC} > N_I - 42$, this field consists of the $N_I - 42$ most significant bytes of the ICC Public Key [15]	b
ICC Public Key Remainder	0 or $N_{IC} - N_I + 42$	This field is only present if $N_{IC} > N_I - 42$ and consists of the $N_I - N_{CA} + 42$ least significant bytes of the ICC Public Key	b
ICC Public Key Exponent	1 or 3	ICC Public Key Exponent equal to 2, 3 or $2^{16} + 1$	b
Static Data to be Authenticated	var.	Static data to be authenticated as specified in the *ICC Application Specification for Payment Systems*	b

Table IV-7 - ICC Public Key Data to be Signed by the Issuer (i.e., input to the hash algorithm)

[15] As can be seen in Annex E2.1, $N_I - 22$ bytes of the data signed are retrieved from the signature. Since the first through the eighth data elements in Table IV-7 total 20 bytes, there are $N_I - 22 - 20 = N_I - 42$ bytes left for the data to be stored in the signature.

Tag	Length	Value	Format
-	5	Registered Application Provider Identifier (RID)	b
'8F'	1	Certification Authority Public Key Index	b
'90'	N_{CA}	Issuer Public Key Certificate	b
'92'	$N_I - N_{CA} + 36$	Issuer Public Key Remainder, if present	b
'9F32'	1 or 3	Issuer Public Key Exponent	b
'9F46'	N_I	ICC Public Key Certificate	b
'9F48'	$N_{IC} - N_I + 42$	ICC Public Key Remainder, if present	b
'9F47'	1 or 3	ICC Public Key Exponent	b
-	var.	Static data to be authenticated as specified in the *ICC Application Specification for Payment Systems*	-

Table IV-8 - Data Objects Required for Public Key Authentication for Dynamic Authentication

2.2 Retrieval of the Certification Authority Public Key

The terminal reads the Certification Authority Public Key Index. Using this index and the RID, the terminal can identify and retrieve the terminal-stored Certification Authority Public Key Modulus and Exponent and the associated key-related information, and the corresponding algorithm to be used. If the terminal does not have the key stored associated with this index and RID, dynamic data authentication has failed.

2.3 Retrieval of the Issuer Public Key

1. If the Issuer Public Key Certificate has a length different from the length of the Certification Authority Public Key Modulus obtained in the previous section, dynamic data authentication has failed.

2. In order to obtain the recovered data specified in Table IV-9, apply the recovery function specified in Annex E2.1 on the Issuer Public Key Certificate using the Certification Authority Public Key in conjunction the corresponding algorithm. If the Recovered Data Trailer is not equal to 'BC', dynamic data authentication has failed.

Field Name	Length	Description	Format
Recovered Data Header	1	Hex. value ‘6A’	b
Certificate Format	1	Hex. value ‘02’	b
Issuer Identification Number	4	Leftmost 3-8 digits from the PAN (padded to the right with hex. ‘F’s)	cn 8
Certificate Expiration Date	2	MMYY after which this certificate is invalid	n 4
Certificate Serial Number	3	Binary number unique to this certificate assigned by the certification authority	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme[11]	b
Issuer Public Key Algorithm Indicator	1	Identifies the digital signature algorithm to be used with the Issuer Public Key[11]	b
Issuer Public Key Length	1	Identifies the length of the Issuer Public Key Modulus in bytes	b
Issuer Public Key Exponent Length	1	Identifies the length of the Issuer Public Key Exponent in bytes	b
Issuer Public Key or Leftmost Digits of the Issuer Public Key	$N_{CA} - 36$	If $N_I \leq N_{CA} - 36$, this field consists of the full Issuer Public Key padded to the right with $N_{CA} - 36 - N_I$ bytes of value ‘BB’ If $N_I > N_{CA} - 36$, this field consists of the $N_{CA} - 36$ most significant bytes of the Issuer Public Key[12]	b
Hash Result	20	Hash of the Issuer Public Key and its related information	b
Recovered Data Trailer	1	Hex. value ‘BC’	b

Table IV-9 - Format of the Data Recovered from the Issuer Public Key Certificate

3. Check the Recovered Data Header. If it is not ‘6A’, dynamic data authentication has failed.

4. Check the Certificate Format. If it is not ‘02’, dynamic data authentication has failed.

5. Concatenate from left to right the second to the tenth data elements in Table IV-9 (that is, Certificate Format through Issuer Public Key or Leftmost Digits of the Issuer Public Key), followed by the Issuer Public Key Remainder (if present) and finally the Issuer Public Key Exponent.

6. Apply the indicated hash algorithm (derived from the Hash Algorithm Indicator) to the result of the concatenation of the previous step to produce the hash result.

7. Compare the calculated hash result from the previous step with the recovered Hash Result. If they are not the same, dynamic data authentication has failed.

8. Verify that the Issuer Identification Number matches the leftmost 3-8 PAN digits (allowing for the possible padding of the Issuer Identification Number with hexadecimal ‘F’s). If not, dynamic data authentication has failed.

9. Verify that the last day of the month specified in the Certificate Expiration Date is equal to or later than today's date. If the Certificate Expiration Date is earlier than today's date, the certificate has expired, in which case dynamic data authentication has failed.

10. Verify that the concatenation of RID, Certification Public Key Index, and Certificate Serial Number is valid. If not, dynamic data authentication has failed[14].

11. If the Issuer Public Key Algorithm Indicator is not recognised, dynamic data authentication has failed.

12. If all the checks above are correct, concatenate the Leftmost Digits of the Issuer Public Key and the Issuer Public Key Remainder (if present) to obtain the Issuer Public Key Modulus, and continue with the next steps for the retrieval of the ICC Public Key.

2.4 Retrieval of the ICC Public Key

1. If the ICC Public Key Certificate has a length different from the length of the Issuer Public Key Modulus obtained in the previous section, dynamic data authentication has failed.

2. In order to obtain the recovered data specified in Table IV-10, apply the recovery function specified in Annex E2.1 on the ICC Public Key Certificate using the Issuer Public Key in conjunction the corresponding algorithm. If the Recovered Data Trailer is not equal to ‘BC’, dynamic data authentication has failed.

Field Name	Length	Description	Format
Recovered Data Header	1	Hex. value '6A'	b
Certificate Format	1	Hex. value '04'	b
Application PAN	10	PAN (padded to the right with hex. 'F's	cn 20
Certificate Expiration Date	2	MMYY after which this certificate is invalid	n 4
Certificate Serial Number	3	Binary number unique to this certificate assigned by the issuer	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme[11]	b
ICC Public Key Algorithm Indicator	1	Identifies the digital signature algorithm to be used with the ICC Public Key[11]	b
ICC Public Key Length	1	Identifies the length of the ICC Public Key Modulus in bytes	b
ICC Public Key Exponent Length	1	Identifies the length of the ICC Public Key Exponent in bytes	b
ICC Public Key or Leftmost Digits of the ICC Public Key	$N_I - 42$	If $N_{IC} \leq N_I - 42$, this field consists of the full ICC Public Key padded to the right with $N_I - 42 - N_{IC}$ bytes of value 'BB'[15] If $N_{IC} > N_I - 42$, this field consists of the $N_I - 42$ most significant bytes of the ICC Public Key	b
Hash Result	20	Hash of the ICC Public Key and its related information	b
Recovered Data Trailer	1	Hex. value 'BC'	b

Table IV-10 - Format of the Data Recovered from the ICC Public Key Certificate

3. Check the Recovered Data Header. If it is not '6A', dynamic data authentication has failed.

4. Check the Certificate Format. If it is not '04', dynamic data authentication has failed.

5. Concatenate from left to right the second to the tenth data elements in Table IV-10 (that is, Certificate Format through ICC Public Key or Leftmost Digits of the ICC Public Key), followed by the ICC Public Key Remainder (if present), the ICC Public Key Exponent and finally the static data to be authenticated specified in the *ICC Application Specification for Payment Systems.*

6. Apply the indicated hash algorithm (derived from the Hash Algorithm Indicator) to the result of the concatenation of the previous step to produce the hash result.

7. Compare the calculated hash result from the previous step with the recovered Hash Result. If they are not the same, dynamic data authentication has failed.

8. Check if the recovered PAN is equal to the Application PAN, read from the ICC. If not, dynamic data authentication has failed.

9. Verify that the last day of the month specified in the Certificate Expiration Date is equal to or later than today's date. If not, dynamic data authentication has failed.

10. If the ICC Public Key Algorithm Indicator is not recognised, dynamic data authentication has failed.

11. If all the checks above are correct, concatenate the Leftmost Digits of the ICC Public Key and the ICC Public Key Remainder (if present) to obtain the ICC Public Key Modulus, and continue with the actual dynamic data authentication described in the two sections below.

2.5 Dynamic Signature Generation

1. After successfully retrieving the ICC Public Key as described above, the terminal issues an INTERNAL AUTHENTICATE command including the concatenation of the data elements specified by the DDOL according to the rules specified in Part II of this specification.

   The ICC may contain the DDOL, but there shall be a default DDOL in the terminal, specified by the payment system, for use in case the DDOL is not present in the ICC.

   It is mandatory that the DDOL contains the Unpredictable Number generated by the terminal (tag '9F37', 4 bytes binary).

   If any of the following cases occur, dynamic data authentication has failed.

   - Both the ICC and the terminal do not contain a DDOL.

   - The DDOL in the ICC does not include the Unpredictable Number.

   - The ICC does not contain a DDOL and the default DDOL in the terminal does not include the Unpredictable Number.

2. The ICC generates a digital signature as described in Annex E2.1 on the data specified in Table IV-11 using its ICC Private Key S_{IC} in conjunction the corresponding algorithm. The result is called the Signed Dynamic Application

Data.

Field Name	Length	Description	Format
Signed Data Format	1	Hex. value ‘05’	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result[1]	b
ICC Dynamic Data Length	1	Identifies the length L_{DD} of the ICC dynamic data in bytes	b
ICC Dynamic Data	L_{DD}	Dynamic data generated by and/or stored in the ICC	-
Pad Pattern	$N_{IC} - L_{DD} - 25$	$(N_{IC} - L_{DD} - 25)$ padding bytes of value ‘BB’[16]	b
Terminal Dynamic Data	var.	Concatenation of the data elements specified by the DDOL	-

Table IV-11 - Dynamic Application Data to be Signed (i.e., input to the hash algorithm)

The length L_{DD} of the ICC Dynamic Data satisfies $0 \leq L_{DD} \leq N_{IC} - 25$. The 3-9 leftmost bytes of the ICC Dynamic Data shall consist of the 1-byte length of the ICC Dynamic Number, followed by the 2-8 byte value of the ICC Dynamic Number (tag ‘9F4C’, 2-8 bytes binary). The ICC Dynamic Number is a time-variant parameter generated by the ICC (it can for example be an unpredictable number or a counter incremented each time the ICC receives and INTERNAL AUTHENTICATE command).

In addition to those specified in Table IV-8, the data objects necessary for dynamic data authentication are specified in Table IV-12.

Tag	Length	Value	Format
‘9F4B’	N_{IC}	Signed Dynamic Application Data	b
‘9F49’	var.	DDOL	b

Table IV-12 - Additional Data Objects Required for Dynamic Signature Generation and Verification

2.6 Dynamic Signature Verification

1. If the Signed Dynamic Application Data has a length different from the length of the ICC Public Key Modulus, dynamic data authentication has failed.

2. To obtain the recovered data specified in Table IV-13, apply the recovery function specified in Annex E2.1 on the Signed Dynamic Application Data using the ICC

[16] As can be seen in Annex E2.1, $N_I - 22$ bytes of the data signed is recovered from the signature. Since the length of the first three data elements in Table IV-11 is three bytes, there are $N_I - 22 - 3 = N_I - 25$ bytes remaining for the data to be stored in the signature.

Public Key in conjunction the corresponding algorithm. If the Recovered Data Trailer is not equal to 'BC', dynamic data authentication has failed.

Field Name	Length	Description	Format
Recovered Data Header	1	Hex. value '6A'	b
Signed Data Format	1	Hex. value '05'	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme[11]	b
ICC Dynamic Data Length	1	Identifies the length of the ICC dynamic data in bytes	b
ICC Dynamic Data	L_{DD}	Dynamic data generated by and/or stored in the ICC	-
Pad Pattern	$N_{IC} - L_{DD} - 25$	($N_{IC} - L_{DD} - 25$) padding bytes of value 'BB'[16]	b
Hash Result	20	Hash of the Dynamic Application Data and its related information	b
Recovered Data Trailer	1	Hex. value 'BC'	b

Table IV-13 - Format of the Data Recovered from the Signed Dynamic Application Data

3. Check the Recovered Data Header. If it is not '6A', dynamic data authentication has failed.

4. Check the Signed Data Format. If it is not '05', dynamic data authentication has failed.

5. Concatenate from left to right the second to the sixth data elements in Table IV-13 (that is, Signed Data Format through Pad Pattern), followed by the data elements specified by the DDOL.

6. Apply the indicated hash algorithm (derived from the Hash Algorithm Indicator) to the result of the concatenation of the previous step to produce the hash result.

7. Compare the calculated hash result from the previous step with the recovered Hash Result. If they are not the same, dynamic data authentication has failed.

If all the above steps were executed successfully, dynamic data authentication was successful.

3. Secure Messaging

The objectives of secure messaging are to ensure data confidentiality, data integrity and authentication of the sender. Data integrity and issuer authentication are achieved using a MAC. Data confidentiality is achieved using encipherment of the data field.

3.1 Secure Messaging Format

Secure messaging shall be according to one of the following two formats.

- **Format 1:** Secure messaging format according to ISO/IEC 7816-4, section 5.6, where the data field of the affected command uses BER-TLV encoding and encoding rules of ASN.1/ISO 8825 apply strictly. This is explicitly specified in the lowest significant nibble of the class byte of the command, which is set to ‘C’. This also implies that the command header is always integrated in MAC calculation.

- **Format 2:** Secure messaging format where the data field of the affected command does not use BER-TLV encoding for secure messaging, but may use it for other purposes. In this case, the data objects contained in the data field and corresponding lengths of these data objects shall be known by the sender of a command using secure messaging and known by the currently selected application. In compliance with ISO/IEC 7816-4, secure messaging according to Format 2 is explicitly specified in the lowest significant nibble of the class byte of the command, which is set to ‘4’.

3.2 Secure Messaging for Integrity and Authentication

3.2.1 Command Data Field

3.2.1.1 Format 1

The data field of the command is composed of the following TLV data objects as shown in Figure IV-3.

- The command data to be signed, if present.

  If the command data field is BER-TLV encoded, it shall either not belong to the context-specific class (the tag shall not lie in the range ‘80’ to ‘BF’) or shall have an odd tag (note that this may be a constructed data object).

  If the command data field is not BER-TLV encoded, it shall be encapsulated with the template ‘81’.

- The second data object is the MAC. Its tag is ‘8E’, and its length shall be in the range of four to eight bytes.

Tag 1	Length 1	Value 1	Tag 2	Length 2	Value 2
T	L	Value (L bytes)	‘8E’	‘04’-‘08’	MAC (4-8 bytes)

Figure IV-3 - Format 1 Command Data Field for Secure Messaging for Integrity and Authentication

3.2.1.2 Format 2

The data elements (including the MAC) contained in the data field and the corresponding lengths shall be known by the sender of a command using secure messaging and known by the currently selected application. The MAC is not BER-TLV coded and shall always be the last data element in the data field and its length shall be in the range of 4 to 8 bytes (see Figure IV-4).

Value 1	Value 2
Command data (if present)	MAC (4-8 bytes)

Figure IV-4 - Format 2 Command Data Field for Secure Messaging for Integrity and Authentication

3.2.2 MAC Session Key Derivation

The first step of the MAC generation for secure messaging for integrity consists of deriving a unique MAC Session Key from the ICC’s unique MAC Master Key as described in Annex E1.3.

3.2.3 MAC Computation

The MAC is computed by applying the mechanism described in Annex E1.2 with the MAC Session Key derived as described in section IV-3.2.2 to the message to be protected.

If secure messaging is according to Format 1, the message to be protected shall be constructed from the header of the command APDU (CLA INS P1 P2) and the command data (if present) according to the rules specified in ISO/IEC 7816-4, Section 5.6.

If secure messaging is according to Format 2, the message to be protected shall be constructed according to the payment scheme proprietary specifications. It shall however always contain the header of the command APDU and the command data (if present).

In all cases, if the MAC used for secure messaging has been specified as having a length less than 8 bytes, the MAC is obtained by taking the leftmost (most significant) bytes from the 8-byte result of the calculation described above.

3.3 Secure Messaging for Confidentiality

3.3.1 Command Data Field

3.3.1.1 Format 1

The format of an enciphered data object in a command data field is shown in Figure IV-5.

Tag	Length	Value
T	L	Cryptogram (enciphered data)

Figure IV-5 - Format 1 Enciphered Data Object in a Command Data Field

Depending on the plaintext data to be enciphered, ISO/IEC 7816-4 specifies the tag to be allocated to the resulting cryptogram. An odd tag shall be used if the object is to be integrated in the computation of a MAC; an even tag shall be used otherwise.

3.3.1.2 Format 2

Data encipherment is applied to the full plaintext command data field with the exception of a MAC (see Figure IV-6).

Value1	Value2
Cryptogram (enciphered data)	MAC (if present)

Figure IV-6 - Format 2 Command Data Field for Secure Messaging for Confidentiality

3.3.2 Encipherment Session Key Derivation

The first step of the encipherment/decipherment for secure messaging for confidentiality consists of deriving a unique Encipherment Session Key from the ICC's unique Encipherment Master Key as described in Annex E1.3.

3.3.3 Encipherment/Decipherment

Encipherment/decipherment of the plain/enciphered command data field takes place according to the mechanism described in Annex E1.1 with the Encipherment Session Key derived as described in the section IV-3.3.2.

4. Personal Identification Number Encipherment

If supported, Personal Identification Number (PIN) encipherment for offline PIN verification is performed by the terminal using an asymmetric based encipherment mechanism in order to ensure the secure transfer of a PIN from a secure tamper-evident PIN pad integrated in the terminal to the ICC.

More precisely, the ICC shall own a public key pair associated with PIN encipherment. The public key is then used by the PIN pad to encipher the PIN, and the private key is used by the ICC to verify the enciphered PIN.

The PIN block used in the data field to be enciphered shall be 8 bytes as shown in Section 4.2.

4.1 Keys and Certificates

If offline PIN encipherment is supported, the ICC shall own a unique public key pair consisting of a public encipherment key and the corresponding private decipherment key. This specification allows the following two possibilities.

1. The ICC owns a specific ICC PIN Encipherment Private and Public Key. The ICC PIN Encipherment Public Key shall be stored on the ICC in a public key certificate in exactly the same way as for the ICC Public Key for dynamic data authentication as specified in Section IV-2.

   The ICC PIN encipherment public key pair has an ICC PIN Encipherment Public Key Modulus of N_{PE} bytes, where $N_{PE} \leq 128$ and $N_{PE} < N_I$, N_I being the length of the Issuer Public Key Modulus (See section IV-2.1). If $N_{PE} > (N_I - 42)$, the ICC PIN Encipherment Public Key Modulus is divided into two parts, one part consisting of the $N_I - 42$ most significant bytes of the modulus (the Leftmost Digits of the ICC PIN Encipherment Public Key) and a second part consisting of the remaining $N_{PE} - (N_I - 42)$ least significant bytes of the modulus (the ICC PIN Encipherment Public Key Remainder).

   The ICC PIN Encipherment Public Key Exponent shall be equal to 3 or $2^{16}+1$.

   The ICC PIN Encipherment Public Key Certificate is obtained by applying the digital signature scheme specified in Annex E2.1 on the data in Table IV-14 using the Issuer Private Key (see Section IV-2.1).

Field Name	Length	Description	Format
Certificate Format	1	Hex. value ‘04’	b
Application PAN	10	PAN (padded to the right with hex. ‘F’s)	cn 20
Certificate Expiration Date	2	MMYY after which this certificate is invalid	n 4
Certificate Serial Number	3	Binary number unique to this certificate assigned by the issuer	b
Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme[11]	b
ICC PIN Encipherment Public Key Algorithm Indicator	1	Identifies the digital signature algorithm to be used with the ICC PIN Encipherment Public Key[11]	b
ICC PIN Encipherment Public Key Length	1	Identifies the length of the ICC PIN Encipherment Public Key Modulus in bytes	b
ICC PIN Encipherment Public Key Exponent Length	1	Identifies the length of the ICC PIN Encipherment Public Key Exponent in bytes	b
ICC PIN Encipherment Public Key or Leftmost Digits of the ICC PIN Encipherment Public Key	$N_I - 42$	If $N_{PE} \leq N_I - 42$, this field consists of the full ICC PIN Encipherment Public Key padded to the right with $N_I - 42 - N_{PE}$ bytes of value ‘BB’ If $N_{PE} > N_I - 42$, this field consists of the $N_I - 42$ most significant bytes of the ICC PIN Encipherment Public Key [17]	b
ICC PIN Encipherment Public Key Remainder	0 or $N_{IC} - N_I + 42$	This field is only present if $N_{PE} > N_I - 42$ and consists of the $N_I - N_{PE} + 42$ least significant bytes of the ICC PIN Encipherment Public Key	b
ICC PIN Encipherment Public Key Exponent	1 or 3	ICC PIN Encipherment Public Key Exponent equal to 3 or $2^{16}+1$	b

Table IV-14 - ICC PIN Encipherment Public Key Data to be Signed by the Issuer (i.e. input to the hash algorithm)

2. The ICC does not own a specific ICC PIN encipherment public key pair, but owns an ICC public key pair for dynamic data authentication a specified in

[17] As can be seen in Annex E2.1, $N_I - 22$ bytes of the data signed are retrieved from the signature. Since the first through the eighth data elements in Table IV-7 total 20 bytes, there are $N_I - 22 - 20 = N_I - 42$ bytes left for the data to be stored in the signature.

Section 2.1 of this specification. This key pair can then be used for PIN encipherment, if and only if the ICC Public Key Exponent is equal to 3 or $2^{16}+1$. The ICC Public Key is stored on the ICC in a public key certificate as specified in Section 2.

The first step of PIN encipherment shall be the retrieval of the public key to be used by the terminal for the encipherment of the PIN. This process takes place as follows.

1. If the terminal has obtained all the data objects specified in Table IV-15 from the ICC, then the terminal retrieves the ICC PIN Encipherment Public Key in exactly the same way as it retrieves the ICC Public Key for dynamic data authentication (see Section IV-2).

2. If the terminal has not obtained all the data objects specified in Table IV-15, but has obtained all the data objects specified in Table IV-8, and the ICC Public Key Exponent is equal to 3 or $2^{16}+1$, then the terminal retrieves the ICC Public Key as described in Section IV-2.

3. If the conditions under points 1 and 2 above are not satisfied, then PIN encipherment has failed.

Tag	Length	Value	Format
-	5	Registered Application Provider Identifier (RID)	b
‘8F’	1	Certification Authority Public Key Index	b
‘90’	N_{CA}	Issuer Public Key Certificate	b
‘92’	$N_I - N_{CA} + 36$	Issuer Public Key Remainder, if present	b
‘9F32’	1 or 3	Issuer Public Key Exponent	b
‘9F2D’	N_I	ICC PIN Encipherment Public Key Certificate	b
‘9F2E’	$N_{PE} - N_I + 42$	ICC PIN Encipherment Public Key Remainder, if present	b
‘9F2F’	1 or 3	ICC PIN Encipherment Public Key Exponent	b

Table IV-15 - Data Objects Required for the Retrieval of the ICC PIN Encipherment Public Key

4.2 PIN Encipherment and Verification

The exchange and verification of an enciphered PIN between terminal and ICC takes place in the following steps.

1. The PIN is entered in plaintext format on the PIN pad and a PIN block is constructed as defined in Part II of this specification.

2. The terminal issues a GET CHALLENGE command to the ICC to obtain an 8-byte unpredictable number from the ICC.

3. The terminal generates a Random Pad Pattern consisting of N – 17 bytes, where N is the length in bytes of the public key to be used for PIN encipherment retrieved as specified in Section IV-4.1 (hence $N = N_{PE}$ or $N = N_{IC}$).

4. Using the PIN Encipherment Public Key or the ICC Public Key retrieved as specified in Section IV-4.1, the terminal applies the RSA Recovery Function specified Annex F2.1.1.3 to the data specified in Table IV-16 in order to obtain the Enciphered PIN Data.

Field Name	Length	Description	Format
Data Header	1	Hex. value '7F'	b
PIN Block	8	PIN in PIN Block	b
ICC Unpredictable Number	8	Unpredictable number obtained from the ICC with the GET CHALLENGE command	b
Random Pad Pattern	$N_{IC} - 17$	Random Pad Pattern generated by the terminal	b

Table IV-16 - Data to be Enciphered for PIN Encipherment

5. The terminal issues a VERIFY command including the Enciphered PIN Data obtained in the previous step.

6. With the ICC Private Key, the ICC applies the RSA Signing Function specified in Annex F2.1.1.2 to the Enciphered PIN Data in order to recover the plain text data specified in Table IV-15.

7. The ICC verifies whether the Data Header recovered is equal to '7F'. If this is not the case, PIN verification has failed.

8. The ICC verifies whether the ICC Unpredictable Number recovered is equal to the ICC Unpredictable Number generated by the ICC with the GET CHALLENGE command. If this is not the case, PIN verification has failed.

9. The ICC verifies whether the PIN included in the recovered PIN Block corresponds with the PIN stored in the ICC. If this is not the case, PIN verification has failed.

If all the above steps were executed successfully, enciphered PIN verification was successful.

In order for this mechanism to be secure, the steps 1 through 4 must be executed in the secure environment of the tamper-evident PIN pad.

THIS PAGE LEFT INTENTIONALLY BLANK

Annexes

Annex A - Examples of Exchanges Using T=0

The following examples illustrate exchanges of data and procedure bytes between the TTL and ICC.

Note the following:

- The use of procedure bytes '60' and $\overline{\text{INS}}$ is not illustrated.
- [Data(x)] means x bytes of data.
- Case 2 and 4 commands have Le = '00' requesting the return of all data from the ICC up to the maximum available.

The examples in sections A1 to A4 illustrate typical exchanges using case 1 to 4 commands. The examples in sections A5 and A6 illustrate the more extensive use of procedure bytes '61 xx' when used with case 2 and 4 commands. The example in section A7 illustrates a warning condition with a case 4 command.

A1. Case 1 Command

A C-APDU of {CLA INS P1 P2} is passed from the TAL to the TTL (note that P3 of the C-TPDU is set to '00').

TTL		ICC
[CLA INS P1 P2 00] ⇒		
	⇐ 90 00	

A R-APDU of {90 00} is returned from the TTL to the TAL

A2. Case 2 Command

A C-APDU of {CLA INS P1 P2 00} is passed from the TAL to the TTL.

TTL		ICC
[CLA INS P1 P2 00] ⇒		
	⇐ 6C Licc	
[CLA INS P1 P2 Licc] ⇒		
	⇐ INS [Data(Licc)] 90 00	

A R-APDU of {[Data(Licc)] 90 00} is returned from the TTL to the TAL.

A3. Case 3 Command

A C-APDU of {CLA INS P1 P2 Lc [Data(Lc)]} is passed from the TAL to the TTL.

TTL	ICC
[CLA INS P1 P2 Lc] ⇒	
	⇐ INS
[Data(Lc)] ⇒	
	⇐ 90 00

A R-APDU of {90 00} is returned from the TTL to the TAL.

A4. Case 4 Command

A C-APDU of {CLA INS P1 P2 Lc [Data (Lc)] 00} is passed from the TAL to the TTL.

TTL	ICC
[CLA INS P1 P2 Lc] ⇒	
	⇐ [INS]
[Data(Lc)] ⇒	
	⇐ 61 Licc
[00 C0 00 00 Licc] ⇒	
	⇐ C0 [Data(Licc)] 90 00

A R-APDU of {[Data(Licc)] 90 00} is returned from the TTL to the TAL.

A5. Case 2 Commands Using the ‘61’ and ‘6C’ Procedure Bytes

A C-APDU of {CLA INS P1 P2 00} is passed from the TAL to the TTL.

TTL	ICC
[CLA INS P1 P2 00] ⇒	
	⇐ 6C Licc
[CLA INS P1 P2 Licc] ⇒	
	⇐ 61 xx
[00 C0 00 00 yy] ⇒	
	⇐ C0 [Data(yy)] 61 zz
[00 C0 00 00 zz] ⇒	
	⇐ C0 [Data(zz)] 90 00

Where yy ≤ xx

A R-APDU of {[Data(yy + zz)] 90 00} is returned from the TTL to the TAL.

A6. Case 4 Command Using the ‘61’ Procedure Byte

A C-APDU of {CLA INS P1 P2 Lc [Data Lc] 00} is passed from the TAL to the TTL.

TTL	**ICC**
[CLA INS P1 P2 Lc] ⇒	
	⇐ [INS]
[Data(Lc)] ⇒	
	⇐ 61 xx
[00 C0 00 00 xx] ⇒	
	⇐ C0 [Data(xx)] 61 yy
[00 C0 00 00 yy] ⇒	
	⇐ C0 [Data(yy)] 90 00

A R-APDU of {[Data(xx + yy)] 90 00} is returned from the TTL to the TAL.

A7. Case 4 Command with Warning Condition

A C-APDU of {CLA INS P1 P2 Lc [Data Lc] 00} is passed from the TAL to the TTL.

TTL	ICC
[CLA INS P1 P2 Lc] ⇒	
	⇐ [INS]
[Data(Lc)] ⇒	
	⇐ 62 xx
[00 C0 00 00 00] ⇒	
	⇐ 6C Licc
[00 C0 00 00 Licc] ⇒	
	⇐ C0 [Data(Licc)] 90 00

A R-APDU of {[Data(Licc)] 62 xx} is returned from the TTL to the TAL containing the data returned together with the warning status bytes.

THIS PAGE LEFT INTENTIONALLY BLANK

Annex B - Data Elements Table

Table B-1 defines those data elements that may be used for financial transaction interchange and their mapping onto data objects and files.

Name	Description	Source	Format	Template	Tag	Length
Acquirer Identifier	Uniquely identifies the acquirer within each payment system	Terminal	n 6-11	-	‘9F01’	6
Additional Terminal Capabilities	Indicates the data input and output capabilities of the terminal	Terminal	b	-	‘9F40’	5
Amount, Authorised (Binary)	Authorised amount of the transaction (excluding adjustments)	Terminal	b	-	‘81’	4
Amount, Authorised (Numeric)	Authorised amount of the transaction (excluding adjustments)	Terminal	n 12	-	‘9F02’	6
Amount, Other (Binary)	Secondary amount associated with the transaction representing a cashback amount	Terminal	b	-	‘9F04’	4
Amount, Other (Numeric)	Secondary amount associated with the transaction representing a cashback amount	Terminal	n 12	-	‘9F03’	6
Amount, Reference Currency	Authorised amount expressed in the reference currency	Terminal	b	-	‘9F3A’	4
Application Cryptogram	Cryptogram returned by the ICC in response of the GENERATE AC command	ICC	b	‘77’ or ‘80’	‘9F26’	8
Application Currency Code	Indicates the currency in which the account is managed according to ISO 4217	ICC	n 3	‘70’ or ‘77’	‘9F42’	2
Application Currency Exponent	Indicates the implied position of the decimal point from the right of the account represented according to ISO 4217	ICC	n 1	‘70’ or ‘77’	‘9F44’	1
Application Discretionary Data	Issuer or payment system specified data relating to the application	ICC	b	‘70’ or ‘77’	‘9F05’	1-32

Name	Description	Source	Format	Template	Tag	Length
Application Effective Date	Date from which the application may be used	ICC	n 6 YYMMDD	‘70’ or ‘77’	‘5F25’	3
Application Expiration Date	Date after which application expires	ICC	n 6 YYMMDD	‘70’ or ‘77’	‘5F24’	3
Application File Locator (AFL)	Indicates the location (SFI, range of records) of the AEFs related to a given application	ICC	var.	‘77’ or ‘80’	‘94’	var. up to 252
Application Identifier (AID)	Identifies the application as described in ISO/IEC 7816-5	ICC	b	‘61’	‘4F’	5-16
Application Identifier (AID)	Identifies the application as described in ISO/IEC 7816-5	Terminal	b	-	‘9F06’	5-16
Application Interchange Profile	Indicates the capabilities of the card to support specific functions in the application	ICC	b	‘77’ or ‘80’	‘82’	2
Application Label	Mnemonic associated with the AID according to ISO/IEC 7816-5	ICC	an 1-16	‘61’ or ‘A5’	‘50’	1-16
Application Preferred Name	Preferred mnemonic associated with the AID	ICC	an 1-16	‘61’	‘9F12’	1-16
Application Primary Account Number (PAN)	Valid cardholder account number	ICC	cn var. up to 19	‘70’ or ‘77’	‘5A’	var. up to 10
Application Primary Account Number (PAN) Sequence Number	Identifies and differentiates cards with the same PAN	ICC	n 2	‘70’ or ‘77’	‘5F34’	1
Application Priority Indicator	Indicates the priority of a given application or group of applications in a directory	ICC	b	‘61’ or ‘A5’	‘87’	1
Application Reference Currency	1-4 currency codes used between the terminal and the ICC when the Transaction Currency Code is different from the Application Currency Code; each code is 3 digits according to ISO 4217	ICC	n 3	‘70’ or ‘77’	‘9F3B’	2-8

Name	Description	Source	Format	Template	Tag	Length
Application Reference Currency Exponent	Indicates the implied position of the decimal point from the right of the amount, for each of the 1-4 reference currencies represented according to ISO 4217	ICC	n 1	‘70 or 77’	‘9F43’	1-4
Application Template	Contains one or more data objects relevant to an application directory entry according to ISO/IEC 7816-5	ICC	b	‘70’ or ‘77’	‘61’	var. up to 252
Application Transaction Counter (ATC)	Counter maintained by the application in the ICC (incrementing the ATC is managed by the ICC)	ICC	b	‘77’ or ‘80’	‘9F36’	2
Application Usage Control	Indicates issuer's specified restrictions on the geographic usage and services allowed for the application	ICC	b	‘70’ or ‘77’	‘9F07’	2
Application Version Number	Version number assigned by the payment system for the application	ICC	b	‘70’ or ‘77’	‘9F08’	2
Application Version Number	Version number assigned by the payment system for the application	Terminal	b	-	‘9F09’	2
Authorisation Code	Value generated by the issuer for an approved transaction	Issuer	an 6	-	‘89’	6
Authorisation Response Code	Code that defines the disposition of a message	Issuer/ Terminal	an 2	-	‘8A’	2
Card Risk Management Data Object List 1 (CDOL1)	List of data objects (tag and length) to be passed to the ICC in the first GENERATE AC command	ICC	b	‘70’ or ‘77’	‘8C’	var. up to 252
Card Risk Management Data Object List 2 (CDOL2)	List of data objects (tag and length) to be passed to the ICC in the second GENERATE AC command	ICC	b	‘70’ or ‘77’	‘8D’	var. up to 252
Cardholder Name	Indicates cardholder name according to ISO 7813	ICC	ans 2-26	‘70’ or ‘77’	‘5F20’	2-26
Cardholder Name Extended	Indicates the whole cardholder name when greater than 26 characters using the same coding convention as in ISO 7813	ICC	ans 27-45	‘70’ or ‘77’	‘9F0B’	27-45

Name	Description	Source	Format	Template	Tag	Length
Cardholder Verification Method (CVM) List	Identifies a method of verification of the cardholder supported by the application	ICC	b	‘70’ or ‘77’	‘8E’	var. up to 252
Cardholder Verification Method (CVM) Results	Indicates the results of the last CVM performed	Terminal	b	-	‘9F34’	3
Certification Authority Public Key Index	Identifies the certification authority's public key in conjunction with the RID	ICC	b	‘70’ or ‘77’	‘8F’	1
Certification Authority Public Key Index	Identifies the certification authority's public key in conjunction with the RID	Terminal	b	-	‘9F22’	1
Command Template	Identifies the data field of a command message	Terminal	b		‘83’	var.
Command to Perform	Series of bytes to be delivered by the terminal to the ICC as a command APDU for the purpose of selecting either a DDF or an ADF.	ICC	b	‘61’	‘52’	var. up to 239
Cryptogram Information Data	Indicates the type of cryptogram and the actions to be performed by the terminal	ICC	b	‘77’ or ‘80’	‘9F27’	1
Data Authentication Code	An issuer assigned value that is retained by the terminal during the verification process of the Signed Static Application Data	ICC	b		‘9F45’	2
Dedicated File (DF) Name	Identifies the name of the DF as described in ISO/IEC 7816-4	ICC	b	‘6F’	‘84’	5-16
Directory Definition File (DDF) Name	Identifies the name of a DF associated with a directory	ICC	b	‘61’	‘9D’	5-16
Directory Discretionary Template	Issuer discretionary part of the directory according to ISO/IEC 7816-5	ICC	var.	‘61’	‘73’	var. up to 252

Name	Description	Source	Format	Template	Tag	Length
Dynamic Data Authentication Data Object List (DDOL)	List of data objects (tag and length) to be passed to the ICC in the INTERNAL AUTHENTICATE command	ICC	b	‘70’ or ‘77’	‘9F49’	up to 252
File Control Information (FCI) Issuer Discretionary Data	Issuer discretionary part of the FCI	ICC	var.	‘A5’	‘BF0C’	var. up to 222
File Control Information (FCI) Proprietary Template	Identifies the data object proprietary to this specification in the FCI template according to ISO/IEC 7816-4	ICC	var.	‘6F’	‘A5’	var.
File Control Information (FCI) Template	Identifies the FCI template according to ISO/IEC 7816-4	ICC	var.	-	‘6F’	var. up to 252
ICC Dynamic Number	Time-variant number generated by the ICC, to be captured by the terminal	ICC	b		‘9F4C’	2-8
Integrated Circuit Card (ICC) PIN Encipherment Public Key Certificate	ICC PIN Encipherment Public Key certified by the issuer	ICC	b	‘70’ or ‘77’	‘9F2D’	N_I
Integrated Circuit Card (ICC) PIN Encipherment Public Key Exponent	ICC PIN Encipherment Public Key Exponent used for PIN encipherment	ICC	b	‘70’ or ‘77’	‘9F2E’	1 or 3

Name	Description	Source	Format	Template	Tag	Length
Integrated Circuit Card (ICC) PIN Encipherment Public Key Remainder	Remaining digits of the ICC PIN Encipherment Public Key Modulus	ICC	b	‘70’ or ‘77’	‘9F2F’	$N_{PE} – N_I + 42$
Integrated Circuit Card (ICC) Public Key Certificate	ICC Public Key certified by the issuer	ICC	b	‘70’ or ‘77’	‘9F46’	N_I
Integrated Circuit Card (ICC) Public Key Exponent	ICC Public Key Exponent used for the verification of the Signed Dynamic Application Data	ICC	b	‘70’ or ‘77’	‘9F47’	1 to 3
Integrated Circuit Card (ICC) Public Key Remainder	Remaining digits of the ICC Public Key Modulus	ICC	b	‘70’ or ‘77’	‘9F48’	$N_{IC} – N_I + 42$
Interface Device (IFD) Serial Number	Unique and permanent serial number assigned to the IFD by the manufacturer	Terminal	an 8	-	‘9F1E’	8
Issuer Action Code - Default	Specifies the issuer's conditions that cause a transaction to be rejected if it might have been approved online, but the terminal is unable to process the transaction online	ICC	b	‘70’ or ‘77’	‘9F0D’	5
Issuer Action Code - Denial	Specifies the issuer's conditions that cause the denial of a transaction without attempt to go online	ICC	b	‘70’ or ‘77’	‘9F0E’	5
Issuer Action Code - Online	Specifies the issuer's conditions that cause a transaction to be transmitted online	ICC	b	‘70’ or ‘77’	‘9F0F’	5
Issuer Application Data	Contains proprietary application data for transmission to the issuer in an online transaction	ICC	b	‘77’ or ‘80’	‘9F10’	var. up to 32
Issuer Authentication Data	Data sent to the ICC for online issuer authentication	Issuer	b	-	‘91’	8-16

Name	Description	Source	Format	Template	Tag	Length
Issuer Code Table Index	Indicates the code table according to ISO 8859 for displaying the Application Preferred Name	ICC	n 2	'A5'	'9F11'	1
Issuer Country Code	Indicates the country of the issuer according to ISO 3166	ICC	n 3	'70' or '77'	'5F28'	2
Issuer Public Key Certificate	Issuer public key certified by a certification authority	ICC	b	'70' or '77'	'90'	N_{CA}
Issuer Public Key Exponent	Issuer public key exponent used for the verification of the Signed Static Application Data and the ICC Public Key Certificate	ICC	b	'70' or '77'	'9F32'	1 to 3
Issuer Public Key Remainder	Remaining digits of the Issuer Public Key Modulus	ICC	b	'70' or '77'	'92'	$N_I - N_{CA} + 36$
Issuer Script Command	Contains a command for transmission to the ICC	Issuer	b	'71' or '72'	'86'	var. up to 261
Issuer Script Identifier	Identification of the Issuer Script	Issuer	b	'71' or '72'	'9F18'	4
Issuer Script Template 1	Contains proprietary issuer data for transmission to the ICC before the second GENERATE AC command	Issuer	b	-	'71'	var.
Issuer Script Template 2	Contains proprietary issuer data for transmission to the ICC after the second GENERATE AC command	Issuer	b	-	'72'	var.
Language Preference	1-4 languages stored in order of preference, each represented by 2 alphabetical characters according to ISO 639	ICC	an 2	'A5'	'5F2D'	2-8
Last Online Application Transaction Counter (ATC) Register	ATC value of the last transaction that went online	ICC	b	-	'9F13'	2
Lower Consecutive Offline Limit	Issuer-specified preference for the maximum number of consecutive offline transactions for this ICC application allowed in a terminal with online capability	ICC	b	'70' or '77'	'9F14'	1
Merchant Category Code	Classifies the type of business being done by the merchant, represented according to ISO 8583:1993 for Card Acceptor Business Code	Terminal	n 4	-	'9F15'	2

Name	Description	Source	Format	Template	Tag	Length
Merchant Identifier	When concatenated with the Acquirer Identifier, uniquely identifies a given merchant	Terminal	ans 15	-	'9F16'	15
Personal Identification Number (PIN) Try Counter	Number of PIN tries remaining	ICC	b	-	'9F17'	1
Point-of-Service (POS) Entry Mode	Indicates the method by which the PAN was entered, according to the first two digits of the ISO 8583:1987 POS Entry Mode	Terminal	n 2	-	'9F39'	1
Processing Options Data Object List (PDOL)	Contains a list of terminal resident data objects (tags and lengths) needed by the ICC in processing the GET PROCESSING OPTIONS command	ICC	b	'A5'	'9F38'	var.
Response Message Template Format 1	Contains the data objects (without tags and lengths) returned by the ICC in response to a command	ICC	var.		'80'	var.
Response Message Template Format 2	Contains the data objects (with tags and lengths) returned by the ICC in response to a command	ICC	var.		'77'	var.
Service Code	Service code as defined on tracks 1 and 2	ICC	n 3	'70' or '77'	'5F30'	2
Short File Identifier (SFI)	Identifies the SFI to be used in the commands related to a given AEF or DDF. The SFI data object is a binary field with the three high order bits set to zero.	ICC	b	'A5'	'88'	1
Signed Dynamic Application Data	Digital signature on critical application parameters for dynamic data authentication	ICC	b	-	'9F4B'	N_{IC}
Signed Static Application Data	Digital signature on critical application parameters for static data authentication	ICC	b	'70' or '77'	'93'	N_I
Static Data Authentication Tag List	List of tags of primitive data objects defined in this specification whose value fields are to be included in the Signed Static or Dynamic Application Data	ICC	--	'70' or '77'	'9F4A'	var.

Name	Description	Source	Format	Template	Tag	Length
Terminal Capabilities	Indicates the card data input, CVM, and security capabilities of the terminal	Terminal	b	-	‘9F33’	3
Terminal Country Code	Indicates the country of the terminal, represented according to ISO 3166	Terminal	n 3	-	‘9F1A’	2
Terminal Floor Limit	Indicates the floor limit in the terminal in conjunction with the AID	Terminal	b		‘9F1B’	4
Terminal Identification	Designates the unique location of a terminal at a merchant	Terminal	an 8	-	‘9F1C’	8
Terminal Risk Management Data	Application-specific value used by the card for risk management purposes	Terminal	b	-	‘9F1D’	1-8
Terminal Type	Indicates the environment of the terminal, its communications capability, and its operational control	Terminal	n 2	-	‘9F35’	1
Terminal Verification Results	Status of the different functions as seen from the terminal	Terminal	b	-	‘95’	5
Track 1 Discretionary Data	Discretionary part of track 1 according to ISO/IEC 7813	ICC	ans	‘70’ or ‘77’	‘9F1F’	var.
Track 2 Discretionary Data	Discretionary part of track 2 according to ISO/IEC 7813	ICC	cn	‘70’ or ‘77’	‘9F20’	var.

Name	Description	Source	Format	Template	Tag	Length
Track 2 Equivalent Data	Contains the data elements of the track 2 according to ISO/IEC 7813, excluding start sentinel, end sentinel, and LRC, as follows:	ICC	b	'70' or '77'	'57'	var. up to 19
	Primary Account Number		n, var. up to 19			
	Field Separator (hex. 'D')		b			
	Expiration Date (YYMM)		n 4			
	Service Code		n 3			
	Discretionary Data(define by individual payment systems)		n, var.			
	Pad with hex. 'FF' if needed to ensure whole bytes		b			
Transaction Certificate Data Object List (TDOL)	List of data objects (tag and length) to be used by the terminal in generating the TC Hash Value	ICC	b	'70' or '77'	'97'	var. up to 252
Transaction Certificate (TC) Hash Value	Result of a hash function specified in Annex F3 of this specification	Terminal	b	-	'98'	20
Transaction Currency Code	Indicates the currency code of the transaction according to ISO 4217	Terminal	n 3	-	'5F2A'	2
Transaction Currency Exponent	Indicates the implied position of the decimal point from the right of the transaction amount represented according to ISO 4217	Terminal	n 1	-	'5F36'	1
Transaction Date	Local date that the transaction was authorised	Terminal	n 6 YYMMDD	-	'9A'	3

Name	Description	Source	Format	Template	Tag	Length
Transaction Personal Identification Number (PIN) Data	Data entered by the cardholder for the purpose of the PIN verification	Terminal	b	-	‘99’	var.
Transaction Reference Currency Code	Code defining the common currency used by the terminal in case the Transaction Currency Code is different from the Application Currency Code	Terminal	n 3	-	‘9F3C’	2
Transaction Reference Currency Exponent	Indicates the implied position of the decimal point from the right of the transaction amount, with the Transaction Reference Currency Code represented according to ISO 4217	Terminal	n 1	-	‘9F3D’	1
Transaction Sequence Counter	Counter maintained by the terminal that is incremented by one for each transaction	Terminal	n 4-8	-	‘9F41’	2-4
Transaction Status Information	Indicates the functions performed in a transaction	Terminal	b	-	‘9B’	2
Transaction Time	Local time that the transaction was authorised	Terminal	n 6 HHMMSS	-	‘9F21’	3
Transaction Type	Indicates the type of financial transaction, represented by the first two digits of ISO 8583:1987 Processing Code	Terminal	n 2	-	‘9C’	1
Unpredictable Number	Value to provide variability and uniqueness to the generation of a cryptogram	Terminal	b	-	‘9F37’	4
Upper Consecutive Offline Limit	Issuer-specified preference for the maximum number of consecutive offline transactions for this ICC application allowed in a terminal without online capability	ICC	b	‘70’ or ‘77’	‘9F23’	1

Table B-1 - Data Elements Dictionary

When the length defined for the data object is greater than the length of the actual data, the following rules apply:

- A data element in format n is right justified and padded with leading hexadecimal zeroes
- A data element in format cn is left justified and padded with trailing hexadecimal 'F's
- A data element in format an is left justified and padded with trailing hexadecimal zeroes
- A data element in format ans is left justified and padded with trailing hexadecimal zeroes

When data is moved from one entity to another (for example, card to terminal), it shall always be passed in order from high order to low order, regardless of how it is internally stored. The same rule applies when concatenating data.

Note: Data that can occur in template '70' or '77' can never occur in both.

The tags allocated to the data elements are according to Table B-2:

Name	Template	Tag
Application Identifier (AID)	‘61’	‘4F’
Application Label	‘61’	‘50’
Command to Perform	‘61’	‘52’
Track 2 Equivalent Data	‘70’ or ‘77’	‘57’
Application Primary Account Number (PAN)	‘70’ or ‘77’	‘5A’
Cardholder Name	‘70’ or ‘77’	‘5F20’
Application Expiration Date	‘70’ or ‘77’	‘5F24’
Application Effective Date	‘70’ or ‘77’	‘5F25’
Issuer Country Code	‘70’ or ‘77’	‘5F28’
Transaction Currency Code	–	‘5F2A’
Language Preference	‘A5’	‘5F2D’
Service Code	‘70’ or ‘77’	‘5F30’
Application Primary Account Number (PAN) Sequence Number	‘70’ or ‘77’	‘5F34’
Transaction Currency Exponent	–	‘5F36’
Application Template	‘70’ or ‘77’	‘61’
File Control Information (FCI) Template	–	‘6F’
Application Elementary File (AEF) Data Template	–	‘70’
Issuer Script Template 1	–	‘71’
Issuer Script Template 2	–	‘72’
Directory Discretionary Template	‘61’	‘73’
Response Message Template Format 2	–	‘77’
Response Message Template Format 1	–	‘80’
Amount, Authorised (Binary)	–	‘81’
Application Interchange Profile	‘77’ or ‘80’	‘82’
Command Template	–	‘83’
Dedicated File (DF) Name	‘6F’	‘84’
Issuer Script Command	‘71’ or ‘72’	‘86’
Application Priority Indicator	‘61’ or ‘A5’	‘87’
Short File Identifier (SFI)	‘A5’	‘88’
Authorisation Code	–	‘89’
Authorisation Response Code	–	‘8A’
Card Risk Management Data Object List 1 (CDOL1)	‘70’ or ‘77’	‘8C’
Card Risk Management Data Object List 2 (CDOL2)	‘70’ or ‘77’	‘8D’
Cardholder Verification Method (CVM) List	‘70’ or ‘77’	‘8E’
Certification Authority Public Key Index	‘70’ or ‘77’	‘8F’
Issuer Public Key Certificate	‘70’ or ‘77’	‘90’
Issuer Authentication Data	–	‘91’
Issuer Public Key Remainder	‘70’ or ‘77’	‘92’
Signed Static Application Data	‘70’ or ‘77’	‘93’
Application File Locator (AFL)	‘77’ or ‘80’	‘94’
Terminal Verification Results	–	‘95’
Transaction Certificate Data Object List (TDOL)	‘70’ or ‘77’	‘97’
Transaction Certificate (TC) Hash Value	–	‘98’
Transaction Personal Identification Number (PIN) Data	–	‘99’
Transaction Date	–	‘9A’

Name	Template	Tag
Transaction Status Information	–	‘9B’
Transaction Type	–	‘9C’
Directory Definition File (DDF) Name	‘61’	‘9D’
Acquirer Identifier	–	‘9F01’
Amount, Authorised (Numeric)	–	‘9F02’
Amount, Other (Numeric)	–	‘9F03’
Amount, Other (Binary)	–	‘9F04’
Application Discretionary Data	‘70’ or ‘77’	‘9F05’
Application Identifier (AID)	–	‘9F06’
Application Usage Control	‘70’ or ‘77’	‘9F07’
Application Version Number	‘70’ or ‘77’	‘9F08’
Application Version Number	–	‘9F09’
Cardholder Name -Extended	‘70’ or ‘77’	‘9F0B’
Issuer Action Code - Default	‘70’ or ‘77’	‘9F0D’
Issuer Action Code - Denial	‘70’ or ‘77’	‘9F0E’
Issuer Action Code - Online	‘70’ or ‘77’	‘9F0F’
Issuer Application Data	‘77’ or ‘80’	‘9F10’
Issuer Code Table Index	‘A5’	‘9F11’
Application Preferred Name	‘61’	‘9F12’
Last Online Application Transaction Counter (ATC) Register	–	‘9F13’
Lower Consecutive Offline Limit	‘70’ or ‘77’	‘9F14’
Merchant Category Code	–	‘9F15’
Merchant Identifier	–	‘9F16’
Personal Identification Number (PIN) Try Counter	–	‘9F17’
Issuer Script Identifier	‘71’ or ‘72’	‘9F18’
Terminal Country Code	–	‘9F1A’
Terminal Floor Limit	–	‘9F1B’
Terminal Identification	–	‘9F1C’
Terminal Risk Management Data	–	‘9F1D’
Interface Device (IFD) Serial Number	–	‘9F1E’
Track 1 Discretionary Data	‘70’ or ‘77’	‘9F1F’
Track 2 Discretionary Data	‘70’ or ‘77’	‘9F20’
Transaction Time	–	‘9F21’
Certification Authority Public Key Index	–	‘9F22’
Upper Consecutive Offline Limit	‘70’ or ‘77’	‘9F23’
Application Cryptogram	‘77’ or ‘80’	‘9F26’
Cryptogram Information Data	‘77’ or ‘80’	‘9F27’
ICC PIN Encipherment Public Key Certificate	‘70’ or ‘77’	‘9F2D’
ICC PIN Encipherment Public Key Exponent	‘70’ or ‘77’	‘9F2E’
ICC PIN Encipherment Public Key Remainder	‘70’ or ‘77’	‘9F2F’
Issuer Public Key Exponent	‘70’ or ‘77’	‘9F32’
Terminal Capabilities	–	‘9F33’
Cardholder Verification Method (CVM) Results	–	‘9F34’
Terminal Type	–	‘9F35’
Application Transaction Counter (ATC)	‘77’ or ‘80’	‘9F36’
Unpredictable Number	–	‘9F37’
Processing Options Data Object List (PDOL)	‘A5’	‘9F38’

Name	Template	Tag
Point-of-Service (POS) Entry Mode	–	‘9F39’
Amount, Reference Currency	–	‘9F3A’
Application Reference Currency	‘70’ or ‘77’	‘9F3B’
Transaction Reference Currency	–	‘9F3C’
Transaction Reference Currency Exponent	–	‘9F3D’
Additional Terminal Capabilities	–	‘9F40’
Transaction Sequence Counter	–	‘9F41’
Application Currency Code	‘70’ or ‘77’	‘9F42’
Application Reference Currency Exponent	‘70’ or ‘77’	‘9F43’
Application Currency Exponent	-	‘9F44’
Data Authentication Code	-	‘9F45’
ICC Public Key Certificate	‘70’ or ‘77’	‘9F46’
ICC Public Key Exponent	‘70’ or ‘77’	‘9F47’
ICC Public Key Remainder	‘70’ or ‘77’	‘9F48’
Dynamic Data Object List (DDOL)	‘70’ or ‘77’	‘9F49’
Static Data Authentication Tag List	-	‘9F4A’
Signed Dynamic Application Data	-	‘9F4B’
File Control Information (FCI) Proprietary Template	‘6F’	‘A5’
File Control Information (FCI) Issuer Discretionary Data	‘A5’	‘BF0C’

Table B-2 - Data Elements Tags

THIS PAGE LEFT INTENTIONALLY BLANK

Annex C - Data Objects

C1. Coding of BER-TLV Data Objects

As defined in ISO/IEC 8825, a BER-TLV data object consists of 2-3 consecutive fields:

- The tag field (T) consists of one or more consecutive bytes. It codes a class, a type, and a number (see Table C-1). The tag field of the data objects described in this specification is coded on one or two bytes.
- The length field (L) consists of one or more consecutive bytes. It codes the length of the following field. The length field of the data objects described in this specification is coded on one, two, or three bytes.
- The value field (V) codes the value of the data object. If L = '00', the value field is not present.

A BER-TLV data object belongs to one of the following two categories:

- A primitive data object where the value field contains a data element for financial transaction interchange.
- A constructed data object where the value field contains one or more primitive or constructed data objects. The value field of a constructed data object is called a template.

The coding of BER-TLV data objects is defined for as follows.

C1.1 Coding of the Tag Field of BER-TLV Data Objects

The first byte of the tag field of a BER-TLV data object is according to Table C-1:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0							Universal class
0	1							Application class
1	0							Context-specific class
1	1							Private class
		0						Primitive data object
		1						Constructed data object
			1	1	1	1	1	See subsequent bytes
			Any other value					Tag number

Table C-1 - Tag Field Structure (First Byte) BER-TLV

According to ISO/IEC 8825, Table C-2 defines the coding rules of the subsequent bytes of a BER-TLV tag when tag numbers ≥ 31 are used (that is, bits b5 - b1 of the first byte equal '11111').

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1								Another byte follows
0								Last tag byte
	Any value > 0							(Part of) tag number

Table C-2 - Tag Field Structure (Subsequent Bytes) BER-TLV

The coding of the tag field of a BER-TLV data object is according to the following rules:

- The following application class templates defined in ISO/IEC 7816 apply: '61' and '6F.'
- The following range of application class templates is defined in Part II: '70' to '7F'. The meaning is then specific to the context of an application according to this specification. Tags '78', '79', '7D', and '7E' are defined in ISO/IEC 7816-6 and are not used in this specification.
- The application class data objects defined in ISO/IEC 7816 and described in Part II are used according to the ISO/IEC 7816 definition.
- Context-specific class data objects are defined in the context of this specification or in the context of the template in which they appear.
- The coding of primitive context-specific class data objects in the ranges '80' to '9E' and '9F00' to '9F4F' is reserved for this specification.
- The coding of primitive context-specific class data objects in the range '9F50' to '9F7F' is reserved for the payment systems.
- The coding of primitive and constructed private class data objects is left to the discretion of the issuer.

C1.2 Coding of the Length Field of BER-TLV Data Objects

The coding of the length field is as follows.

When bit b8 of the most significant byte of the length field is set to 0, the length field consists of only one byte. Bits b7 to b1 code the number of bytes of the value field. The length field is within the range 1 to 127.

When bit b8 of the most significant byte of the length field is set to 1, the subsequent bits b7 to b1 of the most significant byte code the number of subsequent bytes in the length field. The subsequent bytes code an integer representing the number of bytes in the value field. For example, three bytes are necessary to express up to 65,535 bytes in the value field.

C1.3 Coding of the Value Field of Data Objects

A data element is the value field (V) of a primitive BER-TLV data object. A data element is the smallest data field that receives an identifier (a tag).

A primitive data object is structured according to Table C-3:

Tag (T)	Length (L)	Value (V)

Table C-3 - Primitive BER-TLV Data Object (Data Element)

A constructed BER-TLV data object consists of a tag, a length, and a value field composed of one or more BER-TLV data objects. A record in an AEF governed by this specification is a constructed BER-TLV data object. A constructed data object is structured according to Table C-4:

Tag (T)	Length (L)	Primitive or constructed BER-TLV data object number 1	...	Primitive or constructed BER-TLV data object number n

Table C-4 - Constructed BER-TLV Data Object

THIS PAGE LEFT INTENTIONALLY BLANK

Annex D - Examples of Directory Structures

D1. Examples of Directory Structures

Examples shown in this annex are intended to illustrate some possible logical ICC file structures. Hierarchies of directory structures are shown, but there is no implication as to the file hierarchies as defined by ISO.

Figure D-1 illustrates a single application card with only a single level directory. In this example, the MF (with file identification of '3F00', as defined by ISO/IEC 7816-4) acts as the only DDF in the card. The MF shall be given the unique payment systems name assigned to the first level DDF as defined in Section III-1.2, and the FCI of the MF shall contain the SFI data object.

'DIR A' in this example may or may not be the ISO DIR file, but it shall conform to this specification, including the requirement that it has an SFI in the range 1 to 10. The ISO DIR file has a file identifier of '2F00', which may imply that the SFI is not in the correct range.

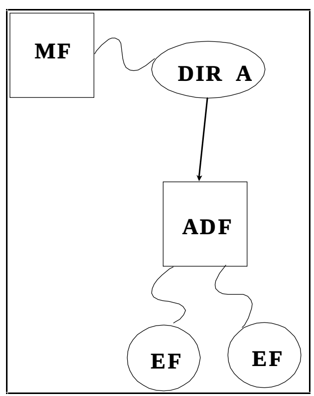

Figure D-1 - Simplest Card Structure Single Application

Figure D-2 gives an example of a multi-application card with a single directory. In this example, the root file (MF) does not support an application complying with this specification, and no restrictions are placed on the function of the MF. According to ISO/IEC 7816-4, a DIR file may be present but is not used by the application selection algorithm defined in Part III. Also note that the directory does not have entries for all ADFs (ADF2 to ADF5), as ADF5 is omitted. ADF5 can be selected only by a terminal that 'knows' ADF5 may exist in the card. The manner in which the terminal finds ADF5 is outside the scope of this specification.

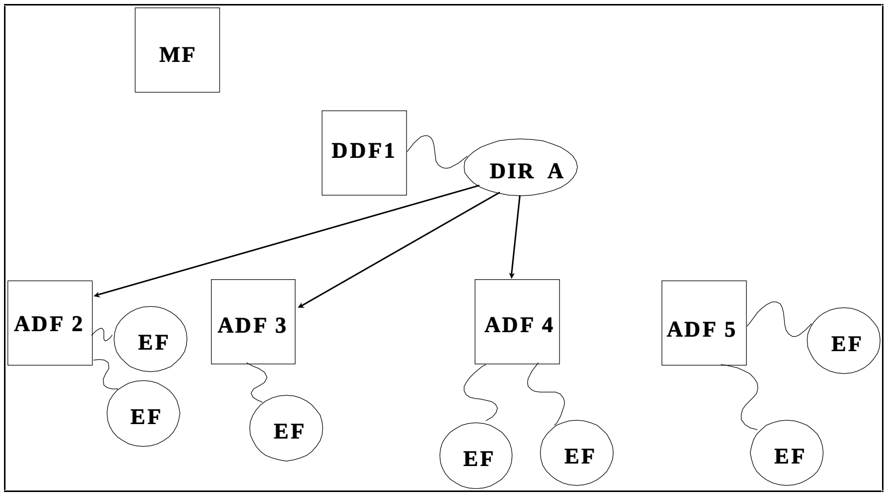

Figure D-2 - Single Level Directory

Figure D-3 is an example of a multi-application card with an *n* level directory structure. The first level directory ('DIR A') has entries for 2 ADFs – ADF3 and ADF4 – and a single DDF – DDF2. The directory attached to DDF2 ('DIR B') has entries for two ADFs – ADF21 and ADF22 – and a single DDF – DDF6. DDF5 has no entry in the root directory and can be found only by a terminal that 'knows' of the existence of DDF5. The manner in which the terminal finds and selects DDF5 is outside the scope of this specification, but the directory attached to DF5 ('DIR C') may conform to this specification, and, if found by the terminal, may lead the terminal to ADFs such as DF51, DF52, and DF53. DIR D, attached to DDF6, is a third level directory and points to four files (not shown), which may be either ADFs or more DDFs.

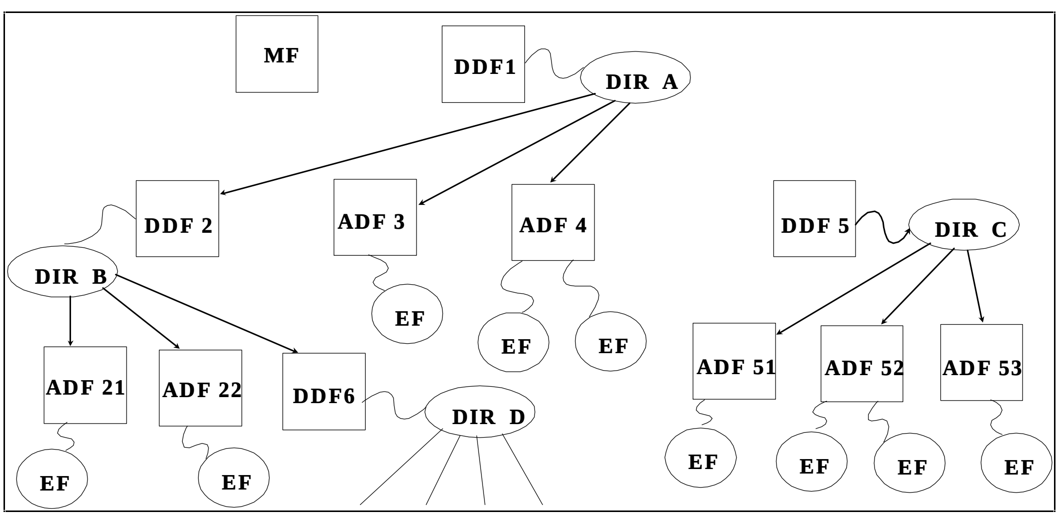

Figure D-3 - Third Level Directory

Annex E - Security Mechanisms

E1. Symmetric Mechanisms

E1.1 Encipherment

The encipherment mechanisms for secure messaging for confidentiality uses a 64-bit block cipher ALG either in Electronic Codebook (ECB) Mode or in Cipher Block Chaining (CBC) mode.

Encipherment of a message MSG of arbitrary length with Encipherment Session Key K_S takes place in the following steps.

1. *Padding and Blocking*

   - If the message MSG has a length that is not a multiple of 8 bytes, add a '80' byte to the right of MSG, and then add the smallest number of '00' bytes to the right such that the length of resulting message <u>MSG</u> := (MSG || '80' || '00' || '00' || . . . || '00') is a multiple of 8 bytes.

   - If the message MSG has a length that is a multiple of 8 bytes, the following two cases can occur depending on pre-defined rules (see Part IV of this specification).

     - No padding takes place: <u>MSG</u> := MSG.

     - MSG is padded to the right with the 8-byte block

       ('80' || '00' || '00' ||'00' || '00' || '00' || '00' || '00')

       to obtain <u>MSG</u>.

   <u>MSG</u> is then divided into 8-byte blocks $X_1, X_2, \ldots, X_k$.

2. *Cryptogram Computation*

   <u>ECB Mode</u>

   Encipher the blocks $X_1, X_2, \ldots, X_k$ into the 8-byte blocks $Y_1, Y_2, \ldots, Y_k$ with the block cipher algorithm in ECB mode using the Encipherment Session Key K_S. Hence compute for i = 1, 2, . . . , k:

$$Y_i := ALG(K_S)[X_i].$$

<u>CBC Mode</u>

Encipher the blocks $X_1, X_2, \ldots, X_k$ into the 8-byte blocks $Y_1, Y_2, \ldots, Y_k$ with the block cipher algorithm in CBC mode using the Encipherment Session Key K_S. Hence compute for i = 1, 2, . . . , k:

$$Y_i := ALG(K_S)[X_i \oplus Y_{i-1}],$$

with initial value Y_0 := (‘00’ | | ‘00’ | | ‘00’ | | ‘00’ | | ‘00’ | | ‘00’ | | ‘00’ | | ‘00’).

Notation:

$$Y := (Y_1 \,||\, Y_2 \,||\, \ldots \,||\, Y_k) = ENC(K_S)[MSG].$$

Decipherment is as follows.

1. *Cryptogram Decipherment*

   <u>ECB Mode</u>

   Compute for i = 1, 2, . . . , k:

   $$X_i := ALG^{-1}(K_S)[Y_i\,].$$

   <u>CBC Mode</u>

   Compute for i = 1, 2, . . . , k:

   $$X_i := ALG^{-1}(K_S)[Y_i\,] \oplus Y_{i-1},$$

   with initial value Y_0 := (‘00’ | | ‘00’ | | ‘00’ | | ‘00’ | | ‘00’ | | ‘00’ | | ‘00’ | | ‘00’).

2. To obtain the original message MSG, concatenate the blocks $X_1, X_2, \ldots, X_k$ and if padding has been used (see above) remove the trailing (‘80’ | | ‘00’ | | ‘00’ | | . . . | | ‘00’) byte-string from the last block X_k.

Notation:

$$M = DEC(K_S)[Y].$$

E1.2 Message Authentication Code

The computation of an s-byte MAC ($4 \le s \le 8$) is according to ISO/IEC 9797 using a 64-bit block cipher ALG in CBC mode. More precisely, the computation of a MAC S over a message MSG consisting of an arbitrary number of bytes with a MAC Session Key K_S takes place in the following steps.

1. *Padding and Blocking*

   Pad the message M according to ISO/IEC 7816-4 (which is equivalent to method 2 of ISO/IEC 9797), hence add a mandatory ‘80’ byte to the right of MSG, and then add the smallest number of ‘00’ bytes to the right such that the length of resulting message $\underline{MSG}$:= (MSG || '80' || '00' || '00' || . . . || '00') is a multiple of 8 bytes.

$\underline{MSG}$ is then divided into 8-byte blocks $X_1, X_2, \ldots, X_k$.

2. *MAC Session Key*

The MAC Session Key K_S either consists of only a leftmost key block $K_S = K_{SL}$ or the concatenation of a leftmost and a rightmost key block $K_S = (K_{SL} \,||\, K_{SR})$.

3. *Cryptogram Computation*

Process the 8-byte blocks $X_1, X_2, \ldots, X_k$ with the block cipher in CBC mode using the leftmost MAC Session Key block K_{SL}:

$$H_i := ALG(K_{SL})[X_i \oplus H_{i-1}], \text{ for } i = 1, 2, \ldots, k.$$

with initial value H_0 := ('00' || '00' || '00' || '00' || '00' || '00' || '00' || '00').

- Compute the 8 byte block H_{k+1} in one of the following two ways. According to the main process of ISO/IEC 9797:

$$H_{k+1} := H_k.$$

- According to Optional Process 1 of ISO/IEC 9797:

$$H_{k+1} := ALG(K_{SL})[ALG^{-1}(K_{SR})[H_k]].$$

The MAC S is then equal to the s most significant bytes of H_{k+1}.

E1.3 Session Key Derivation

Session keys K_S for secure messaging for integrity and confidentiality are derived from unique Master Keys K_M using diversification data R provided by the receiving entity, hence

$$K_S := F(K_M)[R].$$

To prevent replay attacks, the diversification data R should either be unpredictable, or at least different for each session key derivation.

The only requirement for the diversification function F is that the number of possible outputs of the function is sufficiently large and uniformly distributed to prevent an exhaustive key search on the session key.

E2. Asymmetric Mechanisms

E2.1 Digital Signature Scheme Giving Message Recovery

This section describes the digital signature scheme giving message recovery using a hash function according to ISO/IEC 9796-2. The main features of the scheme are the following.

- Adding of redundancy in the signature by applying a hash function to the data to be signed.

- Adding of a header and trailer byte in order to obtain a unique recovery procedure and to prevent certain attacks as described in the informative reference [2] in Annex G.

E2.1.1 Algorithms

The digital signature scheme uses the following two types of algorithms.

- A reversible asymmetric algorithm consisting of a signing function $Sign(S_K)[\ ]$ depending on a Private Key S_K and a recovery function $Recover(P_K)[\ ]$ depending on a Public Key P_K. Both functions map N-byte numbers onto N-byte numbers and have the property that

  $$Recover(P_K)[Sign(S_K)[X]] = X,$$

  for any N-byte number X.

- A hashing algorithm Hash[ ] that maps a message of arbitrary length onto an 20-byte hash code.

E2.1.2 Signature Generation

The computation of a signature S on a message MSG consisting of an arbitrary number L of bytes takes place in the following way.

CASE 1: *The length L of the message to be signed is at most N – 22 bytes*

1. Compute the 20-byte hash value H := Hash[MSG] of the message MSG.

2. Define the (N – 21 – L)-byte block B as follows:

   B := ‘4A’ if L = N – 22,

   B := (‘4B’ || ‘BB’ || ‘BB’ || . . . || ‘BB’ || ‘BA’) if L < N – 22.

3. Define the byte E := ‘BC’.

4. Define the N-byte block X as the concatenation of the blocks B, MSG, H and E, hence

$$X := (B \,||\, MSG \,||\, H \,||\, E).$$

5. The digital signature is then defined as the N-byte number

$$S := \mathrm{Sign}(S_K)[X].$$

CASE 2: *The length L of the message to be signed is larger than N – 22 bytes*

1. Compute the 20-byte hash value H := Hash[MSG] of the message M.

2. Split MSG into two parts MSG =(MSG_1 || MSG_2), where MSG_1 consists of the N – 22 leftmost (most significant bytes) of MSG and MSG_2 of the remaining (least significant) L – N + 22 bytes of MSG.

3. Define the byte B := ‘6A’.

4. Define the byte E := ‘BC’.

5. Define the N-byte block X as the concatenation of the blocks B, MSG_1, H and E, hence

$$X := (B \,||\, MSG_1 \,||\, H \,||\, E).$$

6. The digital signature S is then defined as the N-byte number

$$S := \mathrm{Sign}(S_K)[X].$$

E2.1.3 Signature Verification

Signature verification takes place in the following way.

CASE 1: *The length L of the message signed is at most N – 22 bytes*

1. Check whether the digital signature S consists of N bytes.

2. Retrieve the N-byte number X from the digital signature S:

$$X = \mathrm{Recover}(P_K)[S].$$

3. Partition X as X = (B || MSG || H || E), where

   – B = ‘4A’ or B is a leading byte-string of the form (‘4B’ || ‘BB’ || ‘BB’ || . . . || ‘BB’ || ‘BA’). If none of these cases occur, the signature is rejected.

   – H is 20 bytes long.

- E is one byte long.
- MSG consists of the remaining bytes.

4. Check whether the byte E is equal to ‘BC’.

5. Check whether H = Hash[MSG].

If and only if these checks are correct is the message accepted as genuine.

CASE 2: *The length L of the message signed is larger than N – 22 bytes*

1. Check whether the digital signature S consists of N bytes.

2. Retrieve the N-byte number X from the digital signature S:

$$X = \text{Recover}(P_K)[S].$$

3. Partition X as X= (B || MSG_1 || H || E), where

   - B is one byte long.
   - H is 20 bytes long.
   - E is one byte long.
   - MSG_1 consists of the remaining N – 22 bytes.

4. Check whether the byte B is equal to ‘6A’.

5. Check whether the byte E is equal to ‘BC’.

6. Compute MSG = (MSG_1 || MSG_2) and check whether H = Hash[MSG].

If and only if these checks are correct is the message accepted as genuine.

Annex F - Approved Cryptographic Algorithms

F1. Symmetric Algorithms

F1.1 Data Encryption Standard (DES)

This 64-bit block cipher is the approved algorithm for secure messaging and is standardised in ISO 8731-1, ISO 8372, and ISO/IEC 10116. More precisely, both the single DES encipherment and the triple DES encipherment versions described below are approved to be used in the encipherment and MAC mechanisms described in Annex E.

Triple DES encipherment involves enciphering an 8-byte plaintext block in an 8-byte ciphertext block with a double-length (16-byte) secret key $K = (K_L \,||\, K_R)$ as follows:

$$Y = DES(K_L)[DES^{-1}(K_R)[DES(K_L)[X]]].$$

Decipherment takes place as follows:

$$X = DES^{-1}(K_L)[DES(K_R)[DES^{-1}(K_L)[Y]]].$$

F2. Asymmetric Algorithms

F2.1 RSA Algorithm

This reversible algorithm is the approved algorithm for static and dynamic data authentication and is used to this aim in the digital signature scheme described in Annex E (see ISO/IEC 9796-2 and the informative reference [4] in Annex G). Both odd and even public key exponents are allowed.

The algorithm produces a digital signature whose length equals the size of the modulus used. The mandatory upper bounds for the size of the modulus are specified in Table F-1.

Description	Max. Length
Certification Authority Public Key Modulus	248 bytes
Issuer Public Key Modulus	247 bytes
ICC Public Key Modulus	128 bytes

Table F-1 - Mandatory Upper Bound for the Size in Bytes of the Moduli

Furthermore, the length N_{CA} of the Certification Authority Public Key Modulus, the length N_I of the Issuer Public Key Modulus, and the length N_{IC} of the ICC Public Key Modulus shall satisfy $N_{IC} < N_I < N_{CA}$.

In the choice of the lengths of the public key moduli, one should take into account the life-time of the keys compared to the expected progress in factoring during that life-time. The ranges (upper and lower bounds) for the key lengths mandated by each of the payment systems are specified in their corresponding proprietary specifications.

The length of the Issuer Public Key Exponent and the ICC Public Key Exponent is determined by the issuer. The Certification Authority, Issuer and ICC Public Key Exponents shall be equal to 2, 3, or $2^{16} + 1$

The Public Key Algorithm Indicator for this digital signature algorithm shall be coded as hexadecimal '01'.

The keys and signing and recovery functions for the RSA algorithm are specified below. The cases for an odd and even public key exponent are considered separately. Furthermore, minimum requirements for the key generation process are specified.

F2.1.1 Odd Public Key Exponent

F2.1.1.1 Keys

The private key S_K of the RSA digital signature scheme with an odd public key exponent e consists of two prime numbers p and q such that p – 1 and q – 1 are co-prime to e and a private exponent d such that

$$ed \equiv 1 \bmod (p-1)(q-1).$$

The corresponding public key P_K consists of the public key modulus n = pq and the public key exponent e.

F2.1.1.2 Signing Function

The signing function for RSA with an odd public key exponent is defined as

$$S = \mathrm{Sign}(S_K)[X] := X^d \bmod n,\ 0 < X < n,$$

where X is the data to be signed and S the corresponding digital signature.

F2.1.1.3 Recovery Function

The recovery function for RSA with an odd public key exponent is equal to

$$X = \mathrm{Recover}(P_K)[S] := S^e \bmod n.$$

F2.1.2 Even Public Key Exponent

F2.1.2.1 Keys

The private key S_K of the RSA digital signature scheme with an even public key exponent consists of two prime numbers p and q such that $p \equiv 3 \bmod 8$ and $q \equiv 7 \bmod 8$ and a private exponent d such that

$$ed \equiv 1 \bmod \tfrac{1}{2}\,\mathrm{lcm}(p - 1, q - 1).$$

Note that the special case e = 2 is better known as the Rabin variant of RSA (see reference [3] in Annex G). In that case, the private exponent is equal to

$$d = (pq - p - q + 5)/8.$$

The corresponding public key P_K consists of the public key modulus n = pq and the public key exponent e.

F2.1.2.2 Signing Function

The signing function for RSA with an even public key exponent is defined as

$$S = \mathrm{Sign}(S_K)[X] := X^d \bmod n, \text{ if the Jacobi symbol } (X \mid n) = 1,$$

$$S = \mathrm{Sign}(S_K)[X] := (X/2)^d \bmod n, \text{ if the Jacobi symbol } (X \mid n) = -1,$$

where X is the data to be signed and S the corresponding digital signature.

The data block X is a nonnegative integer less than n with the additional property that its least significant byte is equal to ‘BC’. This is to ensure that the recovery process is unique. Note that this is always the case when RSA is used in the digital signature scheme described in Annex E2.1.

F2.1.2.3 Recovery Function

The recovery function for RSA with an even public key exponent is as follows.

1. Compute $Y := S^e \bmod n$.

2. The recovered data is obtained by one of the following cases.

   (a) If the least significant byte of Y is equal to ‘BC’, X := Y.

   (b) If the least significant byte of 2Y is equal ‘BC’, X := 2Y.

   (c) If the least significant byte of n – Y is equal to ‘BC’, X := n – Y.

   (d) If the least significant byte of 2(n – Y) is equal to ‘BC’, X := 2(n – Y).

If none of these cases occur, the signature shall be rejected.

F2.1.3 Key Generation

Payment systems and issuers shall be responsible for the security of their respective RSA public/private key generation processes. Examples of secure key generation methods can be found in reference [1] in Annex G.

F3. Hashing Algorithms

F3.1 Secure Hash Algorithm (SHA-1)

This algorithm is standardised as FIPS 180-1.[18] SHA-1 takes as input messages of arbitrary length and produces a 20-byte hash value.

The Hash Algorithm Indicator for this hashing algorithm shall be coded as hexadecimal '01'.

[18] SHA-1 is also standardised in ISO/IEC 10118-3.

Annex G - Informative References

1. A. Bosselaers and B. Preneel (eds.), *Integrity Primitives for Secure Information Systems*, Final Report of the RACE Integrity Primitives Evaluation (RIPE, RACE R1040), LNCS 1007, Springer-Verlag, 1995.

2. L. Guillou, J.-J. Quisquater, M. Walker, P. Landrock, and C. Shaer, 'Precautions taken against various potential attacks in ISO/IEC 9796,' *Advances in Cryptology – Proceedings of Eurocrypt'90*, LNCS 473, I. B. Dåmgard Ed., Springer-Verlag, 1991, pp. 465-473.

3. M. O. Rabin, *Digitalized Signatures and Public-Key Functions as Intractable as Factorization*, Massachusetts Institute of Technology Technical Report MIT/LCS/TR-212, 1979.

4. R. L. Rivest, A. Shamir, and L. Adleman, 'A method for obtaining digital signatures and public key cryptosystems,' *Communications of the ACM*, vol. 21, 1978, pp. 120-126.

5. G. J. Simmons Ed., *Contemporary Cryptology: The Science of Information Integrity*, IEEE Press, Piscataway, N.J., 1992.