MasterCard International

PayPass – M/Chip

Technical Specifications

Version 1.3 - September 2005

Media

This document is available in both electronic and printed format.

MasterCard International - CMCOE
Chaussée de Tervuren, 198A
B-1410 Waterloo
Belgium

Fax: +32 2 352 5353

E-mail: specifications@paypass.com

Using this Manual

This chapter contains information that helps you understand and use this document.

Scope

MasterCard *PayPass*™ technology enables fast, easy and globally accepted payments through the use of contactless chip technology on the traditional MasterCard card platform. *PayPass* – M/Chip is designed specifically for authorization networks that presently support chip card authorizations for credit or debit applications.

This document provides the specifications necessary to achieve interoperability between *PayPass* cards and *PayPass* terminals. The application is primarily intended to carry the Maestro or MasterCard brands. It contains:

- The transaction flow (the sequence of events and the commands and responses interchanged between the card and terminal).
- Definition of commands and data elements as they apply to the exchange of information between the card and terminal.
- The implementation of the *PayPass* – M/Chip application on a dual interface card (contact and contactless).

Audience

This document is intended for use by vendors that want to implement the MasterCard *PayPass* – M/Chip application on a card or acceptance device.

This document is also intended for type approval services, which would test the actual implementations against this specification.

It is assumed that the audience already has an understanding of chip card technology in general and of M/Chip 4 and ISO/IEC 14443 in particular.

Related Publications

The following publications contain information directly related to the contents of this manual.

[PAYPASS MAGSTRIPE]	*PayPass – Mag Stripe Technical Specifications, Version 3.1 – November 2003.*
[PAYPASS ISO/IEC 14443]	*PayPass – ISO/IEC 14443 Implementation Specification, Version 1.0 – June 2004.*
[M/CHIP4]	*M/Chip 4 Card Application Specifications for Credit and Debit, Version 1.0 – October 2002.*
[M/CHIP4 CPS]	*M/Chip 4 Common Personalization Specifications, August 2003.*

Reference Materials

The following reference materials may be of use to the reader of this manual.

[ISO/IEC 8825:1990]	*Information technology – Open systems interconnection – Specification of basic encoding rules for abstract syntax notation one (ASN.1).*
[ISO/IEC 7811/2]	*Identification cards – Recording technique – Part 2: Magnetic stripe*
[ISO/IEC 7813:1995]	*Identification cards – Financial transaction cards.*
[ISO/IEC 7816-4:1995]	*Information technology – Identification cards – Integrated circuit(s) cards with contacts - Part 4: Interindustry commands for interchange.*
[ISO/IEC 7816-5:1993]	*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: Interindustry data elements.*
[EMV BOOK 1]	*Integrated Circuit Card Specification for Payment Systems: Application Independent ICC to Terminal Interface Requirements. Version 4.1, May 2004.*
[EMV BOOK 2]	*Integrated Circuit Card Specification for Payment Systems: Security and Key Management. Version 4.1, May 2004.*
[EMV BOOK 3]	*Integrated Circuit Card Specification for Payment Systems: Application Specification. Version 4.1, May 2004.*
[EMV BOOK 4]	*Integrated Circuit Card Specification for Payment Systems: Cardholder, Attendant and Acquirer Interface Requirements. Version 4.1, May 2004.*
[EMV CPS]	*EMV Card Personalization Specification, Version 1.0, June 2003.*

Abbreviations

The following abbreviations are used in this specification:

Abbreviation	Description
AAC	Application Authentication Cryptogram
AC	Application Cryptogram
ADF	Application Definition File
AFL	Application File Locator
AID	Application Identifier
AIP	Application Interchange Profile
ASI	Application Selection Indicator
an	Alphanumeric
ans	Alphanumeric Special
APDU	Application Protocol Data Unit
ARQC	Authorization Request Cryptogram
ATC	Application Transaction Counter
b	Binary
BCD	Binary Coded Decimal
C-APDU	Command APDU
CDOL	Card Risk Management Data Object List
CLA	Class byte of command message
cn	Compressed Numeric
CVC	Card Verification Code
CVM	Cardholder Verification Method
DES	Data Encryption Standard
DDA	Dynamic Data Authentication
DGI	Data Grouping Identifier
EMV	Europay MasterCard Visa
FCI	File Control Information
hex.	Hexadecimal
ICC	Integrated Circuit Card
INS	Instruction byte of command message
ISO	International Organization for Standardization
MAC	Message Authentication Code
n	Numeric
$NATC_{TRACK1}$	Track 1 Number of ATC Digits
$NATC_{TRACK2}$	Track 2 Number of ATC Digits
N_{CA}	Length of the Certification Authority Public Key Modulus
N_I	Length of the Issuer Public Key Modulus

Abbreviation	Description
N_{IC}	Length of the ICC Public Key Modulus
PAN	Primary Account Number
$PCVC3_{TRACK1}$	Track 1 Bit Map for CVC3
$PCVC3_{TRACK2}$	Track 2 Bit Map for CVC3
PDOL	Processing Options Data Object List
PIN	Personal Identification Number
PPSE	*PayPass* Payment System Environment
PSE	Payment System Environment
$PUNATC_{TRACK1}$	Track 1 Bit Map for UN and ATC
$PUNATC_{TRACK2}$	Track 2 Bit Map for UN and ATC
P1	Parameter 1
P2	Parameter 2
R-APDU	Response APDU
RFU	Reserved for Future Use
SDA	Static Data Authentication
SDAD	Signed Dynamic Application Data
SFI	Short File Identifier
SW1	Status Byte One
SW2	Status Byte Two
TC	Transaction Certificate
TDOL	Transaction Certificate Data Object List
TLV	Tag Length Value
TVR	Terminal Verification Results
UDOL	Unpredictable Number Data Object List
UN	Unpredictable Number

Notational Conventions

The following notations apply:

Notation	Description
‘0’ to ‘9’ and ‘A’ to ‘F’	Hexadecimal notation. Values expressed in hexadecimal form are enclosed in single quotes (i.e. ‘_’).
1001b	Binary notation. Values expressed in binary form are followed by a lower case “b”.
“abcd”	an or ans string
digit	Any of the ten Arabic numerals from 0 to 9.
[…]	Optional part
xx	Any value

Notation	Description
A := B	A is assigned the value of B
C := (A || B)	The concatenation of an n-bit number A and an m bit number B, which is defined as $C = 2^m A + B$.
Y := ALG(K)[X]	Encipherment of a 64-bit data block X with a 64-bit block cipher using a secret key K.
X & Y	The bit-wise and of the data block X and Y.
Application File Locator	Data elements used for this specification are written in *italics* to distinguish them from the text.
GENERATE AC	Command APDUs used for this specification are written in SMALL CAPITALS to distinguish them from the text.

The following table lists symbols that are often used throughout the document:

Symbol	Meaning
k_{TRACK1}	Number of non-zero bits in the *Track 1 Bit Map for UN and ATC* (*PUNATC*$_{TRACK1}$)
k_{TRACK2}	Number of non-zero bits in the *Track 2 Bit Map for UN and ATC* (*PUNATC*$_{TRACK2}$)
t_{TRACK1}	The symbol t_{TRACK1} represents the value of *NATC*$_{TRACK1}$ and indicates the number of digits of the *ATC* to be included in the discretionary data field of the *Track 1 Data*.
t_{TRACK2}	The symbol t_{TRACK2} represents the value of *NATC*$_{TRACK2}$ and indicates the number of digits of the *ATC* to be included in the discretionary data field of the *Track 2 Data*.
n_{UN}	The symbol n_{UN} represents the number of positions available in the discretionary data fields of the *Track 1 Data* and *Track 2 Data* for transporting *UN* to the issuer. The value of n_{UN} must be the same for *Track 1 Data* and *Track 2 Data*. Therefore the following holds: $n_{UN} = k_{TRACK1}-t_{TRACK1} = k_{TRACK2}-t_{TRACK2}$.
m_{TRACK1}	The symbol m_{TRACK1} indicates the number of characters present in the discretionary data field of the *Track 1 Data*.
m_{TRACK2}	The symbol m_{TRACK2} indicates the number of digits present in the discretionary data field of the *Track 2 Data*.
q_{TRACK1}	Number of non-zero bits in the *Track 1 Bit Map for CVC3* (*PCVC3*$_{TRACK1}$). The symbol q_{TRACK1} represents the number of CVC3 digits included in the discretionary data field of the *Track 1 Data.*
q_{TRACK2}	Number of non-zero bits in the *Track 2 Bit Map for CVC3* (*PCVC3*$_{TRACK2}$). The symbol q_{TRACK2} represents the number of CVC3 digits included in the discretionary data field of the *Track 2 Data*.

Bit Map

The *PayPass* – M/Chip application uses a bit map to indicate positions in the discretionary data field.

Figure 1 indicates the numbering of the different positions in the discretionary data. The number of digits present in discretionary data is indicated by m.

Figure 1—Numbering of Discretionary Data

Discretionary Data									
p_m	p_{m-1}	p_{m-2}	p_{m-3}	…	p_5	p_4	p_3	p_2	p_1

Each bit in the bit map refers to a position in the discretionary data. The least significant bit of the bit map, i.e. the rightmost bit b_1, refers to position p_1; as indicated in Figure 2. The number of bits in the bit map is always a multiple of 8 and equal to r=(((m-1)/8)+1)*8. For *Track 2 Data* m_{TRACK2} is maximum 13 digits, resulting in a bit map of 16 bits or 2 bytes. For *Track 1 Data* the maximum value of m_{TRACK1} is 48 resulting in a bitmap of length 6 bytes or 48 bits.

Figure 2—Relation between Discretionary Data and Bit Map

					Discretionary Data									
					p_m	p_{m-1}	p_{m-2}	p_{m-3}	…	p_5	p_4	p_3	p_2	p_1
b_r	b_{r-1}	b_{r-2}	…	b_{m+1}	b_m	b_{m-1}	b_{m-2}	b_{m-3}	…	b_5	b_4	b_3	b_2	b_1
Bit Map														

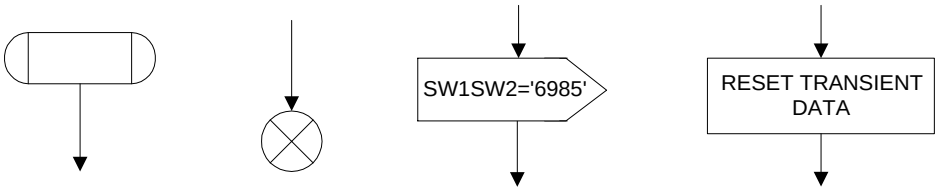
An example is given in Figure 3, for m_{TRACK2}=13, t_{TRACK2}=2 and $PUNATC_{TRACK2}$ = '031A', referring to position $p_{10}p_9p_5p_4p_2$. Based on this, k_{TRACK2} equals 5 and n_{UN} equals 3.

Figure 3—Example $PUNATC_{TRACK2}$ = '031A'

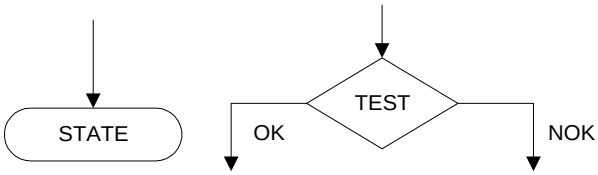
					Discretionary Data									
					p_{13}	p_{12}	p_{11}	**p_{10}**	**p_9**	p_8	p_7	p_6	**p_5**	**p_4**	p_3	**p_2**	p_1
0	0	0	0	0	0	**1**	**1**	0	0	0	**1**	**1**	0	**1**	0
b_{16}	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1
'0'				'3'				'1'				'A'			
Bit Map = '031A'															

Transition Flow Diagrams

The following symbols are used in the transition flow diagrams:



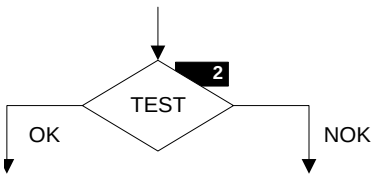
procedure start procedure return output perform task



destination state decision

In most cases a textual description accompanies the transition flow diagram. In this case the symbols in the transition flow diagram are identified with a symbol number. When a paragraph in the textual description starts with '**Symbol n'**, then it corresponds to the symbol bearing the same number in the transition flow diagram. The following example illustrates how it works.

- The decision symbol is used in a flow diagram, identified with number 2.



- An explanation of the check done in symbol 2 is given:

  Symbol 2
  An explanation of how the application checks that the condition is satisfied.

Requirement Numbering

Requirements in this manual are uniquely numbered with the number appearing next to each requirement: For example:

4.4.1.2 If the *PDOL* is not present, then the terminal shall use a command data field of '83 00'.

Document Overview

This document is organized as follows:

Section	Description
Using this Manual	A description of the manual's purpose and its contents
Part I – Introduction	Provides a high-level summary of the *PayPass* – M/Chip application.
Part II – Interface Specification	Describes the functions necessary to ensure that *PayPass* – M/Chip cards conforming to this specification can perform a set of core functions in all terminals that conform to this specification. Application functions unique to individual implementations (and functions not performed in interchange) are not described here. This part includes: • The application selection mechanism by means of the PPSE. • The transaction flow (the sequence of events and the commands and responses interchanged between the card and the terminal). • The definition of commands and data elements as they apply to the exchange of information between the card and the terminal. • The card and terminal interoperability requirements. (This part does not address clearing and settlement issues, or transactions where the *PayPass* – M/Chip card is not present.)
Part III – Card Specification	Includes the behavioral specification of the PPSE and the *PayPass* – M/Chip 4 dual interface card application proposed by MasterCard and conforming to the requirements listed in Part II. The *PayPass* – M/Chip 4 application is an extension of the M/Chip 4 contact-only application specified in *[M/CHIP4]*. The card implementation specification provides a definition of the behavior of the *PayPass* – M/Chip 4 application during the operational phase of the card life cycle. The principles and concepts proposed in this part do not have to be followed in the actual implementation. However, the implementation must behave in exactly the same way as specified in this specification.
Annex A – MAC Algorithm	Describes the MAC algorithm used by the *PayPass* – M/Chip 4 application.
Annex B – *PayPass* Data Groupings	Defines the structure of the data groupings that must be used to personalize the *PayPass* – M/Chip application according to *[M/CHIP4 CPS]*.

Document Word Usage

The following words are used often in this manual and have a specific meaning:

- shall
  Defines a product or system capability which is mandatory.
- should
  Defines a product or system capability which is recommended.
- may
  Defines a product or system capability which is optional.

PART I – Introduction

This part includes an executive summary of the PayPass – M/Chip application.

1 MasterCard Proximity Payment

MasterCard International has initiated the development of a program intended to allow consumers to make MasterCard payment transactions at point of sale using contactless technology. The generic term "contactless technology" is used when the point of interaction is between 1mm and 10 meters. Although the proximity payment program covers multiple technologies and ranges, this document deals only with the technical specifications of the MasterCard *PayPass*™ product built with a contactless chip conforming to *[PAYPASS ISO/IEC 14443]* with a range from 1mm to 10cm.

 **Note** **In this document, the term "*PayPass* interface" will be used to mean "*PayPass* contactless interface".**

2 M/Chip Profile and Mag Stripe Profile

Within the MasterCard *PayPass* transactions we distinguish two different profiles: M/Chip and Mag Stripe.

The *PayPass* – Mag Stripe profile was developed to allow contactless payments using authorization networks that presently support magnetic-stripe authorization for credit or debit applications. A *PayPass* – Mag Stripe card stores track 1 and track 2 data with a discretionary data field that contains a dynamic CVC (Card Verification Code). The dynamic CVC is generated by the *PayPass* – Mag Stripe card using a secret key, the *Application Transaction Counter* of the *PayPass* – Mag Stripe card and an unpredictable number generated by the terminal. The *PayPass* – Mag Stripe card with dynamic CVC provides better security than magnetic-stripe technology because the dynamic CVC is used as authentication code by the issuer to authenticate the *PayPass* – Mag Stripe card during the online authorization processing.

The *PayPass* – M/Chip profile was developed to allow contactless payments in a market that is oriented towards offline acceptance. To manage the offline risk the terminal must be capable of performing terminal risk management and offline authentication of the card (e.g. static data authentication). The *PayPass* – M/Chip card must be capable of performing its own card risk management and of accepting or declining the transaction offline.

Interoperability between the *PayPass* – M/Chip profile and the *PayPass* – Mag Stripe profile is achieved by the requirement that the *PayPass* – M/Chip card and terminal also support the *PayPass* – Mag Stripe profile. This means that a *PayPass* – Mag Stripe card will be accepted on a *PayPass* – M/Chip terminal and that a *PayPass* – M/Chip card will be accepted on a *PayPass* – Mag Stripe terminal.

This document deals with the technical specifications of the *PayPass* – M/Chip card and terminal.

Note **In this document, the terms “card” and “terminal” will be used to mean “*PayPass* – M/Chip card” and “*PayPass* – M/Chip terminal”.**

3 PayPass – M/Chip

3.1 Interface Specification

The *PayPass* – M/Chip interface specification is based on the EMV specifications with the following amendments:

- The application selection mechanism has been adapted to allow for an efficient application selection when multiple applications are supported in the terminal. This *PayPass* specific application selection mechanism makes use of the *PayPass* Payment Systems Environment (PPSE). All *PayPass* cards must support this mechanism.
- Data elements located in the files with SFI 1 to 10 are organized in a pre-defined file structure to allow for efficient data capture by the terminal.
- Offline static data authentication may be performed after the card has been removed from the electromagnetic field. In this case the outcome of the static data authentication process is not taken into account by the terminal action analysis and card action analysis functions. The terminal will however use the outcome of the static data authentication process to accept or decline the transaction offline if the card accepted the transaction offline.
- The *PayPass* card must not perform offline dynamic data authentication. A RSA capable *PayPass* card must be authenticated by the terminal with the combined DDA/AC generation mechanism. This allows the terminal to verify the *Signed Dynamic Application Data* after the card has been removed from the field.
- The *PayPass* card and terminal do not support offline PIN verification. This means the card does not support offline plaintext PIN nor offline enciphered PIN verification.
- Script processing and issuer authentication do not have to be performed as this supposes a response from the issuer while the card is still in the field. An online capable terminal should not issue a 2nd GENERATE AC command after the card has generated an ARQC in response to the 1st GENERATE AC. Instead the terminal should use the *Authorization Response Code* included in the authorization response message of the issuer to accept or decline the transaction online. If the transaction is accepted, then the terminal uses the ARQC as transaction certificate for the clearing process.
  The *PayPass* card must not require a 2nd GENERATE AC after the generation of an ARQC. It must however support the 2nd GENERATE AC command for the exceptional case in which the terminal is able to hold the card within the field until the response from the issuer is available.
  A terminal that performs a 2nd GENERATE AC must do it according to the procedures specified in *[EMV BOOK 3]*.

- Both a *PayPass* – M/Chip card and a *PayPass* – M/Chip terminal must support the COMPUTE CRYPTOGRAPHIC CHECKSUM command to be able to process *PayPass* – Mag Stripe transactions. The COMPUTE CRYPTOGRAPHIC CHECKSUM command provides the terminal with the dynamic track data that must be used for the authorization and clearing in the case of a *PayPass* – Mag Stripe transaction.

3.2 Transaction Flow

The *PayPass* – M/Chip transaction flow follows to a large extent the transaction flow of the traditional contact EMV transaction. The flowchart in Figure 4 illustrates the interaction between card and terminal for a *PayPass* – M/Chip transaction.

1. The terminal begins by selecting the *PayPass* Payment Systems Environment (PPSE) using the SELECT command.
2. The card responds with the *File Control Information (FCI)* including all the *AID*s supported by the card with their priority indicator.
3. The terminal selects the *AID* with the highest priority that is supported by both card and terminal and issues the SELECT command with this *AID*.
4. The card responds with the *File Control Information (FCI)*. The *FCI* may contain the *Processing Options Data Object List (PDOL)*. The *PDOL* is a list of tags and lengths of terminal resident data elements needed by the card in the GET PROCESSING OPTIONS command.
5. The terminal issues the GET PROCESSING OPTIONS command. If there is no *PDOL* in the card, then the terminal uses the command data field '8300'. Otherwise the command data field contains a data object with tag '83' and a value field comprising the concatenated list of data elements resulting from processing the *PDOL*.
6. The card returns the *Application Interchange Profile (AIP)* and the *Application File Locator (AFL)*.

7. – 8. The terminal issues the READ RECORD command to retrieve the generic card application data elements (e.g. *PAN*, *Application Expiry Date*, etc…) located in the first record of the file with SFI 2. The response message of the card contains the record read including all the generic card application data elements.

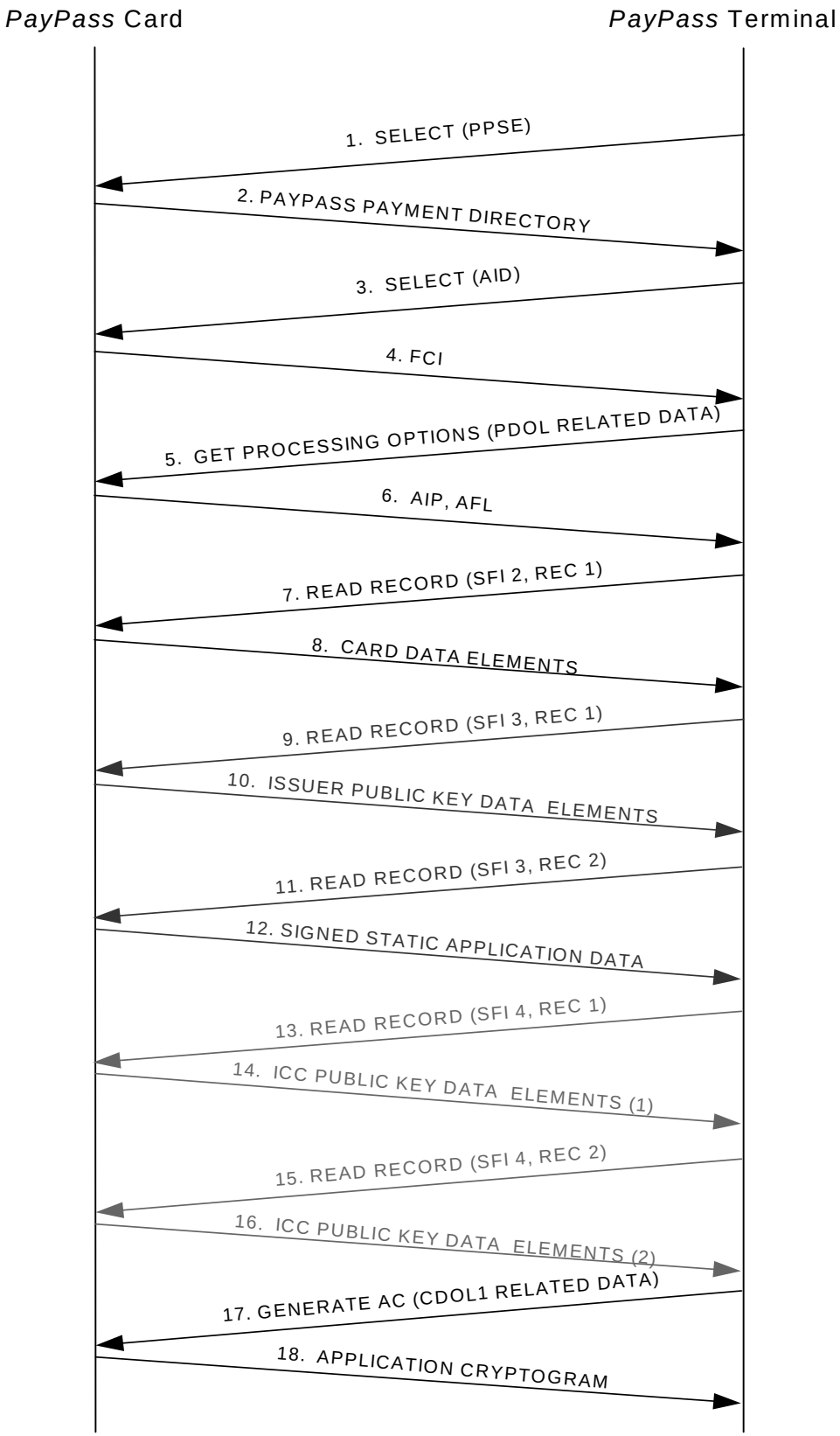
9. – 10. If the card supports offline data authentication (static data authentication or combined DDA/AC generation), then the terminal issues the READ RECORD command to retrieve the card data elements necessary to recover the *Issuer Public Key*. These data elements are located in the first record of the file with SFI 3.

11. – 12. If the card supports static data authentication and the card does not support combined DDA/AC generation, then the terminal issues the READ RECORD command to retrieve the *Signed Static Application Data*. This data element is located in the second record of the file with SFI 3.

13. – 14 – 15. – 16. If the card supports combined DDA/AC generation, then the terminal issues two READ RECORD commands to retrieve the card data elements necessary to recover the *ICC Public Key*. These data elements are located in the first and second record of the file with SFI 4.

17. After terminal risk management (application expiry date checking, terminal floor limit checking, exception file checking, etc.) has been completed, the terminal makes a preliminary decision to decline the transaction offline, complete it online or accept it offline. This decision is based upon the *Terminal Verification Results (TVR)*, the issuer action preferences and acquirer action preferences according to the method described in *[EMV BOOK 3]*. If the decision is to accept the transaction offline, then the terminal issues the GENERATE AC command requesting a TC. If the decision is to decline the transaction offline, then the terminal issues the GENERATE AC command requesting an AAC. In the case the terminal wants to complete the transaction online, then the terminal issues the GENERATE AC command requesting an ARQC.
18. Based upon the *CDOL1* related data included in the data field of the GENERATE AC command, the card may perform its own card risk management. As a result of the card risk management process, the card may decide to complete a transaction online, accept offline or decline the transaction. In all three cases the card will generate an *Application Cryptogram*. If the card responds with a TC or an AAC, then the terminal completes the transaction offline. If the card responds with an ARQC, then the terminal attempts to go online, sending an authorization request message to the issuer. Included in the authorization request message is the ARQC for online card authentication.

Figure 4—*PayPass* Transaction Flow



4 M/Chip 4

PART III of this document contains the behavioral specification of a *PayPass* – M/Chip card application based on *[M/CHIP4]* (the M/Chip Select 4 and M/Chip Lite 4 application specification). The card platform carrying the *PayPass* – M/Chip application must have a dual interface (i.e. a card with an EMV contact interface and a *PayPass* interface) and must be capable of accessing the M/Chip 4 application in both contact and contactless mode.

It is assumed that a number of transactions made with the *PayPass* – M/Chip 4 application will be performed via the contact interface and online to the issuer. The online processing allows the issuer to send *Issuer Authentication Data* to the card together with script commands to reset the offline risk counters which are common for the contact and *PayPass* interface.

The *PayPass* – M/Chip 4 application shares the offline risk counters between the contact and contactless interface. The offline counters will only be updated if the transaction is accepted offline. As the terminal does not have to generate a 2nd GENERATE AC command, the counters remain unchanged if the card generated an ARQC in response to the 1st GENERATE AC command.

The *PayPass* – M/Chip 4 application supports six new data objects: the *Application Interchange Profile (PayPass)*, the *Application File Locator (PayPass)*, the *Application Control (PayPass)* and the *Card Issuer Action Codes (PayPass)*. These data objects replace the existing *Application Interchange Profile*, *Application File Locator*, *Application Control* and *Card Issuer Action Codes* of the M/Chip 4 contact-only application in the case the *PayPass* interface is used. All other data elements are shared between the contact and *PayPass* interface.

The *Application Interchange Profile (PayPass)* contains a *PayPass* specific bit indicating the profile of the card (M/Chip or Mag Stripe).

The two instances of the *Application File Locator* allow the use of a different set of files and records depending on the active interface (*PayPass* or contact).

The *Application Control (PayPass)* is a *PayPass* – M/Chip 4 proprietary data element to activate or de-activate certain functions in the application when the *PayPass* interface is used. The *Application Control (PayPass)* data element must always be personalized in such a way that the VERIFY command is not activated.

The *Card Issuer Action Codes (PayPass)* are represented by three *PayPass* – M/Chip 4 proprietary data elements: *Card Issuer Action Code (PayPass) – Default*, *Card Issuer Action Code (PayPass) – Online* and *Card Issuer Action Code (PayPass) – Decline*. They are compared to the decisional part of the *Card Verification Results* to decide which cryptogram to include in the response to the GENERATE AC (i.e. whether to decline or accept a transaction, or whether to go online to the issuer).

PART II – Interface Specification

PART II describes the interface between a PayPass card and terminal, in terms of the:

- *Application selection mechanism*
- *Definition of commands as they apply to the exchange of information between the card and the terminal*
- *Transaction flow (the sequence of events and the commands and responses interchanged between the card and the terminal)*
- *Interoperability requirements between the card and terminal*
- *Data objects passed between the card and terminal*

---

1 Application Selection

This chapter describes the application selection process from the standpoint of both the *PayPass* card and the terminal. The application selection mechanism minimizes the number of commands between card and terminal. Only two SELECT commands are necessary. The process is described in two steps similar to the EMV application selection mechanism:

1. Create a list of applications that are supported by both the card and the terminal. This list is referred to using the name 'candidate list'.
2. From the candidate list, select the application to be run.

Note **A terminal supporting only one application (= one *AID*), should immediately try to select the *ADF Name* (= *AID*) of the corresponding application in the card and skip the application selection process. In this case only one SELECT command is required.**

1.1 SELECT PPSE

This section describes the structure of the SELECT APDU command-response pair necessary to the functioning of the application selection. In this context the SELECT command is used to select the *PayPass* Payment System Environment (PPSE) directory. The response from the card consists of returning the *FCI* containing the list of *PayPass* applications (*AID*s) supported by the card.

1.1.1 Command Message

The SELECT command message is coded according to Table 1.

Table 1—SELECT PPSE Command Message

Code	Value
CLA	'00'
INS	'A4'
P1	'04'
P2	'00'
Lc	'0E'
Data	'32 50 41 59 2E 53 59 53 2E 44 44 46 30 31'
Le	'00'

The data field of the command message contains the PPSE directory name ("2PAY.SYS.DDF01").

1.1.2 Data Field Returned in the Response Message

The data field of the response message contains the *FCI* specific to the selected PPSE. Additional tags returned in the *FCI* that are not described in this specification must be ignored by the terminal.

Table 2 defines the *FCI* returned by a successful selection of the PPSE directory.

Table 2—SELECT Response Message Data Field (FCI) of the PPSE

Tag	Value			Presence
'6F'	*FCI Template*			M
	'84'	*DF Name*		M
	'A5'	*FCI Proprietary Template*		M
		'BF0C'	*FCI Issuer Discretionary Data*	M

The *FCI Issuer Discretionary Data* is a constructed data object of which the value field is comprised of one or more *Application Templates* (tag '61') as described in Table 3.

Table 3—FCI Issuer Discretionary Data

'BF0C'	Length	'61'	Length of directory entry 1	Directory entry 1	…	'61'	Length of directory entry n	Directory entry n

Each directory entry is the value field of an *Application Template* and contains the information according to Table 4 and Table 5.

Table 4—Directory Entry Format

Tag	Value	Presence
'4F'	*ADF Name* (*AID*)	M
'87'	*Application Priority Indicator* (see Table 5).	M
'50'	*Application Label*	O

Table 5—Application Priority Indicator Format

b8	b7-b5	b4-b1	Definition
0			Application may be selected without confirmation of cardholder
	xxx		RFU
		0000	No priority assigned
		xxxx	Order in which the application is to be listed or selected, ranging from 1-15, with 1 being the highest priority.

1.1.3 Status Bytes for SELECT Command

The status bytes for the SELECT command are listed in Table 6.

Table 6—Status Bytes for SELECT PPSE Command

SW1	SW2	Meaning
'62'	'83'	Selected file invalidated
'67'	'00'	Wrong length
'6A'	'81'	Function not supported
'6A'	'82'	File not found
'6A'	'86'	Incorrect parameters P1-P2
'90'	'00'	Normal processing

1.2 Building the Candidate List

The steps the terminal takes to build the candidate list are as follows:

1.2.1.1 The terminal shall select the *PayPass* Payment Systems Environment (PPSE) using the SELECT command as described in Section 1.1.1 of Part II.

1.2.1.2 If the card is blocked or the SELECT command is not supported (SW1-SW2='6A81'), then the terminal shall terminate the transaction.

1.2.1.3 If the card returns SW1-SW2 ='9000', then the terminal shall proceed to step 1.2.1.6.

1.2.1.4 If the card returns any other value in SW1-SW2, then the terminal shall use the list of applications method described in Section 12.3.3 of *[EMV BOOK 1]* to find a match.

1.2.1.5 If any error occurs in steps 1.2.1.6 through 1.2.1.8, then the terminal shall clear the candidate list and restart the application selection process using the list of applications method as described in Section 12.3.3 of *[EMV BOOK 1]* to find the matching applications.

1.2.1.6 The terminal shall retrieve all the directory entries from the *FCI Issuer Discretionary Data* (tag 'BF0C') in the *FCI* returned by the card.

1.2.1.7 The terminal shall process each directory entry by comparing the *ADF Name* in the directory entry with the *AID*s supported by the terminal. If the *ADF Name* matches one of the applications supported by the terminal (as defined in Section 1.4), then the application joins the candidate list for final application selection.

1.2.1.8 If no directory entries that match applications supported by the terminal are found, then the terminal shall use the list of applications method as described in Section 12.3.3 of *[EMV BOOK 1]* to find a match.

1.3 Final Selection

Once the terminal determined the list of mutually supported applications, it proceeds as follows:

1.3.1.1 The terminal shall remove from the list of mutually supported applications all applications prohibiting selection without cardholder assistance (b8 = '1' in the *Application Priority Indicator* (see Table 5)).

1.3.1.2 The terminal shall order the list of mutually supported applications according to the following rules:

- The highest priority application must be ordered first.
- If all applications in the list have a priority indicator set to 0000b, then the list shall be in the order in which the applications were encountered in the card.
- If some of the applications in the list have a priority indicator set to 0000b and other applications in the list have a priority indicator different from 0000b, then those with no priority indicator shall come last and in the order in which they were encountered in the card.
- If there are duplicate priorities assigned to multiple applications, then these applications shall be included in the list in the order in which they were encountered in the card.

1.3.1.3 If the list of mutually supported applications is empty, then the terminal shall terminate the transaction.

1.3.1.4 The terminal shall pick the first application from the list of mutually supported applications. The terminal shall select this application with a SELECT command coded according to Section 2.6.2 of Part II using the *ADF Name* found in the directory entry of the application.
If the SELECT command fails (i.e. SW1-SW2 ≠ 9000), then the terminal shall remove the application from the list of mutually supported applications, and shall resume processing at step 1.3.1.3.

1.4 Matching Terminal AIDs to ICC AIDs

The terminal determines which applications in the ICC are supported by comparing the *AID*s for applications in the terminal with *AID*s for applications within the card.

In some cases, the terminal supports the card application only if the *AID* of the terminal has the same length and value as the *AID* in the card. In other cases, the terminal supports the card application if the *AID* in the card begins with the entire *AID* kept within the terminal.

The terminal keeps for every *AID* an *Application Selection Indicator* (*ASI*). The *ASI* indicates whether the *AID* in the terminal must match exactly (both in length and value) or need only partially match the associated *AID*.

The following requirements apply for the support of partial name selection:

1.4.1.1 Card support for partial name selection is not mandatory. However, if the card supports partial name selection, then it shall comply with Section 12.3.1 of *[EMV BOOK 1]*.

1.4.1.2 Terminal support for partial name selection is mandatory. For each *AID* within the list of applications supported by the terminal, the terminal shall keep an indication (*ASI*) of which matching criterion to use.

1.4.1.3 For each *PayPass* application within the list of applications supported by the terminal, the terminal shall indicate by means of the *ASI* that a full match (both length and value) is required.

2 Commands

This chapter specifies the commands supported by the *PayPass* – M/Chip application.

2.1 Introduction

The INS byte of the C-APDU is structured according to *[EMV BOOK 1]*. The coding of INS and its relationship to CLA are shown in Table 7.

Table 7—Coding of the Instruction Byte

CLA	INS	Meaning
'80'	'2A'	COMPUTE CRYPTOGRAPHIC CHECKSUM
'80'	'AE'	GENERATE AC
'80'	'A8'	GET PROCESSING OPTIONS
'00'	'B2'	READ RECORD
'00'	'A4'	SELECT

The status bytes returned by the card are coded as specified in *[EMV BOOK 3]*. In addition to the status bytes specific for every command, the card may return the status bytes shown in Table 8.

Table 8—Generic Status Bytes

SW1	SW2	Meaning
'6D'	'00'	Instruction code not supported or invalid
'6E'	'00'	Class not supported
'6F'	'00'	No precise diagnosis

2.2 COMPUTE CRYPTOGRAPHIC CHECKSUM

2.2.1 Definition and Scope

The COMPUTE CRYPTOGRAPHIC CHECKSUM initiates the computation of the dynamic CVC3 on the card. The computation is based on the *Unpredictable Number (Numeric)* (tag '9F6A') sent by the terminal, the *ATC* of the card and the relevant secret key stored in the card.

The response of the card consists of returning the $CVC3_{TRACK2}$, *the* $CVC3_{TRACK1}$ (optional) and the *ATC* to the terminal.

2.2.2 Command Message

The COMPUTE CRYPTOGRAPHIC CHECKSUM command message is coded according to Table 9.

Table 9—COMPUTE CRYPTOGRAPHIC CHECKSUM Command Message

Code	Value
CLA	'80'
INS	'2A'
P1	'8E'
P2	'80'
Lc	var.
Data	*UDOL* related data
Le	'00'

The data field of the command message is coded according to the *UDOL* following the rules as defined in Section 4.2 of PART II. If the card does not have a *UDOL*, then the terminal uses the *Default Terminal UDOL*.

2.2.3 Data Field Returned in the Response Message

The data field of the response message is a constructed data object with tag '77'. The value field may include several BER-TLV coded data objects, but must always include the $CVC3_{TRACK2}$ (tag '9F61') and the *ATC* (tag '9F36'). The value field may also include the $CVC3_{TRACK1}$ (tag '9F60').

Table 10—COMPUTE CRYPTOGRAPHIC CHECKSUM Response Message Data Field

Tag	Value		Presence
'77'	*Response Message Template*		M
	'9F61'	$CVC3_{TRACK2}$	M
	'9F36'	*ATC*	M
	'9F60'	$CVC3_{TRACK1}$	O

Additional data objects returned in the data field that are not described in this specification must be ignored by the terminal.

2.2.4 Status Bytes for COMPUTE CRYPTOGRAPHIC CHECKSUM Command

The status bytes for the COMPUTE CRYPTOGRAPHIC CHECKSUM command are listed in Table 11.

Table 11—Status Bytes for COMPUTE CRYPTOGRAPHIC CHECKSUM Command

SW1	SW2	Meaning
'67'	'00'	Wrong length
'69'	'85'	Conditions of use not satisfied
'6A'	'86'	Incorrect parameters P1-P2
'90'	'00'	Normal processing

2.3 GENERATE APPLICATION CRYPTOGRAM

2.3.1 Definition and Scope

The GENERATE AC command sends transaction-related data to the card, which computes and returns an *Application Cryptogram (AC)*. According to the risk management in the card, the cryptogram returned by the card may differ from that requested in the command message. The card may return an AAC (transaction declined), an ARQC (online authorization request) or a TC (transaction approved).

2.3.2 Command Message

The GENERATE AC command message is coded according to Table 12.

Table 12—GENERATE AC Command Message

Code	Value
CLA	'80'
INS	'AE'
P1	Reference Control Parameter (see Table 13)
P2	'00'
Lc	var.
Data	*CDOL* related data
Le	'00'

Table 13—GENERATE AC Reference Control Parameter

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0							AAC
0	1							TC
1	0							ARQC
1	1							RFU
		x						RFU
			0					Combined DDA/AC generation not requested
			1					Combined DDA/AC generation requested
				x	x	x	x	RFU

The data field of the command message is coded according to the *CDOL* following the rules as defined in Section 4.2 of PART II.

2.3.3 Data Field Returned in the Response Message

In the case of combined DDA/AC generation the response message data field must be coded according to format 2 as specified in Part II of *[EMV BOOK 3]* and must contain at least the three mandatory data objects specified in Table 14, and optionally the *Issuer Application Data.*

Table 14—GENERATE AC Response Message Data Field for Combined DDA/AC Generation

Tag	Value		Presence
'77'	*Response Message Template*		M
	'9F27'	*Cryptogram Information Data*	M
	'9F36'	*Application Transaction Counter*	M
	'9F4B'	*Signed Dynamic Application Data*	M
	'9F10'	*Issuer Application Data*	O

The data field of the response message for an AAC, ARQC or TC in the case no combined DDA/AC generation is performed, is specified in Table 15.

Table 15—GENERATE AC Response Message Data Field without Combined DDA/AC Generation

Tag	Value		Presence
'77'	*Response Message Template*		M
	'9F27'	*Cryptogram Information Data*	M
	'9F36'	*Application Transaction Counter*	M
	'9F26'	*Application Cryptogram*	M
	'9F10'	*Issuer Application Data*	O

Additional data objects returned in the data field that are not described in this specification must be ignored by the terminal.

2.3.4 Status Bytes for GENERATE AC Command

The status bytes for the GENERATE AC command are listed in Table 16.

Table 16—Status Bytes for GENERATE AC Command

SW1	SW2	Meaning
'67'	'00'	Wrong length
'69'	'85'	Conditions of use not satisfied
'6A'	'86'	Incorrect parameters P1-P2
'90'	'00'	Normal processing

2.4 GET PROCESSING OPTIONS

2.4.1 Definition and Scope

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

2.4.2 Command Message

The GET PROCESSING OPTIONS command message is coded according to Table 17.

Table 17—GET PROCESSING OPTIONS Command Message

Code	Value
CLA	'80'
INS	'A8'
P1	'00'
P2	'00'
Lc	var.
Data	*PDOL* related data
Le	'00'

The data field of the command message is the *Command Template* with tag '83' and with a value field coded according to the *PDOL* provided by the card in the response to the SELECT command. If the *PDOL* is not provided by the card, then the length field of the template is set to zero. Otherwise the length field is the total length of the value fields of the data objects transmitted to the card. The value fields are concatenated according to the rules defined in Section 4.2 of PART II.

2.4.3 Data Field Returned in the Response Message

The data field of the response message is a constructed data object with tag '77' (*Response Message Template*). The value field may include several BER-TLV coded objects, but must always include the *AIP* (tag '82') and *AFL* (tag '94').

Table 18—GET PROCESSING OPTIONS Response Message Data Field

Tag	Value		Presence
'77'	*Response Message Template*		M
	'82'	*Application Interchange Profile* (*AIP*)	M
	'94'	*Application File Locator* (*AFL*)	M

Additional data objects returned in the data field that are not described in this specification must be ignored by the terminal.

2.4.4 Status Bytes for GET PROCESSING OPTIONS Command

The status bytes for the GET PROCESSING OPTIONS command are listed in Table 19.

Table 19—Status Bytes for GET PROCESSING OPTIONS Command

SW1	SW2	Meaning
'67'	'00'	Wrong length
'69'	'85'	Conditions of use not satisfied
'6A'	'86'	Incorrect parameters P1-P2
'90'	'00'	Normal processing

2.5 READ RECORD

2.5.1 Definition and Scope

The READ RECORD command reads a file record in a linear file. The response of the card consists of returning the record.

2.5.2 Command Message

The READ RECORD command message is coded according to Table 20.

Table 20—READ RECORD Command Message

Code	Value
CLA	'00'
INS	'B2'
P1	Record Number
P2	See Table 21
Lc	Not present
Data	Not present
Le	'00'

Table 21 specifies the coding of P2.

Table 21—P2 of READ RECORD Command

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

2.5.3 Data Field Returned in the Response Message

The data field of the response message contains the record read. For SFIs in the range 1-10, the record is a BER-TLV constructed data object with tag '70' as shown in Table 22.

Table 22—READ RECORD Response Message Data Field

'70'	Length	Record Template

2.5.4 Status Bytes for READ RECORD Command

The status bytes for the READ RECORD command are listed in Table 23.

Table 23—Status Bytes for READ RECORD Command

SW1	SW2	Meaning
'6A'	'82'	Incorrect parameters P1 P2; file not found
'6A'	'83'	Incorrect parameters P1 P2; record not found
'6A'	'86'	Incorrect parameters P1 P2
'90'	'00'	Normal processing

2.6 SELECT

2.6.1 Definition and Scope

The SELECT command is used to select the *PayPass* application corresponding to the submitted *AID*. The response from the card consists of returning the *FCI*.

2.6.2 Command Message

The SELECT command message is coded according to Table 24.

Table 24—SELECT Command Message

Code	Value
CLA	'00'
INS	'A4'
P1	'04'
P2	'00': first occurrence '02': next occurrence
Lc	'05' – '10'
Data	*AID*
Le	'00'

The data field of the command message contains the *AID* of the *PayPass* application.

2.6.3 Data Field Returned in the Response Message

The data field of the response message contains the *FCI* specific to the selected ADF. Additional tags returned in the *FCI* that are not described in this specification must be ignored by the terminal.

Table 25 defines the *FCI* returned by a successful selection of an ADF.

Table 25—SELECT Response Message Data Field (FCI) of an ADF

Tag	Value			Presence
'6F'	*FCI Template*			M
	'84'	*DF Name (AID)*		M
	'A5'	*FCI Proprietary Template*		M
		'50'	*Application Label*	O
		'87'	*Application Priority Indicator*	O
		'5F2D'	*Language Preference*	O
		'9F38'	*PDOL*	O
		'9F11'	*Issuer Code Table Index*	O
		'9F12'	*Application Preferred Name*	O
		'BF0C'	*FCI Issuer Discretionary Data*	O
			'XXXX' 1 or more additional data elements from application provider, Issuer or ICC supplier	O

2.6.4 Status Bytes for SELECT Command

The status bytes for the SELECT command are listed in Table 26.

Table 26—Status Bytes for SELECT Command

SW1	SW2	Meaning
'62'	'83'	Selected file invalidated
'67'	'00'	Wrong length
'6A'	'81'	Function not supported
'6A'	'82'	File not found
'6A'	'86'	Incorrect parameters P1-P2
'90'	'00'	Normal processing

3 Transaction Flow

This chapter specifies the interaction between a *PayPass* card (Mag Stripe or M/Chip) and a *PayPass* – M/Chip terminal. The flowchart in Section 3.1 describes the transaction flow on an online capable terminal. The flowchart in Section 3.2 describes the transaction flow on an offline-only terminal. The flowcharts are only examples, and the order of processing may differ from that given here.

3.1 Transaction Flow for Online Capable Terminal

The flowchart in Figure 5 gives an example of a transaction flow that may be used by an online capable terminal. The remainder of this section explains the symbols used in Figure 5.

Symbol 0 – Application Selection

The card and terminal perform application selection as specified in Section 1 of PART II.

Symbol 1 – Initiate Application

The terminal issues the GET PROCESSING OPTIONS command. If there is no *PDOL* in the ICC, the terminal uses the command data field '8300'. Otherwise the command data field contains a data object with tag '83' and a value field comprising the concatenated list of data elements resulting from processing the *PDOL*.
The card returns the *Application Interchange Profile (AIP)* and the *Application File Locator (AFL)*.

Symbol 2 – M/Chip profile?

The terminal verifies if the 'M/Chip profile is supported' bit in the *AIP* is set. If this is the case, then the terminal continues with reading the *PayPass* – M/Chip application data. If the bit is not set, then the terminal continues with reading the *PayPass* – Mag Stripe application data.

Symbol 3 – Read Mag Stripe Application Data

The *AFL* has a fixed value indicating that all *PayPass* – Mag Stripe data to be read by the terminal are included in record 1 of the file with SFI 1. The terminal reads the necessary data using one READ RECORD command as specified in Section 4.7 of PART II.

Symbol 4 – Mag Stripe Application Version Number Checking

The terminal verifies the compatibility of the *PayPass* – Mag Stripe application in the terminal with the *PayPass* – Mag Stripe application in the card as specified in Section 4.8 of PART II.

Symbol 5 – COMPUTE CRYPTOGRAPHIC CHECKSUM

The terminal continues with the COMPUTE CRYPTOGRAPHIC CHECKSUM command as specified in Section 4.9 of PART II using a data field comprising the concatenated list of data elements resulting from processing the *UDOL*.

The COMPUTE CRYPTOGRAPHIC CHECKSUM command initiates the computation of the dynamic $CVC3_{TRACK2}$ and $CVC3_{TRACK1}$ (optional) in the *PayPass* card. The computation is based on the *Unpredictable Number* (*UN*) sent by the terminal, the *Application Transaction Counter* (*ATC*) of the card (optional) and the relevant secret key stored in the card.

The card responds with the dynamic $CVC3_{TRACK2}$, the $CVC3_{TRACK1}$ (optional) and the *ATC*. The terminal converts the binary $CVC3_{TRACK2}$ into BCD encoded digits and copies the relevant digits in the discretionary data field of the *Track 2 Data* at the places indicated by the *Track 2 Bit Map for CVC3* ($PCVC3_{TRACK2}$). The terminal copies also the relevant digits of the *Unpredictable Number* (*UN*) into the discretionary data field of the *Track 2 Data*. The *Track 2 Bit Map for UN and ATC* ($PUNATC_{TRACK2}$) indicates where the terminal must copy the *UN* digits in the discretionary data field of the *Track 2 Data*. If the number of *ATC* digits to be included in the discretionary data field is not zero (indicated by $NATC_{TRACK2}$), then the terminal must convert the *ATC* into BCD encoded digits and copy the relevant *ATC* digits into the discretionary data field of the *Track 2 Data* at the places indicated by $PUNATC_{TRACK2}$. After copying the $CVC3_{TRACK2}$, *UN* and *ATC* digits, the terminal copies the number of *UN* digits (n_{UN}) in the least significant digit of the discretionary data field. A similar procedure is repeated for *Track 1 Data* if the card also returned *Track 1 Data* in the response to the READ RECORD command. For the *Track 1 Data* the terminal converts the data returned by the card into ASCII encoded characters before copying them into the discretionary data field of the *Track 1 Data*.

After the completion of the COMPUTE CRYPTOGRAPHIC CHECKSUM, the card may be removed from the electromagnetic field.

Symbol 6 – Mag Stripe Cardholder Verification

The terminal performs Mag Stripe cardholder verification. The cardholder verification function makes use of the *Mag Stripe CVM List* (tag '9F68') data element returned by the *PayPass* – Mag Stripe card in the response to the READ RECORD command performed during Read Mag Stripe Application Data. The *Mag Stripe CVM List* is coded as specified in *[EMV BOOK 3]* with the limitations specified in Section 6.8 of PART II.

Symbol 7 – Read M/Chip Application Data

The *AFL* has a fixed value as indicated in Section 5.13.2 of PART II. The terminal reads the necessary data using READ RECORD commands as specified in Section 4.11 of PART II.

Symbol 8 – Processing Restrictions

The terminal performs the processing restrictions function as specified in Section 4.12 of PART II. This includes application version number checking, application usage control checking and application effective/expiry dates checking.

Symbol 9 – Terminal Risk Management

The terminal performs the terminal risk management functions as specified in Section 4.13 of PART II. This includes floor limit checking, random transaction selection and exception file checking.

Symbol 10 – M/Chip Cardholder Verification

The terminal performs M/Chip cardholder verification. The cardholder verification function makes use of the *M/Chip CVM List* (tag '8E') data element returned by the *PayPass* – M/Chip card during Read M/Chip Application Data. The *M/Chip CVM List* is coded as specified in *[EMV BOOK 3]*.

Symbol 11 – Terminal Action Analysis

Once terminal risk management related to the transaction has been completed, the terminal makes the decision as to whether the transaction should be approved offline, declined offline, or transmitted online. The terminal makes this decision based upon the content of the *TVR*, *Issuer Action Codes* and *Terminal Action Codes* as specified in Section 4.17 of PART II.

Symbol 12 – Card Action Analysis

The card action analysis process is performed when the terminal issues the GENERATE AC command. During the card action analysis the card performs its own card risk management to protect the issuer from fraud or excessive credit risk. As the result of this risk management process, the card may decide to complete a transaction online, offline or decline the transaction.

Symbol 13 – Card Generated AAC/AAR?

If the card generated an AAC or AAR, then the terminal declines the transaction.

Symbol 14 – Combined DDA/AC Generation?

If combined DDA/AC generation is performed, then the terminal continues with retrieving the *ICC Public Key* and verifying the *Signed Dynamic Application Data*. If no combined DDA/AC generation is performed, then the terminal continues with verifying if the card generated an ARQC.

Symbol 15 – Card Generated ARQC?

The terminal verifies if the card generated an ARQC. If this is the case, then the terminal completes the transaction online. If this is not the case, then the terminal continues with offline CAM by performing static data authentication.

Symbol 16 – Static Data Authentication

The terminal performs static data authentication as specified in Section 4.15 of PART II.

Symbol 17 – SDA OK?

If static data authentication failed, then the terminal declines the transaction. Otherwise, the terminal accepts the transaction offline.

Symbol 18 – Retrieve ICC Public Key and Verify SDAD

In the case of combined DDA/AC generation, the terminal retrieves the *ICC Public Key* and verifies the *Signed Dynamic Application Data* generated by the card as specified in Section 4.15 of PART II.

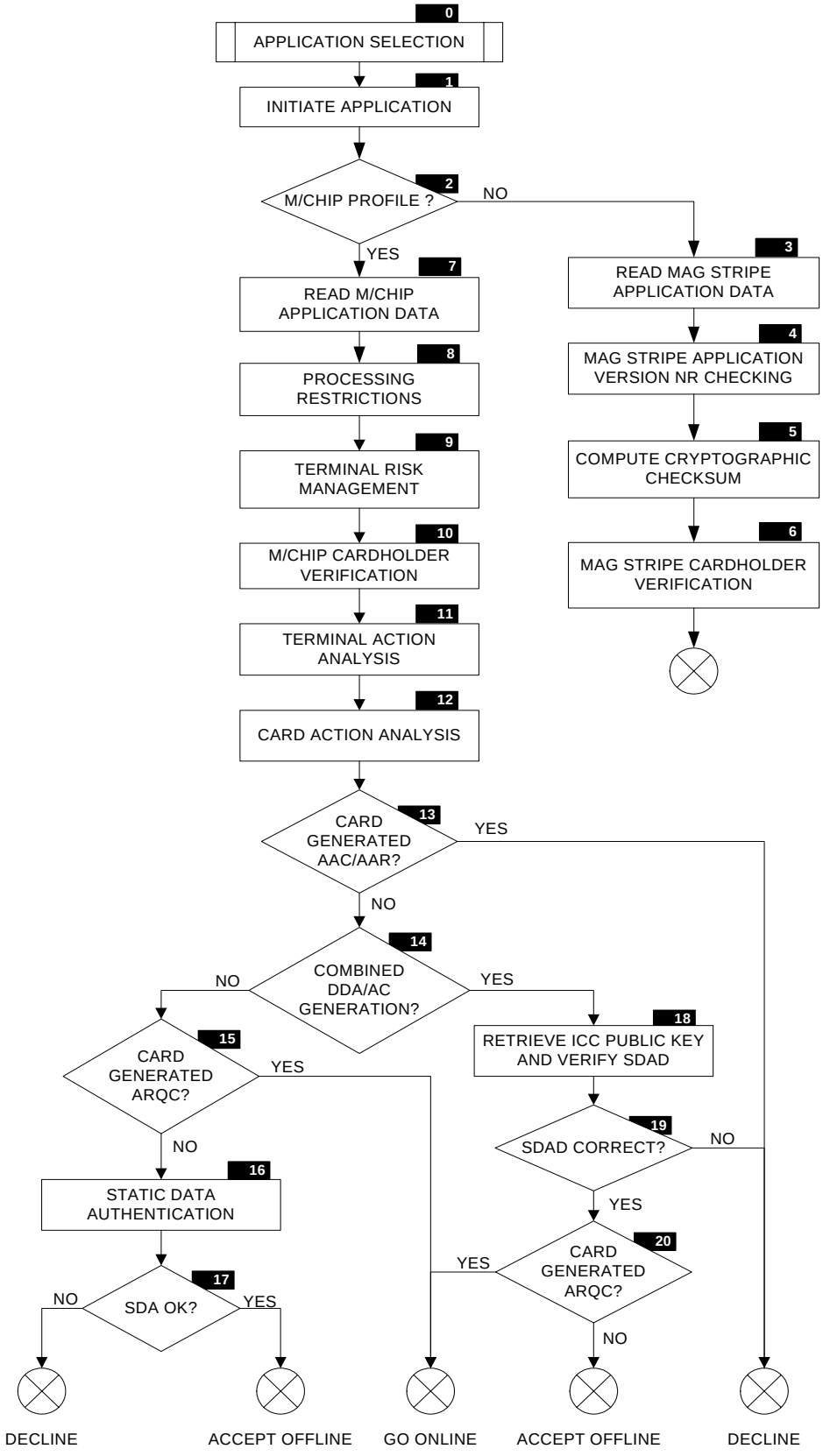
Symbol 19 – SDAD Correct?

If the *Signed Dynamic Application Data* is not correct, then the terminal declines the transaction.

Symbol 20 – Card Generated ARQC?

If the card generated an ARQC, then the terminal completes the transaction online. Otherwise, the transaction is accepted offline.

Figure 5—Transaction Flow for Online Capable *PayPass* Terminals



3.2 Transaction Flow for Offline-Only Terminal

The flowchart in Figure 7 proposes an alternative transaction flow for a *PayPass* – M/Chip transaction on offline-only *PayPass* terminals. The offline-only transaction flow minimizes the terminal processing while the card is in the field.

As the *TVR* is not used by the card risk management of the *PayPass* – M/Chip 4 card application, the terminal may postpone the terminal risk management functions until the card is removed from the field. This way the terminal only needs to retrieve the *CDOL1* from the first record to be able to process the GENERATE AC command. All other terminal processing is done after the GENERATE AC command.

Symbol 0 – Application Selection

The card and terminal perform application selection as specified in Section 1 of PART II.

Symbol 1 – Initiate Application

The terminal issues the GET PROCESSING OPTIONS command. If there is no *PDOL* in the ICC, the terminal uses the command data field '8300'. Otherwise the command data field contains a data object with tag '83' and a value field comprising the concatenated list of data elements resulting from processing the *PDOL*.
The card returns the *Application Interchange Profile (AIP)* and the *Application File Locator (AFL)*.

Symbol 2 – M/Chip profile?

The terminal verifies if the 'M/Chip profile is supported' bit in the *AIP* is set. If this is the case, then the terminal continues with reading the *PayPass* – M/Chip application data. If the bit is not set, then the terminal continues with a *PayPass* – Mag Stripe transaction.

Symbol 3 – Perform *PayPass* – Mag Stripe Transaction

The reader performs a *PayPass* – Mag Stripe transaction. This includes the symbols 3, 4, 5 and 6 of Figure 5.

Symbol 4: Read M/Chip Application Data

The terminal reads all the data from the card as specified in Section 4.11 of PART II, but retrieves only the *CDOL1* from the response message. All other data is saved for later use.

Symbol 5: Generate TC

The terminal always requests a TC.

Symbol 6: Card Generated AAC, AAR or ARQC

If the card generates an AAC, AAR or ARQC, then the terminal declines the transaction without further processing.

Symbol 7 and 9: Combined DDA/AC Generation

If Combined DDA/AC Generation was requested, then the terminal retrieves the *ICC Public Key* and verifies the *Signed Dynamic Application Data* generated by the card as specified in Section 4.15 of PART II. If this verification fails, then the terminal sets the appropriate bit in the *TVR*.

Symbol 8: Static Data Authentication

The terminal performs static data authentication as specified in Section 4.15 of PART II. The terminal sets the appropriate bit in the *TVR* if the static data authentication failed.

Symbol 10, 11: Processing Restrictions, Terminal Risk Management

The terminal performs processing restrictions and terminal risk management as specified in Section 4.12 and 4.13 of PART II. The terminal sets the appropriate bits in the *TVR* if one or more tests fail.

Symbol 12: M/Chip Cardholder Verification

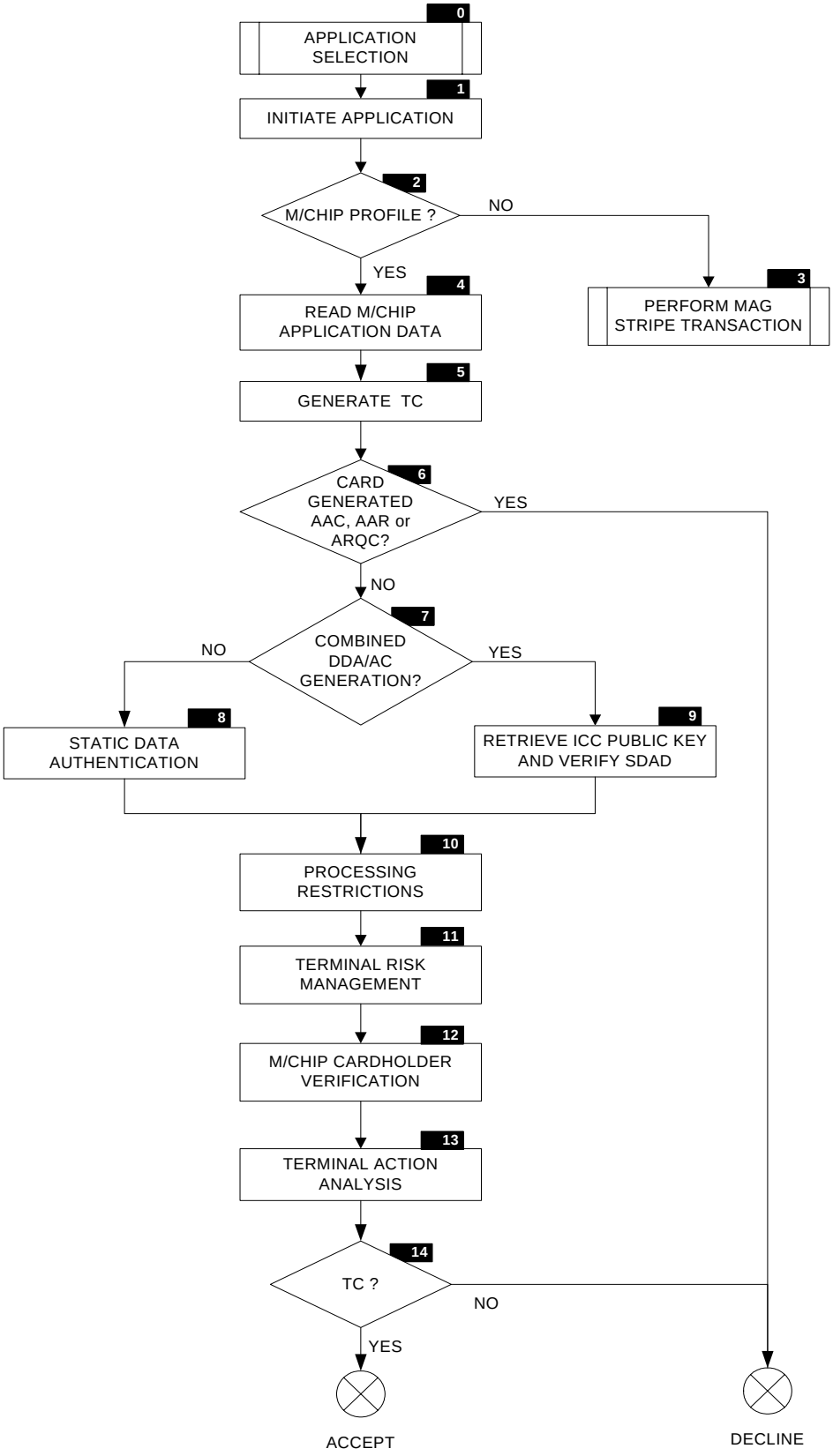
The terminal performs M/Chip cardholder verification as specified in Section 4.14 of PART II.

Symbol 13: Terminal Action Analysis

The terminal performs terminal action analysis as specified in 4.16 of PART II. If the result is a TC request, then the terminal accepts the transaction. Otherwise the transaction is declined.

Note **For the clearing record, the terminal must use the *TVR* as sent to the card, not the *TVR* used to collect the terminal risk management results.**

Figure 6—Transaction Flow for Offline Only *PayPass* Terminals



4 Terminal Interoperability Requirements

4.1 Transmission Protocol

4.1.1.1 A MasterCard *PayPass* terminal shall be compliant with *[PAYPASS ISO/IEC 14443]*.

4.2 DOL Handling

To minimize processing in the card, the data field of the command messages is not TLV encoded. The application in the card indicates the requested data, including format and length, by sending a Data Object List (DOL) to the terminal. DOLs used in this specification include the *PDOL* used with the GET PROCESSING OPTIONS command, the *CDOL1* and *CDOL2* used with the GENERATE AC command, the *TDOL* used to generate the *TC Hash Value* and the *UDOL* used with the COMPUTE CRYPTOGRAPHIC CHECKSUM command.

4.2.1.1 DOL Handling shall be performed according to the rules specified in Section 5.4 of *[EMV BOOK 3]*.

4.3 Exception Processing

4.3.1 Data Objects

Data objects returned by the card must be checked by the terminal as follows:

4.3.1.1 All data elements in the card listed in *[EMV BOOK 1]*, *[EMV BOOK 3]* and Section 6 of PART II of this document are classified as either mandatory or optional. When any mandatory data element is missing, the terminal shall decline the transaction.

4.3.1.2 It is up to the issuer to ensure that data in the card is of the correct format, and no format checking other than that specifically defined is mandated on the part of the terminal. However, if in the course of normal processing the terminal recognizes that data is incorrectly formatted, then the terminal shall decline the transaction. This rule includes (but is not limited to):

- Constructed data objects that do not parse correctly.
- Data that must be in a specific range of values but are not.
- A *CVM List* with no Cardholder Verification rules.
- Multiple occurrences of a data object that shall only appear once.
- An *AFL* with invalid syntax (e.g. a starting record of 0).

4.3.1.3 During a *PayPass* – Mag Stripe transaction the terminal shall verify if the *Track 1 Data* (if available) and *Track 2 Data* are formatted as specified in Sections 6.11 and 6.15 of PART II. The terminal shall perform this verification after all dynamic data (i.e. n_{UN}, *ATC*, *UN* and *CVC3*) are copied into the discretionary data fields (i.e. after the COMPUTE CRYPTOGRAPHIC CHECKSUM command processing as specified in Section 4.9 of PART II has been completed). However, if in the course of copying the dynamic data, the terminal is not able to localize the discretionary data field due to one or more format errors in the *Track 1 Data* or *Track 2 Data* (e.g. missing separator), then the terminal shall decline the transaction immediately.

4.3.2 Status Bytes

4.3.2.1 Any SW1-SW2 returned by the card other than '9000' or '6283' shall cause termination of the transaction.

Note **Requirements 4.3.1.1, 4.3.1.2 and 4.3.2.1 do not apply to the selection of the PPSE and the final SELECT command.**

4.4 Application Selection

4.4.1.1 A terminal supporting more than one application shall process the application selection as specified in Section 1 of PART II.

Note **A terminal that supports only one application (= *AID*) should immediately perform the SELECT command with the appropriate *AID*.**

4.5 Final SELECT Command Processing

This section lists the terminal requirements related to the final SELECT command sent by the terminal to select the *PayPass* application.

4.5.1.1 The terminal shall format the SELECT command as specified in Section 2.6.2 of PART II.

4.5.1.2 The terminal shall verify if the *FCI* included in the response message of the SELECT command is correctly formatted as specified in Section 2.6.3 of PART II. If this is not the case, then the terminal shall terminate the transaction.

4.5.1.3 The terminal shall verify if the *DF Name* (tag '84') returned in the *FCI* is the same as the *AID* provided to the card in the data field of the SELECT command message. If this is not the case, then the terminal shall terminate the transaction.

4.5.1.4 The terminal shall extract the *Application Label* (tag '50'), the *Issuer Code Table Index* (tag '9F11') (if present) and the *Application Preferred Name* (tag '9F12') (if present) from the *FCI* and store them for later use during transaction processing.

4.5.1.5 If the *Language Preference* (tag '5F2D') data object is included in the *FCI*, then the terminal shall perform language selection as specified in EMV BOOK 4, Section 11.1 Language Selection.

4.5.1.6 If the *PDOL* exists, then the terminal shall extract it from the *FCI* to use it for the construction of the data field of the GET PROCESSING OPTIONS command.

4.6 Initiate Application Processing

The initiate application processing is the first function performed after application selection. The terminal issues the GET PROCESSING OPTIONS command to initiate the transaction in the card.

4.6.1.1 The terminal sets all bits in the *Transaction Status Information (TSI)* and the *Terminal Verification Results (TVR)* to 0b.

4.6.1.2 The terminal shall format the GET PROCESSING OPTIONS command as specified in Section 2.4.2 of PART II.

4.6.1.3 If the *PDOL* is not present, then the terminal shall use a command data field of '83 00'.

4.6.1.4 If the *PDOL* is present, then the terminal shall use the *PDOL* to create a concatenated list of data elements without tags or lengths following the rules specified in Section 4.2 of PART II. The terminal shall verify if the tags in the *PDOL* belong to terminal resident data objects. If the tag of any data object identified in the *PDOL* does not belong to a terminal resident data object, then the terminal shall provide a data element with the length specified and a value of all hexadecimal zeros. The terminal shall use the concatenated list as value field of the data object with tag '83'.

4.6.1.5 The terminal shall verify if the response message of the GET PROCESSING OPTIONS command is correctly formatted as specified in Section 2.4.3 of PART II. If this is not the case, then the terminal shall terminate the transaction.

4.6.1.6 The terminal shall retrieve from the response message the *AIP* (tag '82') and *AFL* (tag '94') data objects. If they are not both included, then the terminal shall terminate the transaction.

4.6.1.7 If the card returns SW1-SW2='6985' in response to the GET PROCESSING OPTIONS command, then the terminal shall return to application selection. The terminal shall not allow the application to be selected again.

4.6.1.8 The terminal shall ignore all data objects that are included in the *Response Message Template* (tag '77') and that are different from the *AIP* and *AFL*.

4.7 Read Mag Stripe Application Data

If the 'M/Chip profile is supported' bit in the *AIP* is not set (i.e. the card is a *PayPass* – Mag Stripe card), then the terminal continues with the Read Mag Stripe Application Data processing. Data contained in files in the *PayPass* – Mag Stripe card are required by the terminal to complete the COMPUTE CRYPTOGRAPHIC CHECKSUM command processing. The terminal reads the files and records indicated in the *AFL* using the READ RECORD command.

4.7.1.1 If the value of the 4 most significant bytes of the *AFL* is different from the value of the 4 most significant bytes of the *AFL*s listed in requirement 5.13.2.1 and 5.13.2.2, then the terminal shall process the *AFL* as specified in *EMV BOOK 3*, Section 10.2.

4.7.1.2 If the value of the 4 most significant bytes of the *AFL* is the same as the value of the 4 most significant bytes of the *AFL*s listed in requirement 5.13.2.1 and 5.13.2.2, then the terminal shall not interpret the *AFL* and only read the first record in the file with SFI 1.

4.7.1.3 The terminal shall store all recognized data objects read, whether mandatory or optional, for later use in the transaction processing. Data objects that are not recognized by the terminal (that is, their tags are unknown by the terminal) shall not be stored.

4.7.1.4 If the terminal encounters more than one occurrence of a single primitive data object during read application data processing, the transaction shall be terminated.

4.7.1.5 All mandatory data objects must be present in the *PayPass* – Mag Stripe card. If any mandatory data object is not present, then the terminal shall terminate the transaction. The mandatory data objects are listed in Table 27.

Table 27—Mandatory Mag Stripe Data Objects

Tag	Value
'9F6B'	*Track 2 Data*
'9F66'	*Track 2 Bit Map for UN and ATC* (*PUNATC*$_{TRACK2}$)
'9F65'	*Track 2 Bit Map for CVC3* (*PCVC3*$_{TRACK2}$)
'9F67'	*Track 2 Number of ATC Digits* (*NATC*$_{TRACK2}$)

4.8 Mag Stripe Application Version Number Checking

The application within both the card and the terminal maintain a *Mag Stripe Application Version Number* assigned by the payment system. The terminal shall verify the compatibility of the *Mag Stripe Application Version Number (Terminal)* in the terminal with the *Mag Stripe Application Version Number (Card)* in the card.

4.8.1.1 The terminal shall use the version number (*Mag Stripe Application Version Number (Card)*) in the card to ensure compatibility If the *Mag Stripe Application Version Number (Card)* is not present in the card, then the terminal shall presume the terminal and card application versions are compatible.

4.8.1.2 If the *Mag Stripe Application Version Number (Card)* is present in the card and the terminal supports the application version of the card, then the terminal shall use the appropriate code and/or commands to deal with the card. If the terminal does not recognize the application version of the card, then the terminal shall use its latest version to deal with the card.

4.9 COMPUTE CRYPTOGRAPHIC CHECKSUM Command Processing

The terminal issues the COMPUTE CRYPTOGRAPHIC CHECKSUM command to the card to retrieve the $CVC3_{TRACK2}$, the $CVC3_{TRACK1}$ (optional) and the *ATC* from the card.

4.9.1.1 The terminal shall verify that the number of bits in $PUNATC_{TRACK2}$ (k_{TRACK2}) is greater than or equal to the number of digits of the *ATC* to be included in the discretionary data field of the *Track 2 Data* (t_{TRACK2}). If $k_{TRACK2} < t_{TRACK2}$, then the terminal shall terminate the transaction. Otherwise, the terminal shall set n_{UN} equal to $k_{TRACK2} - t_{TRACK2}$.

4.9.1.2 The terminal shall verify that n_{UN} is less than or equal to 8. If n_{UN} is greater than 8, then the terminal shall terminate the transaction.

4.9.1.3 The terminal shall verify that the number of bits in $PCVC3_{TRACK2}$ is greater than or equal to 3 (i.e. $q_{TRACK2} \geq 3$). If this is not the case, then the terminal shall terminate the transaction.

4.9.1.4 If *Track 1 Data* is included in the data returned from the card, then the terminal shall verify that also $PCVC3_{TRACK1}$, $PUNATC_{TRACK1}$ and $NATC_{TRACK1}$ are returned. If at least one of these data elements is not available, then the terminal shall terminate the transaction.

4.9.1.5 If *Track 1 Data* is available, then the terminal shall verify that the number of bits in $PUNATC_{TRACK1}$ (k_{TRACK1}) is greater than or equal to the number of digits of the *ATC* to be included in the discretionary data field of *Track 1 Data* (t_{TRACK1}). If $k_{TRACK1} < t_{TRACK1}$, then the terminal shall terminate the transaction.

4.9.1.6 If *Track 1 Data* is available, then the terminal shall verify that $k_{TRACK1} - t_{TRACK1}$ is equal to n_{UN}. If this is not the case, then the terminal shall terminate the transaction.

4.9.1.7 If *Track 1 Data* is available, then the terminal shall verify that the number of bits in $PCVC3_{TRACK1}$ is greater than or equal to 3 (i.e. $q_{TRACK1} \geq 3$). If this is not the case, then the terminal shall terminate the transaction.

4.9.1.8 The terminal shall retrieve from the *Track 2 Data* the *PAN* and *Expiry Date*. If *Track 1 Data* is returned from the card, then the terminal shall verify that the *PAN* and *Expiry Date* included in the *Track 1 Data* are the same as the *PAN* and *Expiry Date* included in the *Track 2 Data*. If this is not the case, then the terminal shall terminate the transaction.

4.9.1.9 The terminal shall format the COMPUTE CRYPTOGRAPHIC CHECKSUM command as specified in Section 2.2.2 of PART II.

4.9.1.10 The terminal shall generate an *Unpredictable Number (Numeric)* of 8 digits in length of which the 8-n_{UN} most significant digits are set equal to 0 (refer to Section 6.18 of PART II).

4.9.1.11 If the *UDOL* is returned by the card during the Read Mag Stripe Application Data processing, then the terminal shall create a concatenated list of data elements without tags or lengths following the rules specified in Section 4.2 of PART II.

4.9.1.12 If the *UDOL* is not returned by the card during the Read Mag Stripe Application Data processing, then the terminal shall use the *Default Terminal UDOL* to construct the data field of the command message. Refer to Section 6.5 of PART II for the definition of the *Default Terminal UDOL*.

4.9.1.13 If the terminal does not receive a valid response message from the card (i.e. no response message or an invalid response message), then the terminal shall wait 300 ms before processing is continued. If this is the second consecutive transaction for which no valid response message from the card for the COMPUTE CRYPTOGRAPHIC CHECKSUM command is received, then the terminal shall wait 2 * 300 ms before processing is continued. In general, if this is the n^{th} (n= 1, 2, 3, …) consecutive transaction for which no valid response message from the card for the COMPUTE CRYPTOGRAPHIC CHECKSUM command is received, then the terminal shall wait 2^m * 300 ms (m being the minimum of n-1 and 5) before processing is continued.

4.9.1.14 The terminal shall verify if the response message of the COMPUTE CRYPTOGRAPHIC CHECKSUM command is correctly formatted as specified in Section 2.2.3 of Part II. If not, then the terminal shall terminate the transaction as indicated in 4.9.1.13.

4.9.1.15 The terminal shall retrieve the $CVC3_{TRACK2}$ (tag '9F61') and the *ATC* (tag '9F36') from the *Response Message Template* (tag '77'). If one of these data objects is not available, then the terminal shall terminate the transaction as indicated in 4.9.1.13.

4.9.1.16 The terminal shall convert the binary encoded $CVC3_{TRACK2}$ to the BCD encoding of the corresponding number expressed in base 10. The terminal shall copy the q_{TRACK2} least significant digits of the BCD encoded $CVC3_{TRACK2}$ in the eligible positions of the discretionary data field of *Track 2 Data*. The eligible positions are indicated by the q_{TRACK2} non-zero bits in $PCVC3_{TRACK2}$.

4.9.1.17 The terminal shall replace the n_{UN} least significant eligible positions of the discretionary data field of *Track 2 Data* by the n_{UN} least significant digits of *UN*. The eligible positions in the discretionary data field are indicated by the n_{UN} least significant non-zero bits in $PUNATC_{TRACK2}$.

4.9.1.18 If $t_{TRACK2} \neq 0$, then the terminal shall convert the *ATC* to the BCD encoding of the corresponding number expressed in base 10. The terminal shall replace the t_{TRACK2} most significant eligible positions of the discretionary data field of *Track 2 Data* by the t_{TRACK2} least significant digits of the BCD encoded *ATC*. The eligible positions in the discretionary data field are indicated by the t_{TRACK2} most significant non-zero bits in $PUNATC_{TRACK2}$.

4.9.1.19 The terminal shall copy n_{UN} into the least significant digit of the discretionary data field of the *Track 2 Data*.

4.9.1.20 If *Track 1 Data* is available, then the terminal shall retrieve the $CVC3_{TRACK1}$ from the *Response Message Template* (tag '77'). If the *Track 1 Data* is available and the $CVC3_{TRACK1}$ is not available, then the terminal shall terminate the transaction as indicated in 4.9.1.13.

4.9.1.21 If *Track 1 Data* is available, then the terminal shall convert the binary encoded $CVC3_{TRACK1}$ to the BCD encoding of the corresponding number expressed in base 10. The terminal shall convert the q_{TRACK1} least significant digits of the BCD encoded $CVC3_{TRACK1}$ into the ASCII format and copy the q_{TRACK1} ASCII encoded $CVC3_{TRACK1}$ characters into the eligible positions of the discretionary data field of the *Track 1 Data*. The eligible positions are indicated by the q_{TRACK1} non-zero bits in $PCVC3_{TRACK1}$.

4.9.1.22 If *Track 1 Data* is available, then the terminal shall convert the BCD encoded *UN* into the ASCII format and replace the n_{UN} least significant eligible positions of the discretionary data field of the *Track 1 Data* by the n_{UN} least significant characters of the ASCII encoded *UN*. The eligible positions in the discretionary data field are indicated by the n_{UN} least significant non-zero bits in $PUNATC_{TRACK1}$.

4.9.1.23 If *Track 1 Data* is available and $t_{TRACK1} \neq 0$, then the terminal shall convert the *ATC* to the BCD encoding of the corresponding number expressed in base 10. The terminal shall convert the t_{TRACK1} least significant digits of the *ATC* into the ASCII format. The terminal shall replace the t_{TRACK1} most significant eligible positions of the discretionary data field of the *Track 1 Data* by the t_{TRACK1} ASCII encoded *ATC* characters. The eligible positions in the discretionary data field are indicated by the t_{TRACK1} most significant non-zero bits in $PUNATC_{TRACK1}$.

4.9.1.24 If *Track 1 Data* is available, then the terminal shall convert n_{UN} into the ASCII format and copy the ASCII encoded n_{UN} character into the least significant position of the discretionary data field of the *Track 1 Data*.

4.9.1.25 The terminal shall execute the requirements 4.9.1.16, 4.9.1.17, 4.9.1.18 and 4.9.1.19 and the requirements 4.9.1.21, 4.9.1.22, 4.9.1.23 and 4.9.1.24 in the order as specified above.

4.10 Mag Stripe Cardholder Verification

Cardholder verification is performed to ensure that the person presenting the *PayPass* card is the person to whom the application in the card was issued. This section specifies how the terminal must perform cardholder verification for a *PayPass* – Mag Stripe card. The cardholder verification function makes use of the *Mag Stripe CVM List* (tag '9F68') data element returned by the *PayPass* – Mag Stripe card in the response to the READ RECORD command. Refer to *EMV BOOK 3* and Section 6.8 of PART II for the coding of the *Mag Stripe CVM List*.

The terminal performs cardholder verification as follows:

4.10.1.1 If the *Mag Stripe CVM List* is not present in the *PayPass* – Mag Stripe card, then the terminal may obtain cardholder verification by means of the Cardholder Verification Method (CVM) as supported by the terminal.

4.10.1.2 If the *CVM List* is present in the *PayPass* – Mag Stripe card, then the terminal shall process each Cardholder Verification Rule (CVR) in the order in which it appears in the list. Cardholder verification is completed when any one CVM is successfully performed or when the list is exhausted.

4.10.1.3 If any of the following are true:

- The conditions expressed in the second byte of a CVR are not satisfied, or
- The CVM condition code is outside the range of codes listed in Section 6.8 of PART II,

then the terminal shall bypass the rule and proceed to the next CVR in the *CVM List*. If there are no more CVRs in the list, cardholder verification failed.

4.10.1.4 If the conditions expressed in the second byte of a CVR are satisfied, then the terminal shall attempt to perform the CVM if the CVM code is one of those listed in Section 6.8 of PART II or is otherwise understood by the terminal. If the conditions expressed in the second byte of a CVR are satisfied, but the CVM is not among those listed and is not understood by the terminal then this CVM fails.

4.10.1.5 If the CVM is performed successfully, cardholder verification is complete and successful. Otherwise, the terminal shall examine b7 of byte 1 of the CVM field. If b7 is set to 1b, processing continues with the next CVR, if one is present. If b7 is set to 0b, or there are no more CVRs in the list, cardholder verification failed.

If online PIN processing is the required CVM as determined by the above process, the processing must be performed as follows:

4.10.1.6 The processing is not successfully performed for any one of the following reasons:

- The terminal does not support online PIN.
- The terminal supports online PIN, but the PIN pad is malfunctioning.
- The terminal bypassed PIN entry at the direction of either the merchant or the cardholder.

4.10.1.7 If the online PIN is successfully entered, the cardholder verification is considered successful and complete.

If signature is the required CVM as determined by the above process, the processing must be performed as follows:

4.10.1.8 The terminal shall determine success based upon the terminal's capability to support the signature process. If the terminal is able to support signature, the process is considered successful, and cardholder verification is complete.

4.10.1.9 At the end of the transaction, the terminal shall print a receipt with a line for cardholder signature.

4.11 Read M/Chip Application Data

If the 'M/Chip profile is supported' bit in the *AIP* is set (i.e. the card is a *PayPass* – M/Chip card), then the terminal continues with the Read M/Chip Application Data processing. Data contained in files in the card are required by the terminal to complete the 1st GENERATE AC command and terminal risk management. The terminal reads the files and records indicated in the *AFL* using the READ RECORD command.

4.11.1.1 If the *AFL* is not one of those listed in Section 5.13.2 of PART II, then the terminal read the files and records indicated in the *AFL* as specified in Section 10.2 of *[EMV BOOK 3]*. Otherwise the terminal shall proceed with 4.11.1.2.

4.11.1.2 The terminal always reads record 1 included in the file with SFI 2.

4.11.1.3 If the card supports offline data authentication (static data authentication or combined DDA/AC generation), then the terminal also reads record 1 included in the file with SFI 3.

4.11.1.4 If the card supports static data authentication and the card does not support combined DDA/AC generation, then the terminal also reads record 2 included in the file with SFI 3.

4.11.1.5 If the card supports combined DDA/AC generation, then the terminal also reads the record 1 and 2 included in the file with SFI 4.

4.11.1.6 The terminal shall store all recognized data objects read, whether mandatory or optional, for later use in the transaction processing. Data objects that are not recognized by the terminal (that is, their tags are unknown by the terminal) shall not be stored, but records containing such data objects may still participate in their entirety in offline data authentication, depending upon the coding of the *AFL*.

4.11.1.7 If the terminal encounters more than one occurrence of a single primitive data object during read application data processing, the transaction shall be terminated.

4.11.1.8 All mandatory data objects must be present in the card. If any mandatory data object is not present, then the terminal shall terminate the transaction. The mandatory data objects are listed in Table 28.

Table 28—Mandatory M/Chip Data Objects

Tag	Value
'5F24'	*Application Expiry Date*
'5A'	*Application Primary Account Number* (*PAN*)
'8C'	*Card Risk Management Data Object List 1* (*CDOL1*)
'8D'	*Card Risk Management Data Object List 2* (*CDOL2*)

4.11.1.9 Proprietary data files (i.e. files with SFI outside the range 1 to 10) may or may not conform to this specification (Refer to Table 22). Records in proprietary files may be represented in the *AFL* and may participate in offline data authentication if they are readable without conditions by the READ RECORD command coded according to Section 2.5 of PART II.

4.12 Processing Restrictions

4.12.1.1 Processing restrictions checking shall be performed as specified in *[EMV BOOK 3]* and *[EMV BOOK 4]*. It includes application version number checking, application usage control checking and application effective/expiry dates checking.

4.13 Terminal Risk Management

4.13.1.1 Terminal risk management shall be performed as specified in *[EMV BOOK 3]* and *[EMV BOOK 4]*. It may include floor limit checking, random transaction selection and exception file checking. Velocity checking shall not be performed.

4.14 M/Chip Cardholder Verification

Cardholder verification is performed as specified in *[EMV BOOK 3]* and *[EMV BOOK 4]* taking into account the following restrictions:

4.14.1.1 The terminal shall not support offline PIN. This means that the terminal does not support either offline plaintext PIN verification or offline enciphered PIN verification.

4.14.1.2 If an offline PIN is the selected CVM as determined by the process specified in *[EMV BOOK 3]*, then the terminal shall set to '1' the 'PIN entry required and PIN pad not present or not working' bit in the *TVR*.

4.14.1.3 If online PIN processing is the selected CVM as determined by the process specified in *[EMV BOOK 3]* and the terminal does support online PIN and the PIN pad is functioning, then the terminal shall set to '1' the 'Online PIN entered' bit in the *TVR*. In this case cardholder verification is considered successful and complete. The terminal shall set byte 3 of the *CVM Results* to 'unknown'. The terminal shall perform the online PIN entry after the interaction between card and terminal is completed.

4.15 Offline Data Authentication

Offline data authentication is performed as specified in Chapter 5 and 6 of *[EMV BOOK 2]* and Section 10.3 of *[EMV BOOK 3]*. This section specifies how it is determined whether offline data authentication will be performed, what kind of authentication will be performed and when it will be performed.

4.15.1.1 The terminal shall support static data authentication and combined DDA/AC generation[1]. The terminal may perform dynamic data authentication.

4.15.1.2 If the *Application Interchange Profile (AIP)* of the card indicates that the card supports combined DDA/AC, then the terminal shall perform combined DDA/AC generation. In this case the terminal shall not perform static data authentication or dynamic data authentication.

4.15.1.3 If the *AIP* of the card indicates that the card supports dynamic data authentication and that it does not support combined DDA/AC generation and the terminal supports dynamic data authentication, then the terminal shall perform dynamic data authentication. In this case the terminal shall not perform static data authentication.

4.15.1.4 If the *AIP* of the card indicates that the card does not support combined DDA/AC generation and that it supports dynamic data authentication and static data authentication and the terminal does not support dynamic data authentication, then the terminal shall perform static data authentication.

4.15.1.5 If the *AIP* of the card indicates that the card supports static data authentication and that it does not support combined DDA/AC generation or dynamic data authentication, then the terminal shall perform static data authentication.

4.15.1.6 The terminal may perform static data authentication after the interaction between card and terminal has been completed.

4.16 Terminal Action Analysis

With the terminal action analysis function the terminal makes the decision as to whether the transaction should be approved offline, declined offline, or transmitted online.

4.16.1.1 Terminal action analysis shall be performed as specified in *[EMV BOOK 3]*.

[1] Applicable for POS and CAT 1, 2 and 3 terminals.

4.17 GENERATE AC Processing

As a result of the terminal action analysis processing, the terminal requests the card to generate an *Application Cryptogram* with the GENERATE AC command.

4.17.1.1 The terminal shall format the GENERATE AC command as specified in Section 2.3.2 of PART II.

4.17.1.2 The terminal shall use the *CDOL1* to create a concatenated list of data elements without tags or lengths following the rules specified in Section 4.2 of PART II.

4.17.1.3 The terminal shall verify if the response message of the GENERATE AC command is correctly formatted as specified in Section 2.3.3 of PART II. If not, then the terminal shall terminate the transaction.

4.17.1.4 The terminal shall retrieve the *Cryptogram Information Data* (tag '9F27') and the *Application Transaction Counter* (tag '9F36'). If one of these data objects is not available, then the terminal shall terminate the transaction.

4.17.1.5 The terminal shall verify in the *Cryptogram Information Data* if the card generated an AAC or AAR. If this is the case, then the terminal shall decline the transaction offline.

4.17.1.6 If combined DDA/AC generation was not requested, then the terminal shall verify if the *Application Cryptogram* (tag'9F26') is included in the *Response Message Template* (tag '77'). If this is not the case, then the terminal shall terminate the transaction.

4.17.1.7 If combined DDA/AC generation was requested and the card did not generate an AAC, then the terminal shall verify if the *Signed Dynamic Application Data* (tag '9F4B') is included in the *Response Message Template* (tag '77'). If this is not the case, then the terminal shall decline the transaction.

4.17.1.8 If combined DDA/AC generation was requested and the card did not generate an AAC, then the terminal shall construct the *ICC Public Key* and verify if the *Signed Dynamic Application Data* is correct as specified in *[EMV BOOK 2]*.

4.17.1.9 The terminal shall verify in the *Cryptogram Information Data* if the card generated an ARQC.
If this is the case, then an online capable terminal shall complete the transaction online. If the transaction is accepted online, then the online capable terminal shall use the ARQC as transaction certificate for the clearing process.
If the card generated an ARQC, then an offline-only terminal shall decline the transaction offline.

5 Card Interoperability Requirements

5.1 Transmission Protocol

5.1.1.1 A MasterCard *PayPass* card shall be compliant with *[PAYPASS ISO/IEC 14443]*.

5.2 DOL Handling

A card may include Data Object Lists in the response messages to the terminal.

5.2.1.1 The card shall accept that the terminal fills the requested data fields with hexadecimal zeroes.

5.2.1.2 If the card returns a *PDOL*, then it shall include only tags belonging to data objects having the terminal as source.

5.2.1.3 If the card returns a *UDOL*, then it shall always include the *Unpredictable Number (Numeric)* (tag '9F6A', 4 bytes, numeric format) entry in the *UDOL*.

5.2.1.4 If the card supports combined DDA/AC generation, then it shall always include the *Unpredictable Number* (tag '9F37', 4 bytes, binary format) entry in the *CDOL1*.

5.3 Exception Processing

5.3.1.1 Whenever the card generates status bytes different from '9000' or '6283', it shall return to the state in which it needs a GET PROCESSING OPTIONS command to perform a new transaction.

5.4 *Application Transaction Counter (ATC)*

The card shall maintain an *Application Transaction Counter (ATC)*.

5.4.1.1 The *ATC* shall be binary coded on two bytes.

5.4.1.2 The card shall increment the *ATC* by 1 when it receives the GET PROCESSING OPTIONS command.

5.4.1.3 When the *ATC* reaches the value 'FFFF' the *PayPass* application shall be disabled. In this case the card shall return SW1-SW2 = '6985' in response to the GET PROCESSING OPTIONS command.

5.5 SELECT PPSE Command Processing

This section lists the requirements for the card related to the support of the PPSE.

5.5.1.1 All cards shall support the *PayPass* Payment System Environment (PPSE) directory with a file name of "2PAY.SYS.DDF01".

5.5.1.2 If the card receives a SELECT PPSE command formatted as specified in Section 1.1.1 of PART II, then it shall respond with an *FCI* indicating all the *PayPass* applications supported by the card in the *FCI Issuer Discretionary Data* as specified in Section 1.1.2 of PART II.

5.5.1.3 If the card is blocked, then the card shall return to the terminal the status bytes '6A81' (Function not supported).

5.5.1.4 If the PPSE is blocked, then the card shall return to the terminal the status bytes '6283' (Selected file invalidated).

5.6 SELECT AID Command Processing

This section specifies the behavior of the *PayPass* application in the case where a SELECT command, with the *AID* of the application, is received.

5.6.1.1 If the card receives a SELECT command message, then the card shall verify if the command message is correctly formatted as specified in Section 2.6.2 of PART II.

5.6.1.2 If the card is blocked, then the card shall return to the terminal the status bytes '6A81' (Function not supported).

5.6.1.3 If the selected application is blocked, then the card shall return to the terminal the status bytes '6283' (Selected file invalidated).

5.6.1.4 The card shall return in the response message the *FCI* as specified in Section 2.6.3 of PART II.

5.7 GET PROCESSING OPTIONS Command Processing

5.7.1.1 If the card receives a GET PROCESSING OPTIONS command message, then the card shall verify if the command message is correctly formatted as specified in Section 2.4.2 of PART II.

5.7.1.2 If the card did not provide a *PDOL*, then the card shall verify if the data field contains the value '8300'. If this is not the case, then the card shall return the status bytes '6985' (Conditions of use not satisfied)

5.7.1.3 If the card provided a *PDOL*, then the card shall verify if the length of the *Command Template* (tag '83') included in the command message data field is equal to the length requested in the *PDOL*. If this is not the case, then the card shall return the status bytes '6700' (Wrong length).

5.7.1.4 The card shall update the *ATC* as specified in Section 5.4 of PART II.

5.7.1.5 The card shall return in the response message the *Response Message Template* (tag '77') as specified in Section 2.4.3 of PART II.

5.8 READ RECORD Command Processing

5.8.1.1 If the card receives a READ RECORD command message, then the card shall verify if the command message is correctly formatted, as specified in Section 2.5.2 of PART II.

5.8.1.2 The card shall support the READ RECORD command with P1-P2 coded as specified in Section 2.5.2 of PART II. The card may support other coding formats for P1-P2.

5.8.1.3 All records referenced in the *AFL* shall be readable without conditions by the READ RECORD command coded as specified in Section 2.5.2 of PART II.

5.8.1.4 For all records referenced in the *AFL* the response message shall follow the coding as specified in Section 2.5.3 of PART II.

5.8.1.5 Proprietary data files may or may not conform to this specification. Records in proprietary files may be represented in the *AFL* if they are readable by the READ RECORD command coded according to Section 2.5.2 of PART II and if the response message follows the coding as specified in Section 2.5.3 of PART II.

5.9 COMPUTE CRYPTOGRAPHIC CHECKSUM Command Processing

5.9.1.1 A *PayPass* – M/Chip card shall support the COMPUTE CRYPTOGRAPHIC CHECKSUM command to allow the card to be used on a *PayPass* – Mag Stripe terminal.

5.9.1.2 If the card receives a COMPUTE CRYPTOGRAPHIC CHECKSUM command message, then the card shall verify if the command message is correctly formatted as specified in 2.2.2 of PART II.

5.9.1.3 The card shall verify if this is the first time the COMPUTE CRYPTOGRAPHIC CHECKSUM command is received after a successful GET PROCESSING OPTIONS command. If this is not the case, then the card shall return the status bytes '6985' (Conditions of use not satisfied).

5.9.1.4 If the card did not provide a *UDOL*, then the card shall verify if the length of the command message data field is 4 bytes. If this is not the case, then the card shall return the status bytes '6700' (Wrong length).

5.9.1.5 If the card provided to the terminal a *UDOL*, then the card shall verify if the length of the command message data field is equal to the length requested in the *UDOL*. If this is not the case, then the card shall return the status bytes '6700' (Wrong length).

5.9.1.6 The card shall return in the response message the *Response Message Template* (tag '77') as specified in Section 2.2.3 of PART II.

5.9.1.7 The card shall include in the *Response Message Template* the $CVC3_{TRACK2}$ (tag '9F61') and the *ATC* (tag '9F36').

5.9.1.8 If the card provided the *Track 1 Data* to the terminal in the response to the READ RECORD command, then the card shall include the $CVC3_{TRACK1}$ (tag '9F60') in the *Response Message Template* (tag '77').

5.10 GENERATE AC Command Processing

5.10.1.1 If the card receives a GENERATE AC command message, then the card shall verify if the command message is correctly formatted as specified in 2.3.2 of PART II.

5.10.1.2 The card shall permit at most two GENERATE AC commands after a successful GET PROCESSING OPTIONS command. If the terminal issues more than two GENERATE AC commands, then the card shall return the status bytes '6985' (Conditions of use not satisfied) for the third and all subsequent GENERATE AC commands.

5.10.1.3 The card shall verify if the length of the command message data field is equal to the length requested in the *CDOL*. If this is not the case, then the card shall return the status bytes '6700' (Wrong length).

5.10.1.4 If the GENERATE AC command message requests the generation of a TC, then the card shall reply with a TC, ARQC or AAC.

5.10.1.5 If the GENERATE AC command message requests the generation of an ARQC, then the card shall reply with an ARQC or AAC.

5.10.1.6 If the GENERATE AC command message requests the generation of an AAC, then the card shall reply with an AAC.

5.10.1.7 The card shall never return an AAR.

5.10.1.8 The card risk management performed during the processing of the GENERATE AC command, shall not rely on the content of the *Data Authentication Code* (tag '9F45'). The *Data Authentication Code* is retrieved by the terminal from the *Signed Static Application Data* during the static data authentication process. As the static data authentication process may be performed after the GENERATE AC command is issued, the terminal may fill the corresponding field with hexadecimal zeroes if the *Data Authentication Code* is requested by the card in the *CDOL1*.

Note **This requirement is not relevant for the *ICC Dynamic Number* because in the case of dynamic data authentication, only combined DDA/AC generation is performed. In this case the *ICC Dynamic Number* is retrieved from the *Signed Dynamic Application Data* returned in the response message of the GENERATE AC command.**

5.10.1.9 A card that generates an ARQC as a result of the card risk management performed during the processing of the 1st GENERATE AC command, shall leave the internal state of the card in such a way that no 2nd GENERATE AC is required to ensure the proper working of the card for the proceeding *PayPass* transactions.

5.10.1.10 The card shall return in the response message the *Response Message Template* (tag '77') as specified in Section 2.3.3 of PART II.

5.11 VERIFY Command Processing

5.11.1.1 The card shall not support the VERIFY command. The card shall return the status bytes '6D00' (Instruction code not supported) whenever the VERIFY command is received.

Note **Requirement 5.11.1.1 is independent of the personalization value of the *CVM List.***

5.12 Offline Data Authentication

5.12.1.1 The card shall support static data authentication.

5.12.1.2 If the card supports dynamic data authentication, then it shall support combined DDA/AC generation.

5.13 Card Personalization Requirements

This section lists the specific personalization requirements for a *PayPass* – M/Chip card.

5.13.1 File Organization

All data elements returned by the card during the read application data process have to be organized in a way as specified in this section. This allows the terminal to retrieve the data elements from the card with a minimum number of READ RECORD commands.

5.13.1.1 The file with SFI 1 shall contain the data objects necessary to perform a *PayPass* – Mag Stripe transaction. This record includes at least the *PayPass* – Mag Stripe mandatory data objects as specified in Table 29.

Table 29—SFI 1 – Record 1

Tag	Description	Length
'9F6B'	*Track 2 Data*	var. up to 19
'9F66'	*Track 2 Bit Map for UN and ATC (PUNATC$_{TRACK2}$)*	2
'9F65'	*Track 2 Bit Map for CVC3* (*PCVC3$_{TRACK2}$*)	2
'9F67'	*Track 2 Number of ATC Digits* (*NATC$_{TRACK2}$*)	1

5.13.1.2 The file with SFI 2 shall have only one record. This record includes at least the EMV mandatory data objects as specified in Table 30. If other data objects that are not included in the files with SFI 3 and SFI 4 are returned by the card to the terminal, then they also have to be included in record 1 of SFI 2. Record 1 of SFI 2 is the only record to be used as input for the generation of the *Signed Static Application Data*.

Table 30—SFI 2 – Record 1

Tag	Description	Length
‘5F24’	*Application Expiry Date*	3
‘5A’	*Application Primary Account Number (PAN)*	var. up to 10
‘8C’	*CDOL1*	var
‘8D’	*CDOL2*	var

5.13.1.3 The data objects listed in Table 31 and Table 32 shall be included in the first and second record of the file with SFI 3. These records include the data objects required to retrieve the *Issuer Public Key* and to perform static data authentication.

Table 31—SFI 3 – Record 1

Tag	Description	Length
‘9F4A’	*SDA Tag List*	var. up to 1
‘8F’	*Certification Authority Public Key Index*	1
‘9F32’	*Issuer Public Key Exponent*	var. up to 3
‘92’	*Issuer Public Key Remainder*	N_{CA}-N_I+36
‘90’	*Issuer Public Key Certificate*	N_{CA}

Table 32—SFI 3 – Record 2

Tag	Description	Length
‘93’	*Signed Static Application Data*	N_I

5.13.1.4 If the card supports combined DDA/AC generation, then the data objects listed in Table 33 and Table 34 shall be included in the first and second record of the file with SFI 4. These records include the data objects required to retrieve the *ICC Public Key*.

Table 33—SFI 4 – Record 1

Tag	Description	Length
‘9F47’	*ICC Public Key Exponent*	var. up to 3
‘9F48’	*ICC Public Key Remainder*	N_{IC}-N_I+42

Table 34—SFI 4 – Record 2

Tag	Description	Length
‘9F46’	*ICC Public Key Certificate*	N_I

5.13.2 AFL

The requirements of Section 5.13.1 of PART II result in an *AFL* that is coded as follows:

5.13.2.1 For a card that supports only static data authentication, the *AFL* shall be personalized with the value:
'08 01 01 00 10 01 01 01 18 01 02 00'

5.13.2.2 For a card that supports combined DDA/AC generation, the *AFL* shall be personalized with the value:
'08 01 01 00 10 01 01 01 18 01 02 00 20 01 02 00'

5.13.3 *AIP*

The card must indicate in the *AIP* that it does not support dynamic data authentication. A card being capable of supporting dynamic data authentication must only indicate in the *AIP* that it supports combined DDA/AC generation.

5.13.3.1 Bit 6 of byte 1 of the *AIP* (Offline dynamic data authentication is supported) shall be set to 0b.

6 Data Objects

The data objects that may be used for application selection and financial transaction interchange for EMV transactions are listed in *[EMV BOOK 1]* and *[EMV BOOK 3]*. This chapter contains only the extensions to these data objects and the definition of the additional data elements that are specific for *PayPass* functionality.

6.1 Data Object Format

Data elements moved from the card to the terminal are encapsulated in TLV encoded data objects. Data elements moved from the terminal to the card are identified by a Data Object List (DOL) sent to the terminal by the card or by the definition of the command message.

When data is moved from one entity to another (for example card to terminal), it shall always be passed in decreasing order, regardless of how it is stored internally. The leftmost byte (byte 1) is the most significant byte (MSB). The same rule applies when concatenating data.

Data elements that have the numeric (n) format are BCD encoded, right justified with leading hexadecimal zeroes. Data objects that have the compressed numeric (cn) format are BCD encoded, left justified and padded with trailing 'F's. Note that the length indicator in the numeric format notation (e.g. n 4) specifies the number of digits and not the number of bytes.

Data objects that have the alphanumeric (an) or alphanumeric special (ans) format are ASCII encoded, left justified and padded with trailing hexadecimal zeroes.

Bytes or bits specified as Reserved for Future Use (RFU) shall be set to the value indicated, or to zero if no value is given. An entity receiving data specified as RFU must not examine or depend upon the coding of these bytes or bits.

Card data objects, specified as optional, do not have to be supported by the card. However, the terminal must support all data objects listed in this chapter.

6.2 *Application Interchange Profile* (*AIP*)

Tag: '82'

Source: Card

Presence: Mandatory

Format: b, 2 bytes

Description: The *AIP* indicates the capabilities of the card to support specific functions in the application. The *AIP* is returned in the response message of the GET PROCESSING OPTIONS and is defined in *[EMV BOOK 3]*. This specification extends the definition by allocating the RFU bit b8 in byte 2 to indicate the *PayPass* profile (M/Chip profile or Mag Stripe profile). Table 35 specifies byte 2 of the *Application Interchange Profile* for *PayPass* transactions. Byte 1 is coded as specified in Annex C.1 of *[EMV BOOK 3]*.

Table 35—Byte 2 of the *Application Interchange Profile*

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x								*PayPass* profile
1								M/Chip profile is supported
0								Only Mag Stripe profile supported
	0	0	0	0	0	0	0	RFU

6.3 *CVC3$_{TRACK1}$*

Tag: '9F60'

Source: Card

Presence: Conditional (If the *Track 1 Data* is present, then also the *CVC3$_{TRACK1}$* must be available).

Format: b, 2 bytes

Description: The *CVC3$_{TRACK1}$* is a 2 byte cryptogram returned by the card in the response to the COMPUTE CRYPTOGRAPHIC CHECKSUM command.

6.4 *CVC3$_{TRACK2}$*

Tag: '9F61'

Source: Card

Presence: Mandatory

Format: b, 2 bytes

Description: The *CVC3$_{TRACK2}$* is a 2 byte cryptogram returned by the card in the response to the COMPUTE CRYPTOGRAPHIC CHECKSUM command.

6.5 *Default Terminal UDOL*

Tag:	--
Source:	Terminal
Presence:	Mandatory
Format:	b, 3 bytes
Description:	The *Default Terminal UDOL* is the *UDOL* to be used for constructing the value field of the COMPUTE CRYPTOGRAPHIC CHECKSUM command if the *UDOL* in the card is not present. The *Default Terminal UDOL* must always be present and must contain as its only entry the tag and length of the *Unpredictable Number (Numeric)*. The value of the *Default Terminal UDOL* is therefore: '9F 6A 04'.

6.6 *Mag Stripe Application Version Number (Card)*

Tag:	'9F6C'
Source:	Card
Presence:	Optional
Format:	b, 2 bytes
Description:	The *Mag Stripe Application Version Number (Card)* is the version number assigned by the payment system for the specific *PayPass* – Mag Stripe functionality in the card. If present, then the *Application Version Number (Card)* must be present in the file read during Read Mag Stripe Application Data.

6.7 *Mag Stripe Application Version Number (Terminal)*

Tag:	'9F6D'
Source:	Terminal
Presence:	Mandatory
Format:	b, 2 bytes
Description:	The *Mag Stripe Application Version Number (Terminal)* is the version number assigned by the payment system for the specific *PayPass* – Mag Stripe functionality in the terminal.

6.8 *Mag Stripe CVM List*

Tag: '9F68'

Source: Card

Presence: Optional

Format: b, var up to 252 bytes

Description: The *Mag Stripe CVM List* identifies a method of verification of the cardholder supported by the card for a *PayPass* – Mag Stripe transaction. The *Mag Stripe CVM List* is coded as specified in *[EMV BOOK 3]*. For the *PayPass* – Mag Stripe transaction the supported CVM codes and conditions are limited to those listed in Table 36 and Table 37.

Table 36—CVM Codes for *PayPass* – Mag Stripe

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0								RFU
	0							Fail cardholder verification if this CVM is unsuccessful
	1							Apply succeeding CVR if this CVM is unsuccessful
		0	0	0	0	0	0	Fail CVM processing
		0	0	0	0	1	0	Enciphered PIN verified online
		0	1	1	1	1	0	Signature (paper)
		0	1	1	1	1	1	No CVM required
		0	All other values					RFU
		1	0	x	x	x	x	Reserved for use by the individual payment systems
		1	1	x	x	x	x	Values in the range 110000b – 111110b are reserved for use by the issuer
		1	1	1	1	1	1	Not available for use

Table 37—CVM Condition Codes for *PayPass* – Mag Stripe

Value	Meaning
'00'	Always
'01'	If cash or cashback
'02'	If not cash or cashback
'03'	If terminal supports the CVM. (In the case of online PIN, this means 'If online PIN pad present')
'04' – '7F'	RFU
'80' – 'FF'	Reserved for use by individual payment systems

6.9 *Track 1 Bit Map for CVC3 (PCVC3$_{TRACK1}$)*

Tag: '9F62'

Source: Card

Presence: Conditional (If the *Track 1 Data* is present, then also *PCVC3$_{TRACK1}$* must be present).

Format: b, 6 bytes

Description: *PCVC3$_{TRACK1}$* indicates to the terminal the positions in the discretionary data field of the *Track 1 Data* where the q_{TRACK1} *CVC3$_{TRACK1}$* digits have to be copied. If present, then the *PCVC3$_{TRACK1}$* must be present in the file read during Read Mag Stripe Application Data.

6.10 *Track 1 Bit Map for UN and ATC (PUNATC$_{TRACK1}$)*

Tag: '9F63'

Source: Card

Presence: Conditional (If the *Track 1 Data* is present, then also *PUNATC$_{TRACK1}$* must be present).

Format: b, 6 bytes

Description: *PUNATC$_{TRACK1}$* indicates to the terminal the positions in the discretionary data field of the *Track 1 Data* where the n_{UN} *UN* digits and t_{TRACK1} *ATC* digits have to be copied. If present, then the *PUNATC$_{TRACK1}$* must be present in the file read during Read Mag Stripe Application Data.

6.11 *Track 1 Data*

Tag: '56'

Source: Card

Presence: Optional

Format: ans, variable length up to 76 bytes
The *Track 1 Data* contains the data elements of the track 1 according to ISO/IEC 7813 Structure B, excluding start sentinel, end sentinel and LRC.

Format Code (hex '42' (B))	1 byte
Identification Number (PAN)	var. up to 19 bytes
Field Separator (hex. '5E' (^))	1 byte
Name (see *ISO/IEC 7813*)	2 to 26 bytes
Field Separator (hex. '5E' (^))	1 byte
Expiry Date (YYMM)	4 bytes
Service Code	3 bytes
Discretionary Data	balance of available bytes

Description: The *Track 1 Data* may be present in the file read during Read Mag Stripe Application Data. It may be used by the terminal for authorization and clearing.

6.12 *Track 1 Number of ATC Digits ($NATC_{TRACK1}$)*

Tag: '9F64'

Source: Card

Presence: Conditional (If the *Track 1 Data* is present, then also $NATC_{TRACK1}$ must be present).

Format: b, 1 byte

Description: The value of $NATC_{TRACK1}$ represents the number of digits of the *ATC* to be included in the discretionary data field of the *Track 1 Data*. If present, then the $NATC_{TRACK1}$ must be present in the file read during Read Mag Stripe Application Data.

6.13 *Track 2 Bit Map for CVC3 ($PCVC3_{TRACK2}$)*

Tag: '9F65'

Source: Card

Presence: Mandatory

Format: b, 2 bytes

Description: $PCVC3_{TRACK2}$ indicates to the terminal the positions in the discretionary data field of the *Track 2 Data* where the q_{TRACK2} $CVC3_{TRACK2}$ digits have to be copied. The $PCVC3_{TRACK2}$ must be present in the file read during Read Mag Stripe Application Data.

6.14 *Track 2 Bit Map for UN and ATC ($PUNATC_{TRACK2}$)*

Tag: '9F66'

Source: Card

Presence: Mandatory

Format: b, 2 bytes

Description: $PUNATC_{TRACK2}$ indicates to the terminal the positions in the discretionary data field of the *Track 2 Data* where the n_{UN} *UN* digits and t_{TRACK2} *ATC* digits have to be copied. The $PUNATC_{TRACK2}$ must be present in the file read during Read Mag Stripe Application Data.

6.15 *Track 2 Data*

Tag: '9F6B'

Source: Card

Presence: Mandatory

Format: b, variable length up to 19 bytes

The *Track 2 Data* contains the data elements of the track 2 according to *[ISO/IEC 7813]*, excluding start sentinel, end sentinel and LRC, as follows:

Identification Number (PAN)	n, var. up to 19 digits
Field Separator (hex. 'D')	b
Expiry Date (YYMM)	n 4
Service Code	n 3
Discretionary Data	n, balance of available digits

Padded with hex. 'F' to ensure whole bytes.

Description: The *Track 2 Data* must be present in the file read during Read Mag Stripe Application Data. It may be used by the terminal for authorization and clearing.

6.16 *Track 2 Number of ATC Digits* (*$NATC_{TRACK2}$*)

Tag: '9F67'

Source: Card

Presence: Mandatory

Format: b, 1 byte

Description: The value of *$NATC_{TRACK2}$* represents the number of digits of the *ATC* to be included in the discretionary data field of the *Track 2 Data*. The *$NATC_{TRACK2}$* must be present in the file read during Read Mag Stripe Application Data.

6.17 *Unpredictable Number Data Object List* (*UDOL*)

Tag: '9F69'

Source: Card

Presence: Optional

Format: b, variable length

Description: The *UDOL* is the Data Object List that specifies the data objects to be included in the data field of the COMPUTE CRYPTOGRAPHIC CHECKSUM command. The *UDOL* must at least include the *Unpredictable Number (Numeric)*. The *UDOL* is not mandatory for the card. There will always be a *Default Terminal UDOL*, including as its only entry the tag and length of the *Unpredictable Number (Numeric)* (tag '9F6A'). If the card has its own *UDOL*, then it must be present in the file read during Read Mag Stripe Application Data.

6.18 *Unpredictable Number (Numeric)*

Tag:	'9F6A'
Source:	Terminal
Presence:	Mandatory
Format:	n 8 (the 8-n_{UN} most significant digits must be set to zero)
Description:	Unpredictable number generated by the terminal. The *Unpredictable Number (Numeric)* is passed to the card in the data field of the COMPUTE CRYPTOGRAPHIC CHECKSUM command.

PART III – Card Specification

PART III includes the behavioral specification of the PPSE and the PayPass – M/Chip 4 dual interface card application proposed by MasterCard and conforming to the requirements listed in PART II.

1 Introduction

The *PayPass* – M/Chip card implementation specification aims to provide a definition of the behavior of a dual interface card containing the *PayPass* – M/Chip 4 dual interface application with support for the PPSE. This document is generic in that it does not intend to include or exclude any particular platform.

This specification views support for the PPSE as another application on the card. As a consequence the *PayPass* – M/Chip card implementation specification contains the description of two applications: the PPSE application and the *PayPass* – M/Chip 4 dual interface application. Both applications are specified as state machines. The processing of a C-APDU is considered as a transition between states.

Note **These principles are used in order to present the application concepts. The same principles do not have to be followed in the actual implementation. However, the implementation must behave in a way that is indistinguishable from the behavior specified in this document.**

Note **This chapter uses the following terminology:**

- **M/Chip 4 contact-only application**
  The M/Chip Select 4 and M/Chip Lite 4 applications as specified in *[M/CHIP4].*
- ***PayPass* – M/Chip 4 application**
  The M/Chip Select 4 and M/Chip Lite 4 dual interface applications as specified in this document.

The following list gives an overview of the functionality of the *PayPass – M/Chip* card implementation proposed by MasterCard, conforming to the requirements listed in PART II.

- The PPSE application of the *PayPass* – M/Chip card must support the loop-back functionality. The loop-back functionality facilitates the compliance testing of the *PayPass – M/Chip* card with *[ISO/IEC 14443 PAYPASS]*.
- The *PayPass* – M/Chip 4 application is an extension of the M/Chip 4 contact-only application for implementation on a dual interface card (i.e. a card with an EMV contact interface and a *PayPass* contactless interface). Wherever the document refers to the contactless interface the term *PayPass* interface is used.
- The *PayPass* – M/Chip 4 application supports the COMPUTE CRYPTOGRAPHIC CHECKSUM command to assure acceptance on a *PayPass* – Mag Stripe only terminal.
- The *PayPass* – M/Chip 4 application does not support the *PDOL*.
- The *PayPass* – M/Chip 4 application does not support the *UDOL*.

- The *PayPass* – M/Chip 4 application always returns $CVC3_{TRACK1}$ in the response message of the COMPUTE CRYPTOGRAPHIC CHECKSUM. The $CVC3_{TRACK1}$ will only be taken into account by the terminal if the *PayPass* – M/Chip 4 application also returns the *Track 1 Data* in the response to the READ RECORD command.
- The *PayPass* – M/Chip 4 application provides support for static $CVC3_{TRACK1}$ and static $CVC3_{TRACK2}$. The use of static $CVC3_{TRACK1}$ and static $CVC3_{TRACK2}$ instead of the dynamic $CVC3_{TRACK1}$ and dynamic $CVC3_{TRACK2}$ is indicated by the *Application Control* and defined during the personalization process.
- The *PayPass* – M/Chip 4 application supports the inclusion of the *ATC* in the generation of $CVC3_{TRACK1}$ and $CVC3_{TRACK2}$. The inclusion of the *ATC* in the cryptogram generation is indicated by the *Application Control* and defined during the personalization process.

2 PPSE Application

2.1 Introduction

This section specifies the behavior of the card for the selection of the PPSE. Support for the PPSE is mandatory for all *PayPass* cards. The SELECT PPSE command processing is independent of the actual application(s) implemented on the card. The PPSE may be implemented as a separate application (applet) on a multi-application platform or may be mapped on a DF (Dedicated File) – which may or may not be the MF (Master File) – of an ISO 7816-4 compatible file structure.

In addition to the directory function, the PPSE application provides support for loop-back functionality. Loop-back functionality is implemented by the LOOP BACK C-APDU. Upon receiving a LOOP BACK C-APDU the PPSE application returns without any further action the content of the data field of the C-APDU in the data field of the R-APDU. Loop-back functionality is used during the compliance testing of the *PayPass* card with *[ISO/IEC 14443 PAYPASS]*.

2.2 Application State Machine

The behavior of the PPSE application is specified by its state machine. The application states used in this description are given in Table 38.

Table 38—Application States of the PPSE Application

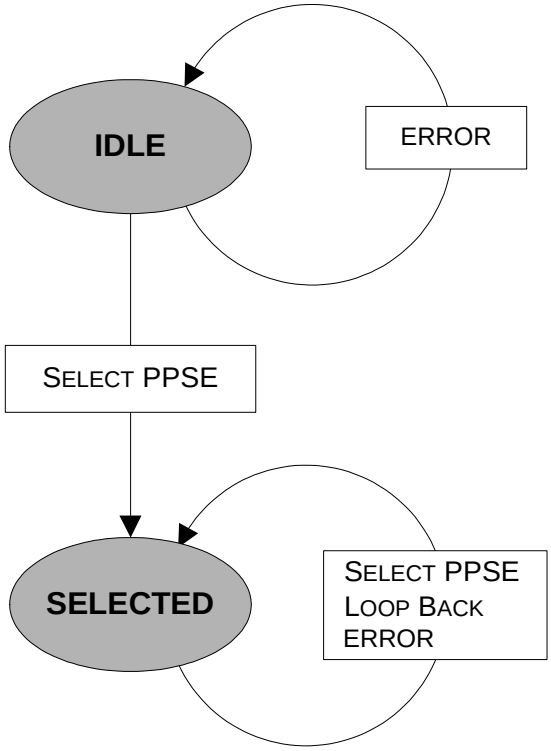
State	Description
IDLE	Application is not currently selected
SELECTED	Application is selected

The PPSE application is in state IDLE if it is not currently activated. There is only one C-APDU which is handled in this state: the SELECT PPSE C-APDU which activates the application. Upon successfully processing of the SELECT PPSE C-APDU, the PPSE application goes to the state SELECTED. The PPSE application remains in the state SELECTED until the PPSE application is de-selected (i.e. another application is selected or the card is powered-off).

The PPSE application does not change state whenever an error occurs. An error means a command response with status bytes different from ‘9000’.

Figure 7 shows the state machine of the PPSE application.

Figure 7—State Machine of PPSE Application



2.3 Command Processing

This section specifies the command processing for the PPSE application.

2.3.1 C-APDU Recognition

C-APDU recognition is specified as a procedure that identifies the C-APDU transmitted by the terminal to the PPSE application. The recognition is based on the CLA and INS byte. Table 39 specifies the CLA and INS coding for the PPSE application.

Table 39—C-APDU Recognition of the PPSE Application

CLA	INS	C-APDU
'00'	'A4'	SELECT PPSE
'80'	'EE'	LOOP BACK

If the CLA and INS byte of the C-APDU is not one of the two combinations listed in Table 39, then the C-APDU recognition procedure returns status bytes '6E00' or '6D00' and the PPSE application remains in its current state.

2.3.2 C-APDU Acceptance

C-APDU acceptance is specified as the procedure that accepts or rejects the C-APDU, depending on the application state. Acceptance or rejection of a C-APDU by the PPSE application is specified in Table 40.

Table 40—Acceptance Matrix of the PPSE Application

	IDLE	SELECTED
SELECT PPSE	Accept	Accept
LOOP BACK	Reject	Accept

In the IDLE state, the LOOP BACK C-APDU is not passed to the PPSE application, but is handled by the multi-application manager (refer to *[M/CHIP4]* for more information about the multi-application manager) or operating system. In this case, the LOOP BACK command should be rejected. Native cards that map the PPSE on the MF file, may however accept the LOOP BACK command without first selecting the PPSE. If the LOOP BACK command is rejected in the IDLE state, then the value of the status bytes is left to the implementation.

If the C-APDU is accepted in the current application state then the C-APDU is processed as specified in the section dedicated to the C-APDU.

2.3.3 SELECT PPSE

Command Message

The SELECT command message for the PPSE is coded as specified in Section 1.1.1 of PART II.

Response Message

The data field of the response message contains the *FCI* and is coded as specified in Section 1.1.2 of PART II.

Processing

Figure 8 specifies the processing of the SELECT PPSE command.

Symbol 0

If P1 ≠ ‘04’ and P2 ≠ ‘00’, then the C-APDU is rejected (SW1-SW2 = ‘6A86’).

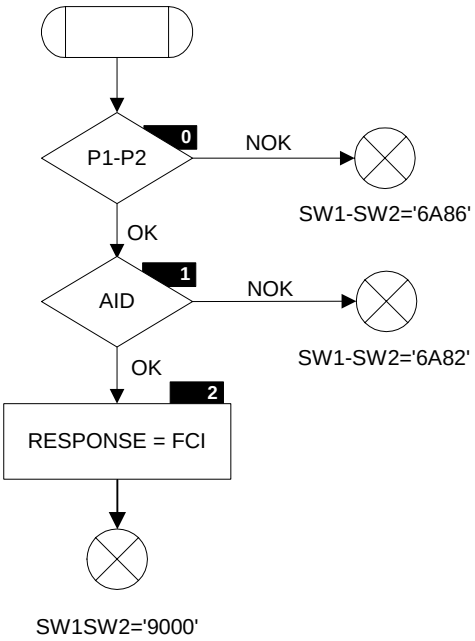
Symbol 1

If the *AID* in the command message data field is different from the PPSE directory name (“2PAY.SYS.DDF01”), then the C-APDU is rejected (SW1-SW2 = ‘6A82’).

Symbol 2

Build the response message as specified in Section 1.1.2 of Part II.

Figure 8—SELECT PPSE processing



Destination States

The destination states for the SELECT PPSE command are listed in Table 41.

Table 41—Destination States for SELECT PPSE Command

SW1	SW2	IDLE	SELECTED
'6A'	'82'	IDLE	SELECTED
'6A'	'86'	IDLE	SELECTED
'90'	'00'	SELECTED	SELECTED
	Other	IDLE	SELECTED

2.3.4 LOOP BACK

The LOOP BACK command returns without any further action the content of the data field of the C-APDU in the data field of the R-APDU.

Command Message

The LOOP BACK command message is coded according to Table 42.

Table 42—LOOP BACK Command Message

Code	Value
CLA	’80’
INS	’EE’
P1	‘00’
P2	‘00’
Lc	var
Data	Test Data
Le	‘00’

The value of Lc defines the number of bytes included in the Test Data. The LOOP BACK command must work for Lc ranging from 1 to 250 and may optionally work for Lc greater than 250. The data field of the command message contains the Test Data to be returned in the data field of the response message.

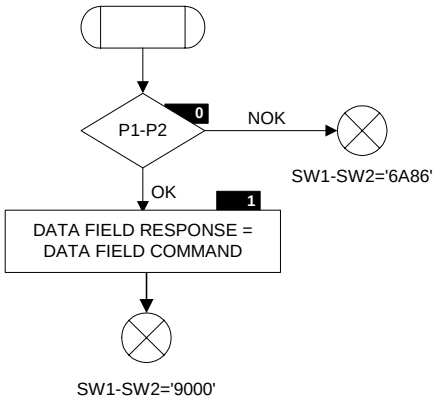
Response Message

The data field of the response message contains the Test Data included in the data field of the command message.

Processing

Figure 9 specifies the processing of the LOOP BACK command.

Figure 9—LOOP BACK Processing



Symbol 0

If P1 ≠ ‘00’ or P2 ≠ ‘00’, then the C-APDU is rejected (SW1-SW2 = ‘6A86’).

Symbol 1

Build the data field of the response message. The data field of the response is set equal to the data field of the command message.

Destination States

The destination states for the LOOP BACK command are listed in Table 43.

Table 43—Destination States for LOOP BACK Command

SW1	SW2	SELECTED
‘6A’	‘86’	SELECTED
‘90’	‘00’	SELECTED
Other		SELECTED

3 PayPass – M/Chip 4 Application

3.1 Introduction

This document specifies the behavior of the M/Chip Select 4 and M/Chip Lite 4 applications when implemented on a dual interface card.

3.1.1 Assumptions

In this specification we make the following assumptions about the use of a dual interface card:

- Only one of the two interfaces is used between the power-on and power-off of the card.
- It is possible to know on the application layer whether the card is communicating via the contact or the *PayPass* interface.

3.1.2 Data Elements

The *PayPass* – M/Chip 4 application and the M/Chip 4 contact-only application support a different set of data elements. In addition to the *PayPass* – Mag Stripe specific data elements, the *PayPass* – M/Chip 4 application also supports six new instances of existing M/Chip 4 contact-only application data elements: *Application Interchange Profile (PayPass)*, *Application File Locator (PayPass)*, *Application Control (PayPass)* and *Card Issuer Action Codes (PayPass)*. These data elements can not be shared between the contact and *PayPass* interface and need to be personalized with a specific value for the *PayPass* interface. All other existing M/Chip 4 contact-only application data elements are shared between the contact and the *PayPass* interface.

3.1.3 Offline Counters

The *PayPass* – M/Chip 4 application shares the offline risk management counters between the contact and the *PayPass* interface. Taking into account that during a *PayPass* transaction the card is removed after the 1st GENERATE AC command, then the offline counters will only be updated if the transaction is accepted offline. The counters remain unchanged if a *PayPass* transaction is completed online. The counters can only be reset during an online contact transaction.

3.1.4 Log of Transactions

The *PayPass* – M/Chip 4 application stores transaction information in the Log of Transactions whenever a TC is generated. Therefore, the issuer should notice that online accepted *PayPass* transactions may not appear in the Log of Transactions. This is the case when the *PayPass* – M/Chip 4 application generates an ARQC and the terminal does not send a second GENERATE AC command.

3.2 Application State Machine

When the application is in operational phase (i.e. personalized), its behavior can be specified as a state machine.

The application states used in this description are the same as the states defined for the M/Chip 4 contact-only application and are listed in Table 44.

Table 44—Application states of the *PayPass* – M/Chip 4 Application

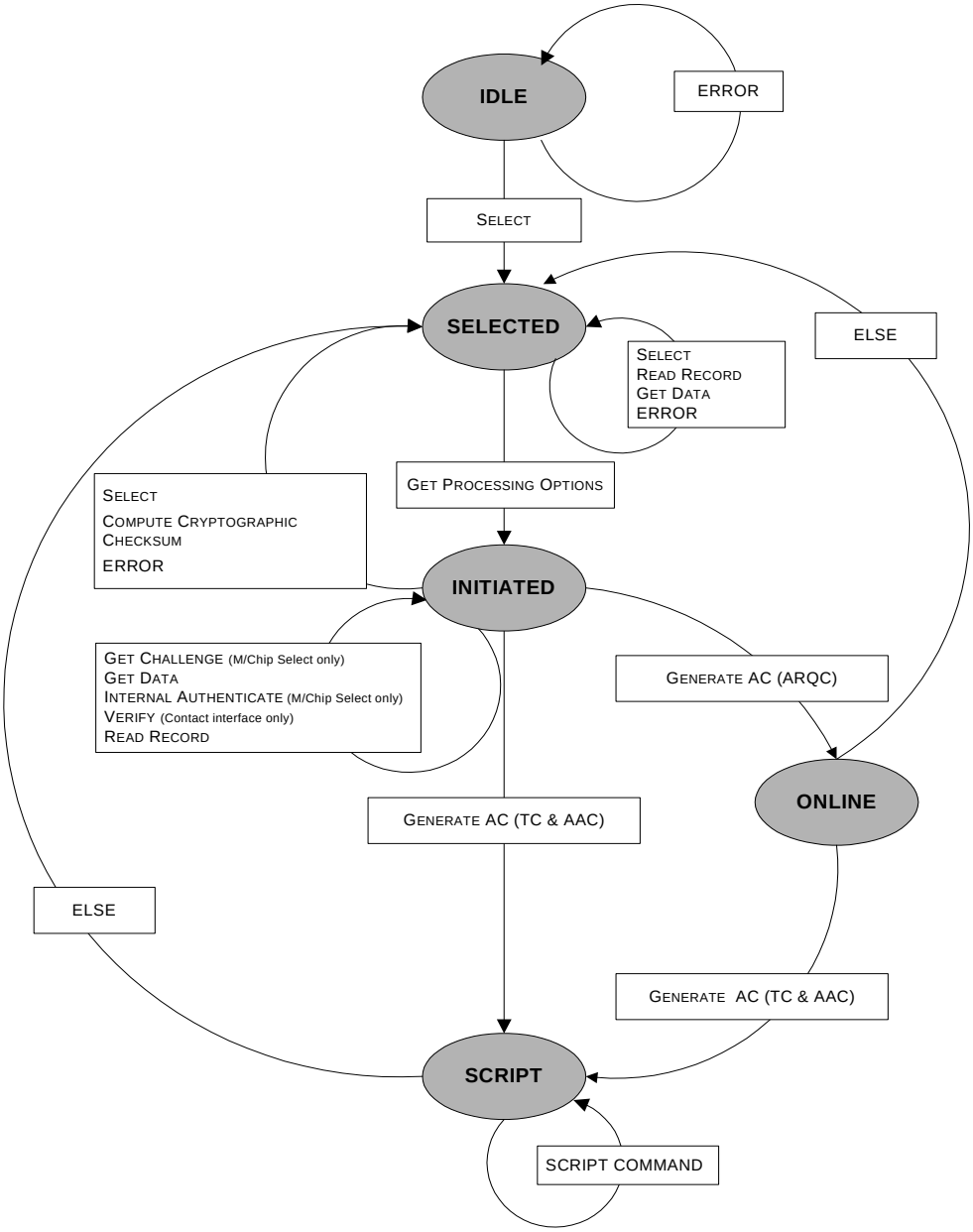
State	Description
IDLE	Application is not currently selected
SELECTED	Application is selected
INITIATED	Transaction is initiated
ONLINE	Application expects a connection with the issuer
SCRIPT	Application is ready to accept a script command

The *PayPass* – M/Chip 4 application state machine supports in addition to the state transitions supported by the M/Chip 4 contact-only application also the following state transition:

- If the *PayPass* – M/Chip 4 application is in the state INITIATED, then it goes back to the state SELECTED after the successful processing of the COMPUTE CRYPTOGRAPHIC CHECKSUM command.

Figure 10 illustrates the state machine of the *PayPass* – M/Chip 4 application.

Figure 10—State Machine of the *PayPass* – M/Chip 4 Application



3.3 C-APDU PRE-PROCESSING

3.3.1 C-APDU Recognition

C-APDU recognition is specified as a procedure that identifies the C-APDU transmitted by the terminal to the *PayPass* – M/Chip 4 application. The recognition is based on the CLA and INS byte. The *PayPass* – M/Chip 4 application supports the CLA and INS bytes specified in Table 45.

The C-APDU recognition procedure takes as input the CLA and INS bytes and produces as output one of the responses as listed in the third column of Table 45.

If the CLA byte of the C-APDU is not one of those listed in Table 45, then the C-APDU Recognition procedure returns BAD CLA. If the INS byte of the C-APDU is not one of those listed in Table 45, then the C-APDU Recognition procedure returns BAD INS.

Table 45—C-APDU Recognition

CLA	INS	C-APDU
'84'	'1E'	APPLICATION BLOCK
'84'	'18'	APPLICATION UNBLOCK
'80'	'2A'	COMPUTE CRYPTOGRAPHIC CHECKSUM
'80'	'AE'	GENERATE AC
'00'	'84'	GET CHALLENGE[b]
'80'	'CA'	GET DATA
'80'	'A8'	GET PROCESSING OPTIONS
'00'	'88'	INTERNAL AUTHENTICATE[b]
'84'	'24'	PIN CHANGE/UNBLOCK
'84'	'DA'	PUT DATA
'00'	'B2'	READ RECORD
'00'	'A4'	SELECT
'00'	'20'	VERIFY[a]
'84'	'DC'	UPDATE RECORD

a. Only applicable for the contact interface. If the C-APDU is received via the *PayPass* interface, then the C-APDU Recognition must return BAD INS.

b. Only applicable for M/Chip 4 Select.

When the application has recognized the C-APDU it must perform a validity check on the following:

- Consistency between Lc and the length of data sent
- Le

These checks are protocol dependent and can not be specified independently from the transport layer. However, when the validity check detects an error on the lengths, the output of the procedure C-APDU Recognition is BAD LENGTH.

If the output of the C-APDU Recognition is BAD CLA, BAD INS or BAD LENGTH, then the C-APDU is not supported by the *PayPass* – M/Chip 4 application over the active interface.

3.3.2 C-APDU Acceptance

C-APDU acceptance is specified as the procedure that accepts or rejects the C-APDU in function of the application state. Acceptance or rejection of a C-APDU by the *PayPass* – M/Chip 4 application is specified in Table 46. The C-APDU acceptance procedure takes as input the response of the C-APDU recognition procedure.

If the C-APDU is accepted in the current application state (P: processed), then the C-APDU is processed as specified in Section 3.4 of PART III.

If the C-APDU is rejected in the current state (R/CNS: rejected, conditions of use not satisfied), then the processing is specified in the section Rejected C-APDU processing.

Table 46—Acceptance Matrix

	SELECTED	INITIATED	ONLINE	SCRIPT
APPLICATION BLOCK	R/CNS	R/CNS	R/CNS	P
APPLICATION UNBLOCK	R/CNS	R/CNS	R/CNS	P
COMPUTE CRYPTOGRAPHIC CHECKSUM	R/CNS	P	R/CNS	R/CNS
GENERATE AC	R/CNS	P	P	R/CNS
GET CHALLENGE	R/CNS	P	R/CNS	R/CNS
GET DATA	P	P	R/CNS	R/CNS
GET PROCESSING OPTIONS	P	R/CNS	R/CNS	R/CNS
INTERNAL AUTHENTICATE	R/CNS	P	R/CNS	R/CNS
PIN CHANGE/UNBLOCK	R/CNS	R/CNS	R/CNS	P
PUT DATA	R/CNS	R/CNS	R/CNS	P
READ RECORD	P	P	R/CNS	R/CNS
SELECT	P	P	P	P
VERIFY	R/CNS	P	R/CNS	R/CNS
UPDATE RECORD	R/CNS	R/CNS	R/CNS	P

3.3.3 Rejected C-APDU Processing

Two reasons may lead to C-APDU rejection:

- The bytes received are not recognized as a supported C-APDU (i.e. the couple (CLA,INS) does not correspond to a C-APDU supported by the *PayPass* – M/Chip 4 application over the current active interface or there is an error on the lengths). In this case the rejection happens in the procedure C-APDU Recognition.

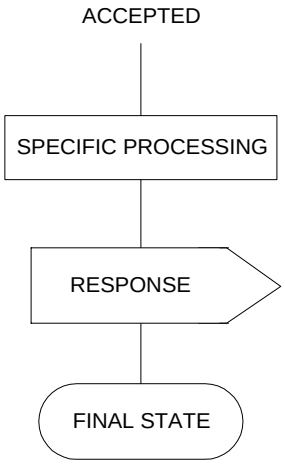
- The C-APDU is supported by the *PayPass* – M/Chip 4 application, but the application is in a state where it is not accepted. In this case rejection happens during the C-APDU Acceptance procedure.

Refer to *[M/CHIP4]* for the description of the processing of the four cases R/CNS, BAD CLA, BAD INS and BAD LENGTH.

3.4 Processing C-APDUs

Figure 11 illustrates the actions taken by the *PayPass* – M/Chip 4 application when a C-APDU is processed.

Figure 11—Processing a C-APDU



A C-APDU is processed if the C-APDU acceptance procedure has determined that the application state is consistent with the C-APDU. The processing that is specific to the C-APDU is specified in *[M/CHIP4]* and in Section 3.5 of PART III for the COMPUTE CRYPTOGRAPHIC CHECKSUM command.

Commands that access the *AIP*, *AFL*, *Application Control* and *Card Issuer Action Codes* internal data elements must use the correct instance of the data element dependent on the active interface. This includes:

- The GENERATE AC command accessing the *Application Control* and *Card Issuer Action Codes* for the contact interface and the *Application Control (PayPass)* and *Card Issuer Action Codes (PayPass)* for the *PayPass* interface. If the *AIP* is used as input to the generation of the *Application Cryptogram*, then the *Application Interchange Profile* must be used for the contact interface and the *Application Interchange Profile (PayPass)* must be used for the *PayPass* interface.
- The GET PROCESSING OPTIONS command accessing the *Application Interchange Profile* and *Application File Locator* for the contact interface and the *Application Interchange Profile (PayPass)* and *Application File Locator (PayPass)* for the *PayPass* interface.

- The COMPUTE CRYPTOGRAPHIC CHECKSUM command accessing the *Application Control (PayPass)* for both the contact and *PayPass* interface.

The R-APDU resulting from the processing of a C-APDU is specified in the section dedicated to the C-APDU. The way the response is sent depends on the protocol and is outside the scope of this specification.

The destination state of the application when the C-APDU is processed is specified in the section dedicated to the C-APDU.

3.5 COMPUTE CRYPTOGRAPHIC CHECKSUM

3.5.1 Command Message

The COMPUTE CRYPTOGRAPHIC CHECKSUM command message is coded according to Table 47.

Table 47—COMPUTE CRYPTOGRAPHIC CHECKSUM Command Message

Code	Value
CLA	'80'
INS	'2A'
P1	'8E'
P2	'80'
Lc	'04'
Data	*Unpredictable Number (Numeric)*
Le	'00'

As the *UDOL* is not provided by the *PayPass* – M/Chip 4 application, the data field of the command message is the value field of the *Unpredictable Number (Numeric)* data object.

3.5.2 Data Field Returned in the Response Message

The data field of the response message is a constructed data object with tag '77'. The value field of the constructed data object includes the $CVC3_{TRACK1}$, the $CVC3_{TRACK2}$ and the *ATC*.

Table 48—COMPUTE CRYPTOGRAPHIC CHECKSUM Response Message

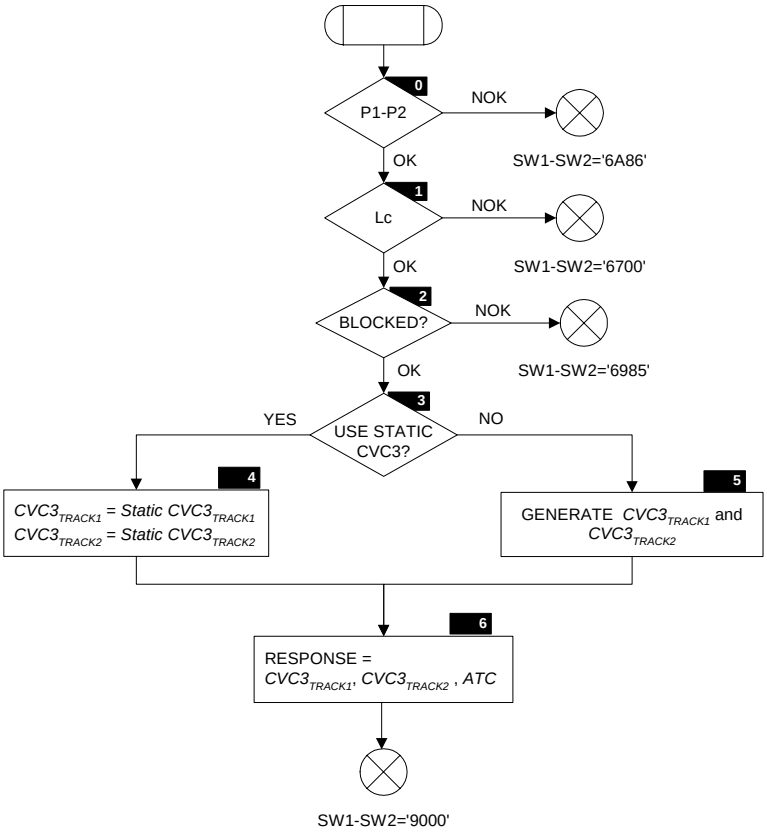
Data Element	Tag	Length
Response Message Template	'77'	15
$CVC3_{TRACK2}$	'9F61'	2
$CVC3_{TRACK1}$	'9F60'	2
ATC	'9F36'	2

The $CVC3_{TRACK2}$ and the $CVC3_{TRACK1}$ are cryptograms generated by the *PayPass* – M/Chip 4 application according the algorithm specified in Section 3.8 of PART III.

3.5.3 Processing

Figure 12 specifies the flow of the COMPUTE CRYPTOGRAPHIC CHECKSUM command processing.

Figure 12—COMPUTE CRYPTOGRAPHIC CHECKSUM Processing



Symbol 0

If P1 ≠ '8E' or P2 ≠ '80', then the C-APDU is rejected (SW1-SW2 = '6A86').

Symbol 1

If Lc ≠ 4, then the C-APDU is rejected (SW1-SW2 = '6700').

Symbol 2

If the application is blocked (i.e. if *Previous Transaction History*[5] = 1b), then the C-APDU is rejected (SW1-SW2='6985').

Symbol 3

The *PayPass* – M/Chip 4 application checks if the *Static CVC3* must be used (i.e. *Application Control(PayPass)*[3][8] = 1b).

Symbol 4
If *Static CVC3* must be used, then the *PayPass* – M/Chip 4 application sets $CVC3_{TRACK1}$ equal to *Static* $CVC3_{TRACK1}$ and $CVC3_{TRACK2}$ equal to *Static* $CVC3_{TRACK2}$.

Symbol 5
The *PayPass* – M/Chip 4 application generates $CVC3_{TRACK1}$ and $CVC3_{TRACK2}$ as specified in Section 3.8 of PART III.

Symbol 6
The *PayPass* – M/Chip 4 application generates the response message template containing the $CVC3_{TRACK1}$, the $CVC3_{TRACK2}$ and the *ATC*.

3.5.4 Destination State

The destination states for the COMPUTE CRYPTOGRAPHIC CHECKSUM command are listed in Table 49.

Table 49—Destination State for COMPUTE CRYPTOGRAPHIC CHECKSUM Command

SW1	SW2	INITIATED
'67'	'00'	SELECTED
'69'	'85'	SELECTED
'6A'	'86'	SELECTED
'90'	'00'	SELECTED
Other		SELECTED

3.6 GET DATA

The GET DATA command is processed as specified in *[M/CHIP4]*. This section specifies the additional tag value that has to be supported by the GET DATA command of the *PayPass* – M/Chip 4 application.

Table 50—Additional Tag Value for GET DATA

P1/P2	Data Element	Length
'00CD'	*Card Issuer Action Code (PayPass) – Default*	3
'00CE'	*Card Issuer Action Code (PayPass) – Online*	3
'00CF'	*Card Issuer Action Code (PayPass) – Decline*	3
'00D7'	*Application Control (PayPass)*	3

3.7 PUT DATA

The PUT DATA command is processed as specified in *[M/CHIP4]*. This section specifies the additional tag values that have to be supported by the PUT DATA command of the *PayPass* – M/Chip 4 application.

Table 51—Additional Tag Values for PUT DATA

P1/P2	Data Element	Length
'00CD'	*Card Issuer Action Code (PayPass) – Default*	3
'00CE'	*Card Issuer Action Code (PayPass) – Online*	3
'00CF'	*Card Issuer Action Code (PayPass) – Decline*	3
'00D7'	*Application Control (PayPass)*	3
'00D8'	*AIP (PayPass)*	2
'00D9'	*AFL (PayPass)*	12 or 16
'00DA'	*Static $CVC3_{TRACK1}$*	2
'00DB'	*Static $CVC3_{TRACK2}$*	2
'00DC'	*$IVCVC3_{TRACK1}$*	2
'00DD'	*$IVCVC3_{TRACK2}$*	2

3.8 Dynamic CVC3

This section specifies how the *PayPass* – M/Chip 4 application constructs the dynamic CVC3.

The *PayPass* – M/Chip 4 application generates a dynamic CVC3 for the *Track 1 Data* ($CVC3_{TRACK1}$) and a dynamic CVC3 for the *Track 2 Data* ($CVC3_{TRACK2}$). Both cryptograms are generated with the same dynamic data (*UN* and *ATC*) and with the same secret key (*ICC Derived Key for CVC3 Generation*), but with a different initialization vector ($IVCVC3_{TRACK1}$ *for* $CVC3_{TRACK1}$ and $IVCVC3_{TRACK2}$ for $CVC3_{TRACK2}$).

3.8.1 ICC Derived Key for CVC3 Generation (KD_{CVC3})

This section specifies the key derivation method used to generate the *ICC Derived Key for CVC3 Generation* (KD_{CVC3}).

KD_{CVC3} is a 16-byte DES3 key derived from the *Issuer Master Key for CVC3 Generation* (IMK_{CVC3}) as follows:

1. Concatenate from left to right the PAN (without any hex 'F' padding) with the PAN sequence number (if the PAN sequence number is not available, then it is replaced by a '00' byte). If the result X is less than 16 digits long, pad it to the left with hexadecimal zeros in order to obtain an eight-byte number Y in numeric (n) format. If X is at least 16 digits long, then Y consists of the 16 rightmost digits of X in numeric (n) format.

2. Compute the two eight-byte numbers:

   Z_L := DES3(IMK_{CVC3})[Y]
   Z_R := DES3(IMK_{CVC3})[Y ⊕ (‘FF’||‘FF’||‘FF’||‘FF’||‘FF’||‘FF’||‘FF’||‘FF’)]

   and define:

   Z := (Z_L || Z_R).

KD_{CVC3} is defined to be Z, with the exception of the least significant bit of each byte of Z which is set to a value that ensures that each of the 16 bytes of KD_{CVC3} has an odd number of nonzero bits (this is to conform with the odd parity requirements for DES keys).

3.8.2 Dynamic CVC3 Generation

The $CVC3_{TRACK1}$ is generated using DES3 encipherment as follows:

1. Concatenate the data listed in Table 52 in the order specified to obtain an 8 byte data block (D):

Table 52—Track 1 CVC3 Data Elements

Data Element	Length
$IVCVC3_{TRACK1}$	2 bytes
Unpredictable Number	4 bytes
Application Transaction Counter[a]	2 bytes

[a] If *Application Control*[3][7] = 0b (do not include the *ATC* in dynamic CVC3 generation), then the 2 bytes are filled with hexadecimal zeroes (’00 00’).

2. Calculate O as follows:

   O := DES3(KD_{CVC3})[D]

3. The two least significant bytes of O are the $CVC3_{TRACK1}$.

The $CVC3_{TRACK2}$ is generated in the same way by replacing $IVCVC3_{TRACK1}$ with $IVCVC3_{TRACK2}$.

3.8.3 *IVCVC3* Generation

The $IVCVC3_{TRACK1}$ and $IVCVC3_{TRACK2}$ are issuer proprietary static data elements that are used as input for the generation of the $CVC3_{TRACK1}$ and $CVC3_{TRACK2}$ cryptograms.

$IVCVC3_{TRACK1}$ is a MAC calculated over the static part of the *Track 1 Data* using the *ICC Derived Key for CVC3 Generation.* $IVCVC3_{TRACK2}$ is a MAC calculated over the static part of the *Track 2 Data* also using the *ICC Derived Key for CVC3 Generation*.

The MAC is generated using DES encipherment as specified in Annex A using KD_{CVC3} as the key. For the generation of $IVCVC3_{TRACK1}$ the message M consists of the static part of the *Track 1 Data*. For the generation of $IVCVC3_{TRACK2}$ the message M consists of the static part of the *Track 2 Data.* The two least significant bytes of the MAC are the $IVCVC3_{TRACK1}$ or $IVCVC3_{TRACK2}$.

3.9 Data Elements Dictionary

The *PayPass* – M/Chip 4 application supports all data elements supported by the M/Chip 4 contact-only application as listed in the Data Elements Dictionary of *[M/CHIP4]*. This section lists only the additional data elements that are supported by the *PayPass* – M/Chip 4 application and the extensions to existing M/Chip 4 contact-only data elements. This section complements the data elements listed in Chapter 6 of PART II.

3.9.1 *Application Control (PayPass)*

Tag: 'D7'

Format: b, 3 bytes

Description: The *Application Control (PayPass)* activates or de-activates functions in the application when the *PayPass* interface is used. The *PayPass* – M/Chip 4 application extends the definition of the *Application Control* of the M/Chip 4 contact-only application with 1 byte. This byte allows to activate or de-activate options for the generation of the dynamic CVC3. Table 53 shows the coding of byte 3. Byte 1 and byte 2 are coded as the *Application Control* (tag 'D5') specified in *[M/CHIP4]*.

Table 53—Byte 3 of the *Application Control (PayPass)*

b8	b7	b6	b5	b4	b3	b2	b1	Description
x								Indicate if *Static CVC3* must be used: • 0b: Do not use *Static CVC3* • 1b: Use *Static CVC3*
	x							Include *ATC* in *CVC3* generation • 0b: Do not include *ATC* • 1b: Include *ATC*
		0	0	0	0	0	0	RFU

3.9.2 *Application File Locator (PayPass)*

Tag: 'D9'

Format: b, 12 or 16 bytes

Description: The *Application File Locator (AFL) (PayPass)* indicates the location (SFI and range of records) of the AEFs when the *PayPass* interface is used.

Note **The tag 'D9' of the *AFL (PayPass)* must only be used to identify the data element for the PUT DATA command. When the *AFL (PayPass)* is returned in the response message of the GET PROCESSING OPTIONS command, then the EMV tag '94' must be used.**

3.9.3 *Application Interchange Profile (PayPass)*

Tag: 'D8'

Format: b, 2 bytes

Description: The *Application Interchange Profile (AIP) (PayPass)* indicates the capabilities of the card to support specific functions in the application when the *PayPass* interface is used. The *AIP (PayPass)* is coded as specified in Section 6.2 of PART II.

Note **The tag 'D8' of the *AIP (PayPass)* must only be used to identify the data element for the PUT DATA command. When the *AIP (PayPass)* is returned in the response message of the GET PROCESSING OPTIONS command, then the EMV tag '82' must be used.**

3.9.4 *Card Issuer Action Codes (PayPass) – Decline, Default, Online*

Tags: *Card Issuer Action Code (PayPass) – Default: 'CD'*
Card Issuer Action Code (PayPass) – Online: 'CE'
Card Issuer Action Code (PayPass) – Decline: 'CF'

Format: b, 3 bytes

Description: The *Card Issuer Action Codes (PayPass)* are represented by three *PayPass* – M/Chip 4 proprietary data elements: *Card Issuer Action Code (PayPass) – Default*, *Card Issuer Action Code (PayPass) – Online* and *Card Issuer Action Code (PayPass) – Decline*. They are compared to the decisional part of the *Card Verification Results* to decide which cryptogram to include in the response to the GENERATE AC command.

The *Card Issuer Action Codes (PayPass)* are formatted as the *Card Issuer Action Codes* (tags 'C3', 'C4' and 'C5') specified in *[M/CHIP4]*.

3.9.5 *Static CVC3$_{TRACK1}$*

Tag: 'DA'

Format: b, 2 bytes

Description: The *Static CVC3$_{TRACK1}$* is the static variant of the dynamic CVC3 of the track 1 data converted into the binary format (e.g. a *Static CVC3$_{TRACK1}$* with value "812" in ans format is stored as '032C'). The *PayPass* – M/Chip 4 application returns the *Static CVC3$_{TRACK1}$* instead of the dynamically calculated *CVC3$_{TRACK1}$* if *Application Control (PayPass)*[3][8] = 1b.

3.9.6 *Static CVC3$_{TRACK2}$*

Tag: 'DB'

Format: b, 2 bytes

Description: The *Static CVC3$_{TRACK2}$* is the static variant of the dynamic CVC3 of the track 2 data converted into the binary format (e.g. a *Static CVC3$_{TRACK2}$* with value 812 in numeric (n) format is stored as '032C'). The *PayPass* – M/Chip 4 application returns the *Static CVC3$_{TRACK2}$* instead of the dynamically calculated *CVC3$_{TRACK2}$* if *Application Control (PayPass)*[3][8] = 1b.

3.9.7 *IVCVC3$_{TRACK1}$*

Tag: 'DC'

Format: b, 2 bytes

Description: The *IVCVC3$_{TRACK1}$* is an issuer proprietary static data element that is used as input for the generation of the *CVC3$_{TRACK1}$* cryptogram. Refer to Section 3.8.3 of PART III for a detailed description of the generation of *IVCVC3$_{TRACK1}$*.

3.9.8 *IVCVC3$_{TRACK2}$*

Tag: 'DD'

Format: b, 2 bytes

Description: The *IVCVC3$_{TRACK2}$* is an issuer proprietary static data element that is used as input for the generation of the *CVC3$_{TRACK2}$* cryptogram. Refer to Section 3.8.3 of PART III for a detailed description of the generation of *IVCVC3$_{TRACK2}$*.

3.10 Data Elements Location

3.10.1 Transient Data Elements that Span a Single C-APDU Processing

Some transient data elements have a lifetime that spans a single C-APDU processing. All these transient data elements are created during application selection and are listed in *[M/CHIP4]*.

3.10.2 Additional Persistent Data Elements

All the persistent data elements that are listed in *[M/CHIP4]* are supported by the *PayPass* – M/Chip 4 application. Table 54 lists the additional persistent data elements of the *PayPass* – M/Chip 4 application and their access conditions.

Table 54—Additional Persistent Data Elements

Tag	Name	read record	update record	internal read	internal update	get data	put data
'56'	*Track 1 Data*	Yes	Yes	No	No	No	No
'9F62'	*PCVC3$_{TRACK1}$*	Yes	Yes	No	No	No	No
'9F63'	*PUNATC$_{TRACK1}$*	Yes	Yes	No	No	No	No
'9F64'	*NATC$_{TRACK1}$*	Yes	Yes	No	No	No	No
'9F65'	*PCVC3$_{TRACK2}$*	Yes	Yes	No	No	No	No
'9F66'	*PUNATC$_{TRACK2}$*	Yes	Yes	No	No	No	No
'9F67'	*NATC$_{TRACK2}$*	Yes	Yes	No	No	No	No
'9F68'	*Mag Stripe CVM List*	Yes	Yes	No	No	No	No
'9F6B'	*Track 2 Data*	Yes	Yes	No	No	No	No
'9F6C'	*Mag Stripe Application Version Number (Card)*	Yes	Yes	No	No	No	No
'CD'	*Card Issuer Action Code (PayPass) – Decline*	No	No	Yes	No	Yes	Yes
'CE'	*Card Issuer Action Code (PayPass) –Default*	No	No	Yes	No	Yes	Yes
'CF'	*Card Issuer Action Code (PayPass) – Online*	No	No	Yes	No	Yes	Yes
'D7'	*Application Control (PayPass)*	No	No	Yes	No	Yes	Yes
'D8'	*AIP (PayPass)*	No	No	Yes	No	No	Yes
'D9'	*AFL (PayPass)*	No	No	Yes	No	No	Yes
'DA'	*Static CVC3$_{TRACK1}$*	No	No	Yes	No	No	Yes
'DB'	*Static CVC3$_{TRACK2}$*	No	No	Yes	No	No	Yes
'DC'	*IVCVC3$_{TRACK1}$*	No	No	Yes	No	No	Yes
'DD'	*IVCVC3$_{TRACK2}$*	No	No	Yes	No	No	Yes

3.10.3 Secret Keys

All the secret keys that are listed in *[M/CHIP4]* are supported by the *PayPass* – M/Chip 4 application. Table 55 defines the additional Triple DES key for CVC3 generation that must be supported by the *PayPass* – M/Chip 4 application.

Table 55—Triple DES Key for CVC3 Generation

Data Element	length	get data	internal update	put data
ICC Derived Key for CVC3 Generation (KD$_{CVC3}$)	16	No	No	No

3.11 Personalization

This section specifies the data elements that are available to the issuer for personalization. The personalization commands are not in the scope of this specification. They are left to the implementation.

All data elements available for personalization are stored in persistent memory of the card and are listed in *[M/CHIP4]*. This section specifies only the specific personalization requirements for the *PayPass* – M/Chip 4 application.

Note **When the *PayPass* – M/Chip 4 application is personalized according to the EMV Card Personalization Specification as defined in *[M/CHIP4 CPS]*, then the data groupings for the *PayPass* specific data elements are listed in Annex B.**

3.11.1 Application Selection Data Elements

Table 56 specifies the data elements used during the application selection process.

Table 56—Data Elements for Application Selection

Name	Length	Value
AID	7	See Table 57
FCI	var up to 48	See *[M/CHIP4]*

Table 57 specifies the *AID* and *Application Label* for the MasterCard and Maestro products.

Table 57—*AID* and *Application Label* for MasterCard and Maestro

	MasterCard	Maestro
AID	'A0 00 00 00 04 10 10'	'A0 00 00 00 04 30 60'
Application Label	"MasterCard" or "MASTERCARD"	"Maestro" or "MAESTRO"

3.11.2 COMPUTE CRYPTOGRAPHIC CHECKSUM Data Objects

Table 58 lists the persistent card data elements used during the generation of $CVC3_{TRACK1}$ and $CVC3_{TRACK2}$.

Table 58—Data Elements for CVC3 Generation

Tag	Data Element	Length (bytes)	Format	Value
--	*Static* $CVC3_{TRACK1}$ (1)	2	Binary	Refer to Section 3.9.4
--	*Static* $CVC3_{TRACK2}$ (1)	2	Binary	Refer to Section 3.9.6
--	$IVCVC3_{TRACK1}$ (1)	2	Binary	Refer to Section 3.9.7
--	$IVCVC3_{TRACK2}$	2	Binary	Refer to Section 3.9.8

(1) Mandatory, may be filled with hexadecimal zeros if not used.

3.11.3 Persistent Data Referenced in the *AFL (PayPass)*

Data objects returned by the *PayPass* – M/Chip 4 application during the read application data process when the *PayPass* interface is used, have to be organized in a way as specified in Section 5.13.1 of PART II.

The data elements listed in Table 59 are the *PayPass* – Mag Stripe specific data elements and are included in record 1 of the record file with SFI 1.

Table 59—SFI 1 – Record 1

Tag	Name	Length (bytes)	Presence
'9F6C'	*Mag Stripe Application Version Number (Card)*	2	M
'9F62'	*Track 1 Bit Map for CVC3* (*$PCVC3_{TRACK1}$*)	6	C(1)
'9F63'	*Track 1 Bit Map for UN and ATC* (*$PUNATC_{TRACK1}$*)	6	C(1)
'56'	*Track 1 Data*	var up to 76	O
'9F64'	*Track 1 Nr of ATC Digits* (*$NATC_{TRACK1}$*)	1	C(1)
'9F65'	*Track 2 Bit Map for CVC3* (*$PCVC3_{TRACK2}$*)	2	M
'9F66'	*Track 2 Bit Map for UN and ATC* (*$PUNATC_{TRACK2}$*)	2	M
'9F6B'	*Track 2 Data*	var up to 19	M
'9F67'	*Track 2 Nr of ATC Digits* (*$NATC_{TRACK2}$*)	1	M
'9F68'	*Mag Stripe CVM List*	var up to 32	M

(1) This data element must be present if *Track 1 Data* is present.

The *Mag Stripe Application Version Number (Card)* must be personalized with the value '00 01'.

The personalization of the *Mag Stripe CVM List* depends on the product (Maestro or MasterCard) and on the risk profile of the issuer. For the MasterCard product there are two possibilities: Signature + Online PIN + No CVM (Table 60) or Online PIN + Signature + No CVM (Table 61).

Table 60—MasterCard – *Mag Stripe CVM List* (Signature + Online PIN + No CVM)

CVM	Bit 7 of byte 1 If CVM not successful	Byte 1 setting	Byte 2 setting	Meaning of Byte 2
Signature	Apply next	'5E'	'03'	If supported
Online PIN	Apply next	'42'	'03'	If supported
No CVM	fail	'1F'	'03'	If supported

Table 61—MasterCard – *Mag Stripe CVM List* (Online PIN + Signature + No CVM)

CVM	Bit 7 of byte 1 If CVM not successful	Byte 1 setting	Byte 2 setting	Meaning of Byte 2
Online PIN	Apply next	‘42’	‘03’	If supported
Signature	Apply next	‘5E’	‘03’	If supported
No CVM	fail	‘1F’	‘03’	If supported

For the Maestro product the *Mag Stripe CVM List* must be personalized as specified in Table 62.

Table 62—Maestro – *Mag Stripe CVM List* (Online PIN + Signature)

CVM	Bit 7 of byte 1 If CVM not successful	Byte 1 setting	Byte 2 setting	Meaning of Byte 2
Online PIN	Apply next	‘42’	‘00’	Always
Signature	fail	‘1E’	‘03’	If supported

Table 63 lists the data elements that may be included in record 1 of the file with SFI 2. Record 1 of SFI 2 is the only record to be used as input for the generation of the *Signed Static Application Data*.

Table 63—SFI 2 – Record 1

Tag	Description	Length
‘57’	*Track 2 Equivalent Data*	var up to 19
‘5A’	*Application Primary Account Number (PAN)*	var. up to 10
‘5F20’	*Cardholder Name*	var. up to 26
‘5F24’	*Application Expiry Date*	3
‘5F25’	*Application Effective Date*	3
‘5F28’	*Issuer Country Code*	2
‘5F34’	*PAN Sequence Number*	1
‘8C’	*CDOL1*	var (refer to *[M/CHIP4]*)
‘8D’	*CDOL2*	var (refer to *[M/CHIP4]*)
‘8E’	*CVM List*	var
‘9F07’	*Application Usage Control*	2
‘9F08’	*Application Version Number*	2
‘9F0D’	*Issuer Action Code – Default*	5
‘9F0E’	*Issuer Action Code – Denial*	5
‘9F0F’	*Issuer Action Code – Online*	5
‘9F42’	*Application Currency Code*	2

Table 64 and Table 65 list the data elements included in the first and second record of the file with SFI 3. These records include the data objects required to retrieve the *Issuer Public Key* and to perform static data authentication.

Table 64—SFI 3 – Record 1

Tag	Description	Length
'9F4A'	*SDA Tag List*	var. up to 1
'8F'	*Certification Authority Public Key Index*	1
'9F32'	*Issuer Public Key Exponent*	var. up to 3
'92'	*Issuer Public Key Remainder*	N_I-N_{CA}+36
'90'	*Issuer Public Key Certificate*	N_{CA}

Table 65—SFI 3 – Record 2

Tag	Description	Length
'93'	*Signed Static Application Data*	N_I

Table 66 and Table 67 list the data objects required to retrieve the *ICC Public Key* and to perform dynamic data authentication. This file is only present for a *PayPass* – M/Chip Select 4 application.

Table 66—SFI 4 – Record 1

Tag	Description	Length
'9F47'	*ICC Public Key Exponent*	var. up to 3
'9F48'	*ICC Public Key Remainder*	N_{IC}-N_I+42

Table 67—SFI 4 – Record 2

Tag	Description	Length
'9F46'	*ICC Public Key Certificate*	N_I

Note **If the the *SDA Tag List* (tag '9F4A') is returned by the *PayPass* – M/Chip 4 application and the *AIP* (tag '82') is included, then the *Signed Static Application Data* (if SDA is supported) and the *ICC Public Key Certificate* (if DDA or CDA are supported) are different for the contact and *PayPass* interface. The *Signed Static Application Data* and *ICC Public Key Certificate* for the contact interface include the *Application Interchange Profile*, while the *Signed Static Application Data* and *ICC Public Key Certificate* for the *PayPass* interface include the *Application Interchange Profile (PayPass)*. In this case the *Signed Static Application Data* and *ICC Public Key Certificate* for the contact interface must be included in records that are not already used by the *PayPass* interface and the *AFL* for the contact interface must be coded accordingly.**

3.11.4 *Application Interchange Profile (PayPass)*

The *AIP (PayPass)* includes the 'M/Chip profile is supported' bit and must be personalized as specified in Table 68 and Table 69.

Table 68—Byte 1 of the *Application Interchange Profile (PayPass)*

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0								RFU
	1							Offline static data authentication supported
		0						Offline dynamic data authentication supported
			1					Cardholder verification supported
				1				Terminal risk management to be performed
					0			Issuer authentication supported
						0		RFU
							0/1[a]	Combined DDA – GENERATE AC supported

[a] 0b for M/Chip 4 Lite, 1b for M/Chip 4 Select.

Table 69—Byte 2 of the *Application Interchange Profile (PayPass)*

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1								M/Chip profile is supported
	0	0	0	0	0	0	0	RFU

3.11.5 Persistent Data Elements for Card Risk Management

The data elements listed in Table 70 are the *PayPass* – M/Chip 4 specific data elements for card risk management.

Table 70—Persistent Data Elements for Card Risk Management

Tag	Name	Length (bytes)
'CD'	*Card Issuer Action Code (PayPass) – Decline*	3
'CE'	*Card Issuer Action Code (PayPass) – Default*	3
'CF'	*Card Issuer Action Code (PayPass) – Online*	3

3.11.6 *Application File Locator (PayPass)*

The *AFL (PayPass)* of the *PayPass* – M/Chip 4 application must be personalized as specified in Section 5.13.2 of PART II.

3.11.7 *Application Control (PayPass)*

The *Application Control* data element activates or de-activates functions in the application. The first byte (leftmost) of the *Application Control (PayPass)* must be personalized as specified in Table 71.

Table 71—Byte 1 of the *Application Control (PayPass)*

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0								Magstripe grade issuer not activated
	0/1							Skip CIAC-default on CAT3 (0b: do not skip CIAC default, 1b: skip CIAC default)
		0						Reserved
			0					Key for offline encrypted PIN
				0				Offline encrypted PIN not supported
					0			Offline plaintext PIN not supported
						0/1		Session key derivation (0b: EPI/MCI, 1b: EMV 2000)
							0/1	Encrypt offline counters (0b: Do not encrypt, 1b: Encrypt)

No specific personalization requirements exist for byte 2 and byte 3 of the *Application Control (PayPass)*.

3.11.8 Triple DES Key

Table 72 lists the Triple DES key used for CVC3 generation.

Table 72—Triple DES Key for CVC3 Generation

Data Element	Length
ICC Derived Key for CVC3 Generation (KD_{CVC3})	16

PART IV – Annexes

PART IV includes the annexes of the PayPass – M/Chip Technical Specifications.

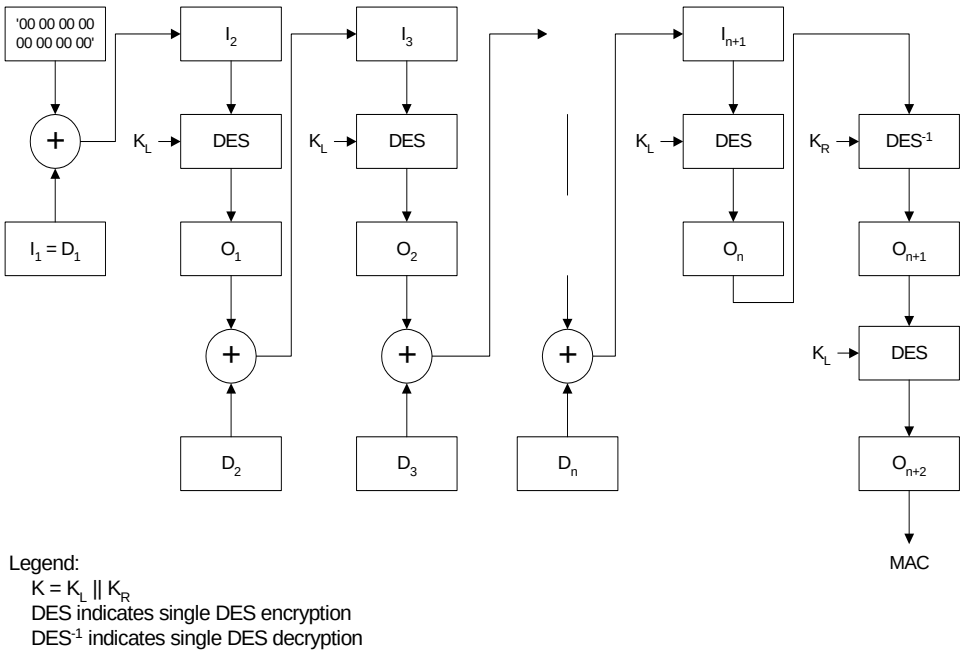
Annexes

Annex A: MAC Algorithm

Data integrity of a message M is achieved by generating a MAC using a 16-byte key K as follows:

1. An initial vector is set to eight bytes of hexadecimal zeroes.
2. For the generation of the MAC, the message M is formatted into eight-byte data blocks, labeled D_1, D_2, D_3, D_4, etc.
3. If the size of the last data block is eight bytes, an additional eight-byte data block is concatenated to the right of the last data block: '80 00 00 00 00 00 00 00'. Proceed to step 4.
   If the size of the last data block is less than eight bytes, it is padded to the right with a one-byte hexadecimal '80'. If the last data block is now eight bytes in length, then proceed to step 4. If the last data block is still less than eight bytes, it is right-filled with hexadecimal zeroes until it is eight bytes in length.
4. The MAC is generated using the key K as shown in Figure 13. Figure 13 assumes that after the padding there are n data blocks (D).

Figure 13—MAC Algorithm



Annex B: PayPass Data Groupings

This annex defines the structure of the data groupings that must be used to personalize the *PayPass* specific data elements when personalizing the *PayPass* – M/Chip 4 application according to the EMV Card Personalization Specification as defined in *[M/CHIP4 CPS]*.

DGI 'B002'

Data Element	Length
Application Control (PayPass)	3
Static CVC3$_{TRACK1}$	2
Static CVC3$_{TRACK2}$	2
IVCVC3$_{TRACK1}$	2
IVCVC3$_{TRACK2}$	2

DGI 'B005'

Data Element	Length
Application Interchange Profile (PayPass)	2
Application File Locator (PayPass)	var.

DGI '8400' – Encrypted

Data Element	Length
ICC Derived Key for CVC3 Generation (*KD$_{CVC3}$*)	16

***** End of Document *****