



PayPass – Mag Stripe

Technical Specifications

Version 3.3 • December 2007

Copyright

The information contained in this document is proprietary and confidential to MasterCard International Incorporated, one or more of its affiliated entities (collectively “MasterCard”) or both.

This material may not be duplicated, published, or disclosed, in whole or in part, without the prior written permission of MasterCard.

Trademarks

Trademark notices and symbols used in this manual reflect the registration status of MasterCard trademarks in the United States. Please consult with the Customer Operations Services team or the MasterCard Law Department for the registration status of particular product, program, or service names outside the United States.

All third-party product and service names are trademarks or registered trademarks of their respective owners.

Media

This document is available in both electronic and printed format.

MasterCard Worldwide - CCOE
Chaussée de Tervuren, 198A
B-1410 Waterloo
Belgium

E-mail: specifications@paypass.com

Using this Manual	ix
Purpose.....	ix
Audience.....	ix
Overview	x
References	xi
Abbreviations.....	xii
Notational Conventions	xiv
Transition Flow Diagram	xvi
Requirement Numbering.....	xvii
Document Word Usage	xvii
Guidance on Terminology	xvii
 PART I – Introduction.....	 1
1 PayPass – Mag Stripe Specifications	3
2 Dynamic CVC (CVC3)	3
3 Architecture.....	4
 PART II – Interface Specification	 7
1 Application Selection	11
1.1 Introduction.....	11
1.2 Select Command	12
1.2.1 Command Message	12
1.2.2 Data Field Returned in the Response Message.....	12
1.2.3 Status Bytes for SELECT Command.....	14
1.3 Building the Candidate List	15
1.4 Final Selection	15
2 Commands.....	17
2.1 Introduction.....	17
2.2 COMPUTE CRYPTOGRAPHIC CHECKSUM.....	18
2.2.1 Definition and Scope	18
2.2.2 Command Message.....	18
2.2.3 Data Field Returned in the Response Message.....	18

Table of Contents

2.2.4	Status Bytes for COMPUTE CRYPTOGRAPHIC CHECKSUM Command.....	19
2.3	GET PROCESSING OPTIONS.....	19
2.3.1	Definition and Scope.....	19
2.3.2	Command Message	19
2.3.3	Data Field Returned in the Response Message	20
2.3.4	Status Bytes for GET PROCESSING OPTIONS Command	20
2.4	READ RECORD.....	21
2.4.1	Definition and Scope.....	21
2.4.2	Command Message	21
2.4.3	Data Field Returned in the Response Message	21
2.4.4	Status Bytes for READ RECORD Command.....	22
3	PayPass Reader Transaction Processing.....	23
3.1	Overview of Command – Response APDU Flow.....	23
3.2	PayPass – Mag Stripe Transaction Flow	25
3.3	Exception Processing	26
3.3.1	Data Objects.....	26
3.3.2	Status Bytes.....	27
3.3.3	Compute Cryptographic Checksum	27
3.4	Functions Used in Transaction Processing.....	28
3.4.1	Get Processing Options Command Processing	28
3.4.2	Read Application Data	28
3.4.3	Application Version Number Checking.....	29
3.4.4	Compute Cryptographic Checksum Command Processing	30
3.4.5	Completion.....	32
4	Card Interoperability Requirements.....	33
4.1	DOL Handling.....	33
4.2	Exception Processing	33
4.3	ATC.....	33
4.4	Select PPSE Command Processing.....	34
4.5	Select AID Command Processing	34
4.6	Get Processing Options Command Processing	35
4.7	Read Record Command Processing	35
4.8	Compute Cryptographic Checksum Command Processing	36
4.9	Card Personalization Requirements	37
5	Data Objects	39
5.1	Data Object Format.....	39
5.2	DOL Handling.....	40
5.3	Bitmap	41
5.4	Data Object Dictionary	42
5.4.1	ADF Name	42
5.4.2	AFL	43
5.4.3	AID (Reader).....	43
5.4.4	AIP	43

5.4.5	Application Label	44
5.4.6	Application Preferred Name	44
5.4.7	Application Priority Indicator	44
5.4.8	Application Version Number (Card)	45
5.4.9	Application Version Number (Reader)	45
5.4.10	ATC	45
5.4.11	CVC3 _{TRACK1}	46
5.4.12	CVC3 _{TRACK2}	46
5.4.13	DD _{CARD,TRACK1}	46
5.4.14	DD _{CARD,TRACK2}	47
5.4.15	Default UDOL	47
5.4.16	DF Name	47
5.4.17	Issuer Code Table Index	48
5.4.18	Language Preference	48
5.4.19	NATC _{TRACK1}	48
5.4.20	NATC _{TRACK2}	49
5.4.21	PayPass Third Party Data	49
5.4.22	PCVC3 _{TRACK1}	50
5.4.23	PCVC3 _{TRACK2}	50
5.4.24	PDOL	50
5.4.25	PUNATC _{TRACK1}	51
5.4.26	PUNATC _{TRACK2}	51
5.4.27	Track 1 Data	52
5.4.28	Track 2 Data	53
5.4.29	UDOL	53
5.4.30	UN	54

PART III – Card Specification 55

1	Introduction	59
1.1	Support for the PPSE	59
1.2	Personalization Process	60
1.3	State Machine	60
1.4	Functional Overview	62
1.5	Structure of the Document	63
2	PPSE Application	65
2.1	Introduction	65
2.2	Application State Machine	65
2.3	Command Processing	66
2.3.1	C-APDU Recognition	66
2.3.2	C-APDU Acceptance	67
2.3.3	SELECT PPSE	68
2.3.4	LOOP BACK	70

Table of Contents

3	PayPass – Mag Stripe Application.....	73
3.1	Application State Machine.....	73
3.2	Variables	76
3.2.1	Internal Flags in Non Volatile Memory	76
3.2.2	ATC.....	76
3.3	C-APDU Processing.....	76
3.3.1	C-APDU Recognition	76
3.3.2	C-APDU Acceptance	77
3.3.3	C-APDU Processing.....	78
3.4	SELECT	79
3.4.1	Command Message	79
3.4.2	Response Message	79
3.4.3	Processing	79
3.4.4	Destination States.....	80
3.5	GET PROCESSING OPTIONS.....	81
3.5.1	Command Message	81
3.5.2	Response Message	81
3.5.3	Processing	82
3.5.4	Destination States.....	84
3.6	READ RECORD.....	84
3.6.1	Command Message	84
3.6.2	Response Message	85
3.6.3	Processing	85
3.6.4	Destination States.....	86
3.7	COMPUTE CRYPTOGRAPHIC CHECKSUM.....	87
3.7.1	Command Message	87
3.7.2	Data Field Returned in the Response Message	87
3.7.3	Processing	88
3.7.4	Destination States.....	89
3.8	Dynamic CVC3.....	90
3.8.1	KD _{CVC3}	90
3.8.2	Dynamic CVC3 Generation	91
3.8.3	IVCVC3 Generation	91
3.9	Data Objects Dictionary	92
3.9.1	IVCVC3 _{TRACK1}	92
3.9.2	IVCVC3 _{TRACK2}	92
4	PayPass Application Personalization.....	93
4.1	Personalization Processing Flow.....	93
4.2	Personalization State Machine	94
4.2.1	Personalization States.....	94
4.2.2	State Diagram.....	95
4.3	Personalization Command Processing	96
4.3.1	C-APDU Recognition	96
4.3.2	C-APDU Acceptance	97
4.3.3	SELECT Command.....	97

4.3.4	GET DATA Command.....	98
4.3.5	STORE DATA Command.....	100
4.4	Personalization Data	104
4.4.1	DGI '0101' – Data Objects Referenced in the AFL	104
4.4.2	DGI 'A001' – Persistent Data Objects for CVC3 Generation	105
4.4.3	DGI 'A002' – Secret Key	105
4.5	Cryptography for Personalization	106
4.5.1	KD _{PERSO}	106
4.5.2	MAC for STORE DATA	106
4.5.3	Encryption/Decryption.....	106
5	PayPass Application Pre-Personalization	107
5.1	Application Selection Data Objects	107
5.2	GET PROCESSING OPTIONS Response	108
5.3	Secret Key Information.....	108
5.4	KD _{PERSO}	108
5.5	ATC	108
Annex A	Example.....	109
A.1	Pre-Personalization Data.....	109
A.2	Personalization Data	110
A.3	Personalization Commands.....	111
A.4	Transaction Processing	112
Annex B	MAC Algorithm.....	115
Annex C	Multi-Application Manager	117
C.1	Introduction.....	117
C.2	The Multi-Application Manager	117
C.3	Processing of a C-APDU	118
C.4	Processing of the SELECT Command	119

Table of Contents

Using this Manual

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

Purpose

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

This manual contains the technical specifications for the *PayPass – Mag Stripe* application. These specifications are necessary to achieve interoperability between *PayPass* cards and *PayPass* readers. They define:

- The transaction flow (the sequence of events and the commands and responses interchanged between *PayPass* card and reader)
- The commands and data objects as they apply to the exchange of information between *PayPass* card and reader
- The implementation of the *PayPass – Mag Stripe* application on a *PayPass* card

Audience

MasterCard provides this manual for members and their authorized agents. Specifically, the following personnel will find this manual useful:

- Customers and vendors who want to implement the *PayPass – Mag Stripe* application on a card or reader
- Those involved in type approval services and in verifying actual implementations against these specifications

**Note**

It is assumed that readers already have an understanding of chip card technology in general, and of ISO/IEC 14443, ISO/IEC 7816-4 and EMV in particular.

Overview

The following table provides an overview of this manual:

Section	Description
Table of Contents	A list of the manual's sections and subsections.
Using this Manual	A description of the manual's purpose and its contents.
PART I – Introduction	Provides a high-level summary of the <i>PayPass – Mag Stripe</i> application.
PART II – Interface Specification	Describes the functions necessary to ensure that <i>PayPass – Mag Stripe</i> cards conforming to this specification can perform a set of common core functions in all <i>PayPass</i> readers that conform to this specification. Application functions unique to individual implementations (and functions not performed in interchange) are not described here. This part describes: • The application selection mechanism • The transaction flow (the sequence of events and the commands and responses interchanged between <i>PayPass</i> card and reader) • The definition of commands and data objects as they apply to the exchange of information between <i>PayPass</i> card and reader • The <i>PayPass</i> card and <i>PayPass</i> reader interoperability requirements (This part does not address clearing and settlement issues or transactions where the <i>PayPass</i> card is not present.)
PART III – Card Specification	Describes the <i>PayPass – Mag Stripe</i> card implementation proposed by MasterCard and conforming to the requirements listed in PART II. The card implementation specification provides a definition of the behavior of the contactless card during both the personalization and the operational phase of the card life cycle.
A – Personalization and Transaction Examples	Provides examples that explain how a <i>PayPass – Mag Stripe</i> card is personalized, how the dynamic data is calculated, and how commands and responses are formatted.
B – MAC Algorithm	Describes the MAC algorithm used by the <i>PayPass – Mag Stripe</i> card.
C – Multi-Application Manager	Introduces the PPSE application and the <i>PayPass – Mag Stripe</i> application in a multi-application environment.

References

The following reference materials contain provisions that are referenced in this specification. The latest version shall apply unless a publication date is explicitly stated.

[ISO 639]	Codes for the representation of names and languages.
[ISO/IEC 7813]	Identification cards – Financial transaction cards.
[ISO/IEC 7816-4]	Information technology – Identification cards – Integrated circuit(s) cards with contacts - Part 4: Interindustry commands for interchange.
[ISO/IEC 7816-5]	Identification cards – Integrated circuit(s) cards with contacts – Part 5: Numbering system and registration procedure for application identifiers.
[ISO 8859]	Information processing – 8-bit single byte coded graphic character sets.
[ISO/IEC 10116]	Information Technology – Modes of operation of an n-bit block cipher algorithm.
[EMV 4.1 BOOK 1]	Integrated Circuit Card Specifications for Payment Systems: Application Independent ICC to Terminal Interface Requirements. Version 4.1, June 2007.
[EMV 4.1 BOOK 3]	Integrated Circuit Card Specifications for Payment Systems: Application Specification. Version 4.1, June 2007.
[EMV PERSO]	EMV Card Personalization Specification. Version 1.0, June 2003.
[EMV PROTOCOL]	EMV Contactless Specifications for Payment Systems: EMV Contactless Communication Protocol Specification. Version 2.0, August 2007.

Abbreviations

The following abbreviations are used in this manual:

Abbreviation	Description
ADF	Application Definition File
AID	Application Identifier
AIP	Application Interchange Profile
AFL	Application File Locator
an	Alphanumeric
ans	Alphanumeric Special
APDU	Application Protocol Data Unit
ATC	Application Transaction Counter
b	Binary
BCD	Binary Coded Decimal
CLA	Class byte of command message
cn	Compressed Numeric
CSN	Chip Serial Number
CVC	Card Validation Code
DD	Discretionary Data
DES	Data Encryption Standard
DGI	Data Grouping Identifier
EMV	Europay MasterCard Visa
FCI	File Control Information
ICC	Integrated Circuit Card
IMK _{CVC3}	Issuer Master Key for CVC3 Generation
INS	Instruction byte of command message
ISO	International Organization for Standardization
KD _{CVC3}	ICC Derived Key for CVC3 Generation
KD _{PERSO}	ICC Derived Key for Personalization
KMC	Master Key for Personalization
Lc	Number of bytes present in the data field of the C-APDU
Le	Maximum length of bytes expected in the data field of the R-APDU.
M	Mandatory
MAC	Message Authentication Code
n	Numeric

Abbreviation	Description
NATC _{TRACK1}	Track 1 Number of ATC Digits
NATC _{TRACK2}	Track 2 Number of ATC Digits
O	Optional
PAN	Primary Account Number
PDOL	Processing Options Data Object List
PIN	Personal Identification Number
PPSE	Proximity Payment System Environment
PSE	Payment System Environment
PCVC3 _{TRACK1}	Track 1 Bitmap for CVC3
PCVC3 _{TRACK2}	Track 2 Bitmap for CVC3
PUNATC _{TRACK1}	Track 1 Bitmap for UN and ATC
PUNATC _{TRACK2}	Track 2 Bitmap for UN and ATC
P1	Parameter 1
P2	Parameter 2
RFU	Reserved for Future Use
SW1	Status Byte One
SW2	Status Byte Two
TLV	Tag Length Value
UDOL	Unpredictable Number Data Object List
UN	Unpredictable Number

Notational Conventions

The following notations apply:

Notation	Description
'0' to '9' and 'A' to 'F'	16 hexadecimal digits.
1001b	Binary notation. Values expressed in binary form are followed by a lower case "b".
"abcd"	an or ans string.
xx	Any value.
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. Typical values for ALG are DES, DES3, DES ⁻¹ and DES3 ⁻¹ .
X & Y	The bit-wise AND of the data block X and Y.
$X \oplus Y$	The bit-wise EXOR of the data block X and Y.
<i>Data Object</i>	Data objects are written in <i>italics</i> to distinguish them from the text.
COMMAND APDU	Command APDUs are written in SMALL CAPITALS to distinguish them from the text.
STATE	States are written in COURIER FONT to distinguish them from the text.

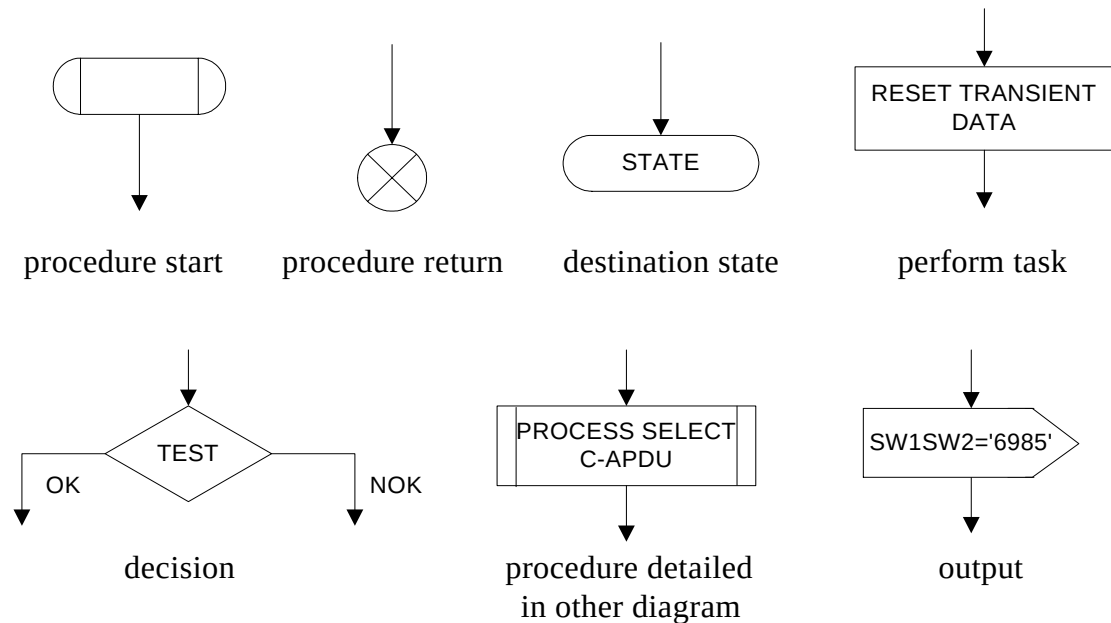
The following table lists symbols that are used throughout the document:

Symbol	Meaning
k_{TRACK1}	Number of non-zero bits in $PUNATC_{\text{TRACK1}}$
k_{TRACK2}	Number of non-zero bits in $PUNATC_{\text{TRACK2}}$
t_{TRACK1}	The symbol t_{TRACK1} represents the value of $NATC_{\text{TRACK1}}$ and indicates the number of digits of the ATC to be included in the discretionary data field of the <i>Track 1 Data</i> .
t_{TRACK2}	The symbol t_{TRACK2} represents the value of $NATC_{\text{TRACK2}}$ and indicates the number of digits of the ATC to be included in the discretionary data field of the <i>Track 2 Data</i> .
n_{UN}	The symbol n_{UN} represents the number of positions available in the discretionary data fields of the <i>Track 1 Data</i> and <i>Track 2 Data</i> for transporting UN to the issuer. The value of n_{UN} must be the same for <i>Track 1 Data</i> and <i>Track 2 Data</i> . Therefore the following holds: $n_{\text{UN}} = k_{\text{TRACK1}} - t_{\text{TRACK1}} = k_{\text{TRACK2}} - t_{\text{TRACK2}}$.

Symbol	Meaning
m _{TRACK1}	The symbol m _{TRACK1} indicates the number of characters present in the discretionary data field of the <i>Track 1 Data</i> .
m _{TRACK2}	The symbol m _{TRACK2} indicates the number of digits present in the discretionary data field of the <i>Track 2 Data</i> .
q _{TRACK1}	Number of non-zero bits in the PCVC3 _{TRACK1} . The symbol q _{TRACK1} represents the number of CVC3 digits included in the discretionary data field of the <i>Track 1 Data</i> .
q _{TRACK2}	Number of non-zero bits in the PCVC3 _{TRACK2} . The symbol q _{TRACK2} represents the number of CVC3 digits included in the discretionary data field of the <i>Track 2 Data</i> .

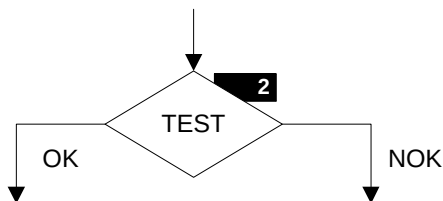
Transition Flow Diagram

The following symbols are used in the transition flow diagrams:



Where a text description accompanies the transition flow diagram, each symbol in the transition flow diagram is assigned a symbol number. A paragraph in the text description starting with Symbol n, corresponds to the symbol bearing the same number in the transition flow diagram, as shown in the following example:

- The decision symbol is used in a flow diagram, and is assigned symbol number 2.



- The step in the process flow diagram assigned symbol number 2 is described:

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:

3.4.1.2 If the *PDOL* is not present, then the *PayPass* reader must use a command data field of '8300'.

A requirement may have different numbers in different versions of the specifications. Hence, all references to a requirement must include the version of the document as well as the requirement's number.

This convention is adopted to make the development of test specifications easier.

Document Word Usage

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

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

Guidance on Terminology

Due to the legacy of the plastic card industry and the fact that the first *PayPass* compliant form factor is card based, the term “card” is used frequently throughout. However, the contactless nature of *PayPass* permits non-card form factors.

The functionality of both *PayPass* cards and devices is driven by the chip inside and is independent of the form factor in which the chip resides, therefore the default reference for the consumer token in this document is “*PayPass* card” or “card”.

PART I – Introduction

This part includes an executive summary of the *PayPass* – Mag Stripe specifications.

1	<i>PayPass</i> – Mag Stripe Specifications	3
2	Dynamic CVC (CVC3)	3
3	Architecture.....	4

1 *PayPass – Mag Stripe Specifications*

The purpose of the *PayPass – Mag Stripe* specifications is to allow contactless chip payments to use authorization networks (proprietary and shared) that presently support magnetic stripe authorizations for credit or debit applications.

The chip authentication process is performed between the chip embedded in the *PayPass* card and the issuer, which means that the additional functionality required in terminals to support *PayPass – Mag Stripe* is minimal. There is also little impact on the acquirer host system, the authorization network and on other entities that may be involved in the transaction.

2 *Dynamic CVC (CVC3)*

The *PayPass* card must be able to store track 1 and track 2 data and a secret key for device authentication. The security of *PayPass – Mag Stripe* is built around a dynamic Card Validation Code (CVC3) included in the discretionary data field of the track data. The CVC3 is generated by the *PayPass* card using a diversified secret key and the following input data:

- The track data
- The *Application Transaction Counter (ATC)* of the *PayPass* card
- The *Unpredictable Number (UN)* provided by the *PayPass* reader

3 Architecture

The general principle of a *PayPass*-Mag Stripe operation is as follows: a *PayPass* card generates an authentication token for the issuer in the form of a dynamic CVC (CVC3). The CVC3, the *UN* and (part of) the *ATC* are sent to the issuer in the discretionary data fields of the track 1 and track 2 data, requiring little or no modification to the acquirer systems and the networks that are presently used for magnetic stripe card authorizations.

This specification considers the *PayPass* – Mag Stripe reader (further on referred to as *PayPass* reader) to be a peripheral device of the terminal. The *PayPass* reader performs the interaction with the *PayPass* card.



Note There is no requirement to create devices using this architecture. This logical architecture is put in place to specify an externally observable behavior. A POS for which terminal and *PayPass* reader are integrated in one physical device can meet the requirements listed in this specification.

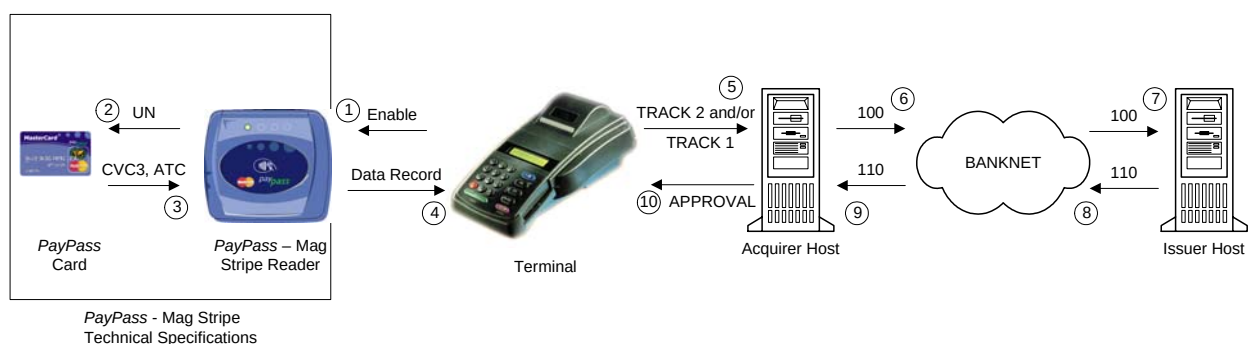
A *PayPass*-Mag Stripe transaction consists of the following main operations:

- The terminal enables the *PayPass* reader.
- The *PayPass* reader creates a list of applications that are supported by both the *PayPass* card and *PayPass* reader.
- The *PayPass* reader picks the highest priority application from the list of mutually supported applications, and issues the SELECT command to select the application on the *PayPass* card.
- The *PayPass* reader issues the GET PROCESSING OPTIONS command to initiate the transaction on the *PayPass* card. The *PayPass* card updates the *ATC*.
- The *PayPass* reader issues the READ RECORD command to retrieve the static data from the *PayPass* card. Amongst other data objects, the *Track 1 Data* (optional) and *Track 2 Data* and the bitmaps are returned. The bitmaps inform the *PayPass* reader where the CVC3, the *ATC* and the *UN* are to be located in the discretionary data fields of the *Track 1 Data* and *Track 2 Data*.
- The *PayPass* reader generates the *UN* and issues the COMPUTE CRYPTOGRAPHIC CHECKSUM command. The *PayPass* card returns the CVC3s for the *Track 1 Data* and *Track 2 Data* and the *ATC*.
- Based on the bitmaps returned in the response from the READ RECORD, the *PayPass* reader inserts the CVC3, (part of) the *ATC* and the *UN* in the discretionary data fields of the *Track 1 Data* and *Track 2 Data*. The *PayPass* reader copies the number of *UN* digits (n_{UN}) in the least significant position of the discretionary data fields of the *Track 1 Data* and *Track 2 Data*.

- The *PayPass* reader prepares a Data Record for the terminal.
The Data Record includes the *Track 1 Data* (if present) and *Track 2 Data* including the dynamic CVC3 data. The Data Record also includes the discretionary data fields of the *Track 1 Data* (if present) and *Track 2 Data* as returned by the card (i.e. without *UN*, *ATC*, CVC3 and *n_{UN}* included).
The Data Record may also include *PayPass Third Party Data* (e.g. loyalty data returned by the card), card data for receipt printing (e.g. *Application Label*, *Application Preferred Name* and *Issuer Code Table Index*) and the cardholder's language preference (*Language Preference*).
- The *PayPass* reader transfers the data objects of the Data Record required by the terminal to the terminal.
- Depending on the product rules, the terminal may request the cardholder to enter his PIN for online PIN processing.
- The terminal formats the authorization request as for a magnetic stripe transaction and sends the authorization request to the issuer.
- Upon receipt of the authorization request, the issuer verifies the CVC3 and processes the authorization in a similar way to a magnetic stripe card authorization.
- Depending on the product rules, the terminal prints a receipt with a line for cardholder signature.

Figure 1 gives a schematic overview of a *PayPass* – Mag Stripe transaction and illustrates the scope of this document.

Figure 1—*PayPass* – Mag Stripe Transaction



1. Terminal enables reader.
2. *PayPass* reader sends *UN* to *PayPass* card.
3. *PayPass* card sends dynamic CVC3 and *ATC* to *PayPass* reader.
4. *PayPass* reader sends Data Record to terminal.
5. Terminal sends authorization data to acquirer host.
6. Acquirer host sends 100 message on Banknet.
7. Banknet forwards 100 message to issuer.
8. Issuer responds with 110 message.
9. Banknet forwards 110 message to acquirer.
10. Acquirer host informs terminal about outcome of transaction.

PART II – Interface Specification

PART II of this document describes the interface between a *PayPass* card and *PayPass* reader, in terms of:

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

1	Application Selection	11
1.1	Introduction.....	11
1.2	Select Command	12
1.2.1	Command Message	12
1.2.2	Data Field Returned in the Response Message.....	12
1.2.3	Status Bytes for SELECT Command.....	14
1.3	Building the Candidate List	15
1.4	Final Selection	15
2	Commands	17
2.1	Introduction.....	17
2.2	COMPUTE CRYPTOGRAPHIC CHECKSUM.....	18
2.2.1	Definition and Scope	18
2.2.2	Command Message.....	18
2.2.3	Data Field Returned in the Response Message.....	18
2.2.4	Status Bytes for COMPUTE CRYPTOGRAPHIC CHECKSUM Command	19
2.3	GET PROCESSING OPTIONS	19
2.3.1	Definition and Scope	19
2.3.2	Command Message.....	19
2.3.3	Data Field Returned in the Response Message.....	20
2.3.4	Status Bytes for GET PROCESSING OPTIONS Command	20
2.4	READ RECORD.....	21
2.4.1	Definition and Scope	21
2.4.2	Command Message.....	21
2.4.3	Data Field Returned in the Response Message.....	21
2.4.4	Status Bytes for READ RECORD Command.....	22

3	PayPass Reader Transaction Processing.....	23
3.1	Overview of Command – Response APDU Flow.....	23
3.2	PayPass – Mag Stripe Transaction Flow.....	25
3.3	Exception Processing.....	26
3.3.1	Data Objects.....	26
3.3.2	Status Bytes.....	27
3.3.3	Compute Cryptographic Checksum.....	27
3.4	Functions Used in Transaction Processing.....	28
3.4.1	Get Processing Options Command Processing.....	28
3.4.2	Read Application Data.....	28
3.4.3	Application Version Number Checking.....	29
3.4.4	Compute Cryptographic Checksum Command Processing.....	30
3.4.5	Completion.....	32
4	Card Interoperability Requirements.....	33
4.1	DOL Handling.....	33
4.2	Exception Processing.....	33
4.3	ATC.....	33
4.4	Select PPSE Command Processing.....	34
4.5	Select AID Command Processing.....	34
4.6	Get Processing Options Command Processing.....	35
4.7	Read Record Command Processing.....	35
4.8	Compute Cryptographic Checksum Command Processing.....	36
4.9	Card Personalization Requirements.....	37
5	Data Objects	39
5.1	Data Object Format.....	39
5.2	DOL Handling.....	40
5.3	Bitmap.....	41
5.4	Data Object Dictionary	42
5.4.1	ADF Name.....	42
5.4.2	AFL.....	43
5.4.3	AID (Reader).....	43
5.4.4	AIP.....	43
5.4.5	Application Label.....	44
5.4.6	Application Preferred Name.....	44
5.4.7	Application Priority Indicator.....	44
5.4.8	Application Version Number (Card).....	45
5.4.9	Application Version Number (Reader).....	45
5.4.10	ATC.....	45
5.4.11	CVC3 _{TRACK1}	46
5.4.12	CVC3 _{TRACK2}	46
5.4.13	DD _{CARD,TRACK1}	46
5.4.14	DD _{CARD,TRACK2}	47
5.4.15	Default UDOL.....	47
5.4.16	DF Name.....	47

5.4.17	Issuer Code Table Index	48
5.4.18	Language Preference.....	48
5.4.19	NATC _{TRACK1}	48
5.4.20	NATC _{TRACK2}	49
5.4.21	<i>PayPass</i> Third Party Data.....	49
5.4.22	PCVC3 _{TRACK1}	50
5.4.23	PCVC3 _{TRACK2}	50
5.4.24	PDOL	50
5.4.25	PUNATC _{TRACK1}	51
5.4.26	PUNATC _{TRACK2}	51
5.4.27	Track 1 Data.....	52
5.4.28	Track 2 Data.....	53
5.4.29	UDOL	53
5.4.30	UN.....	54

1 Application Selection

This chapter describes the application selection mechanism from the standpoint of both the *PayPass* card and the *PayPass* reader.

1.1 Introduction

The application selection mechanism is designed to minimize the number of commands sent to the card – only two **SELECT** commands are necessary.

The process consists of two steps:

- Create a list of applications that are supported by both the *PayPass* card and the *PayPass* reader. This list is called the candidate list.
- From the candidate list, select the application to be run.



Note

A *PayPass* reader which supports only one application (and therefore only one *AID*), should immediately try to select the *ADF Name (AID)* of the corresponding application on the card, and skip the application selection process. In this case, only one **SELECT command is required.**

1.2 Select Command

The SELECT command is used to select the PPSE and the *PayPass* application. The response from the card consists of returning the *FCI*.

1.2.1 Command Message

The SELECT command message is coded according to Table 1.

Table 1—SELECT Command Message

Code	Value
CLA	‘00’
INS	‘A4’
P1	‘04’
P2	‘00’
Lc	‘05’ – ‘10’
Data	File Name
Le	‘00’

The data field of the command message contains the PPSE directory name (“2PAY.SYS.DDF01”) or the *ADF Name* (or *AID*) of the *PayPass* application in the card.¹

1.2.2 Data Field Returned in the Response Message

The data field of the response message contains the *FCI* associated with the PPSE or with the selected *PayPass* application. Additional tags returned in the *FCI* that are not described in this specification must be ignored by the *PayPass* reader.

Table 2 defines the structure of the *FCI* returned by successful selection of the PPSE directory. The *FCI* contains the list of *PayPass* applications (*ADF Names*) supported by the card.

¹ Depending on the value of the File Name, the SELECT command is also referred to as SELECT PPSE and SELECT AID command.

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

Tag	Value	Presence
'6F'	<i>FCI Template</i>	M
'84'	<i>DF Name</i>	M
'A5'	<i>FCI Proprietary Template</i>	M
'BF0C'	<i>FCI Issuer Discretionary Data</i>	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'	<i>ADF Name</i>	M
'87'	<i>Application Priority Indicator</i> (see Table 5)	M
'50'	<i>Application Label</i>	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.

Table 6 defines the FCI returned in response to a successful selection of a *PayPass* application.

Table 6—SELECT Response Message Data Field (FCI) of a *PayPass* Application

Tag	Value	Presence
'6F'	<i>FCI Template</i>	M
'84'	<i>DF Name</i>	M
'A5'	<i>FCI Proprietary Template</i>	M ²
'50'	<i>Application Label</i>	O
'87'	<i>Application Priority Indicator</i>	O
'5F2D'	<i>Language Preference</i>	O
'9F38'	<i>PDOL</i>	O
'9F11'	<i>Issuer Code Table Index</i>	O
'9F12'	<i>Application Preferred Name</i>	O
'BF0C'	<i>FCI Issuer Discretionary Data</i>	O
'XXXX'	1 or more additional data objects from application provider, Issuer or ICC supplier	O

1.2.3 Status Bytes for SELECT Command

The status bytes which may be returned in response to the **SELECT** command are listed in Table 7.

Table 7—Status Bytes for SELECT Command

SW1	SW2	Meaning
'62'	'83'	Selected file invalidated ³
'67'	'00'	Wrong length
'6A'	'81'	Function not supported
'6A'	'82'	File not found
'6A'	'86'	Incorrect parameters P1-P2
'90'	'00'	Normal processing

² The *FCI Proprietary Template* may be empty. In this case the length must be set to zero.

³ These specifications do not specify how to block the PPSE or *PayPass* application. For a dual-interface card (contact and contactless), this may be done by using the contact interface. The card implementation specified in PART III of this document does not support the blocking of the PPSE or *PayPass* application.

1.3 Building the Candidate List

The steps the *PayPass* reader takes to build the candidate list are as follows:

- 1.3.1.1 The *PayPass* reader must select the PPSE using the **SELECT** command as described in Section 1.2 of PART II.
- 1.3.1.2 If the *PayPass* card returns status bytes other than '9000', then the *PayPass* reader must terminate the transaction.
- 1.3.1.3 If any error occurs in 1.3.1.4 or 1.3.1.5, then the *PayPass* reader must terminate the transaction.
- 1.3.1.4 The *PayPass* reader must retrieve all the directory entries from the *FCI Issuer Discretionary Data* (tag 'BF0C') in the *FCI* returned by the *PayPass* card. The *PayPass* reader must process each directory entry by comparing the *ADF Name* in the directory entry with the *AIDs* supported by the *PayPass* reader. If the *ADF Name* matches the *AID* of one of the applications supported by the *PayPass* reader, then the directory entry is added to the candidate list.
The *ADF Name* in the directory entry matches an *AID* in the *PayPass* reader if the *ADF Name* has the same length and value as the *AID* or the *ADF Name* begins with the entire *AID*.
- 1.3.1.5 If no directory entries are found that match applications supported by the *PayPass* reader, then the *PayPass* reader must terminate the transaction.

1.4 Final Selection

Once the *PayPass* reader has determined the candidate list of mutually supported applications, it proceeds as follows:

- 1.4.1.1 The *PayPass* reader must remove from the candidate list any directory entries prohibiting selection of the application without cardholder assistance (b8 = 1b in the *Application Priority Indicator* (see Table 5)).
- 1.4.1.2 The *PayPass* reader must order the candidate list according to the following rules:
 - The candidate list must be ordered in priority sequence, with the directory entry with the highest priority listed first.
 - If all directory entries in the candidate list have a priority indicator set to 0000b, then the candidate list must be in the order in which the directory entries were encountered in the card.
 - If some of the directory entries in the candidate list have a priority indicator set to 0000b and other directory entries in the candidate list have a priority indicator different from 0000b, then those with no priority indicator must come last and in the order in which they were encountered in the card.
 - If there is more than one directory entry with the same priority, then these directory entries must be included in the candidate list in the order in which they were encountered in the card.

- 1.4.1.3 If the candidate list is empty, then the *PayPass* reader must terminate the transaction.
- 1.4.1.4 The *PayPass* reader must pick the first directory entry from the candidate list. The *PayPass* reader must select this application with a *SELECT* command coded according to Section 1.2.1 of PART II using the *ADF Name* found in the directory entry.
- 1.4.1.5 The *PayPass* reader must verify if the *FCI* included in the response message of the *SELECT* command is correctly formatted, as specified in Table 6. If this is not the case, then the *PayPass* reader must remove the directory entry from the candidate list, and must resume processing at step 1.4.1.3.
- 1.4.1.6 If the *SELECT* command returns status bytes other than '9000', or mandatory data objects are missing in the *SELECT* response, then the *PayPass* reader must remove the directory entry from the candidate list, and must resume processing at step 1.4.1.3.
- 1.4.1.7 The *PayPass* reader must verify if the *DF Name* (tag '84') returned in the *FCI* is the same as the *ADF Name* provided to the card in the data field of the *SELECT* command. If this is not the case, then the *PayPass* reader must remove the directory entry from the candidate list, and must resume processing at step 1.4.1.3.
- 1.4.1.8 The *PayPass* reader must extract the *Application Label* (tag '50') (if present), the *Issuer Code Table Index* (tag '9F11') (if present), the *PDOL* (if present), the *Application Preferred Name* (tag '9F12') (if present) and the *Language Preference* (tag '5F2D') (if present) from the *FCI* and store them for later use.
- 1.4.1.9 If the *Language Preference* (tag '5F2D') data object is included in the *FCI*, then the *PayPass* reader must perform language selection as specified in [EMV 4.1 BOOK 4], Section 11.1 Language Selection, except for the interactive cardholder language selection. If no match is found and the *PayPass* reader supports more than one language, the *PayPass* reader must automatically select the local language.

2 Commands

This chapter specifies the commands supported by *PayPass – Mag Stripe*.

2.1 Introduction

The INS byte of the C-APDU is structured according to [EMV 4.1 BOOK 1]. The coding of the INS byte and its relationship to the CLA byte are shown in Table 8.

Table 8—Coding of the Instruction Byte

CLA	INS	Meaning
‘80’	‘A8’	GET PROCESSING OPTIONS
‘00’	‘B2’	READ RECORD
‘80’	‘2A’	COMPUTE CRYPTOGRAPHIC CHECKSUM

The status bytes returned by the card are coded according to the specifications in [EMV 4.1 BOOK 3]. In addition to the specific status bytes for every command, the card may also return the status bytes shown in Table 9.

Table 9—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 command initiates the computation of the dynamic CVC3 by the card. The computation is based on the *UN* sent by the *PayPass* reader to the card in the data field of the COMPUTE CRYPTOGRAPHIC CHECKSUM command.

The card responds by returning the *CVC3_{TRACK2}*, the *CVC3_{TRACK1}* (optional) and the *ATC* to the *PayPass* reader.

2.2.2 Command Message

The COMPUTE CRYPTOGRAPHIC CHECKSUM command message is coded according to Table 10.

Table 10—COMPUTE CRYPTOGRAPHIC CHECKSUM Command Message

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

The data field of the command message is coded according to the *UDOL* following the rules defined in Section 5.2 of PART II. If the card does not have a *UDOL*, then the *PayPass* reader uses the *Default 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' (*Response Message Template*). 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').

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

2.2.4 Status Bytes for COMPUTE CRYPTOGRAPHIC CHECKSUM Command

The status bytes which may be returned in response to the COMPUTE CRYPTOGRAPHIC CHECKSUM command are listed in Table 11.

Table 11—Status Bytes for COMPUTE CRYPTOGRAPHIC CHECKSUM Command

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

2.3 GET PROCESSING OPTIONS

2.3.1 Definition and Scope

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

2.3.2 Command Message

The GET PROCESSING OPTIONS command message is coded according to Table 12.

Table 12—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 *Command 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 5.2 of PART II.

2.3.3 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’). Additional data objects returned in the data field that are not described in this specification must be ignored by the *PayPass* reader.

2.3.4 Status Bytes for GET PROCESSING OPTIONS Command

The status bytes which may be returned in response to the *GET PROCESSING OPTIONS* command are listed in Table 13.

Table 13—Status Bytes for GET PROCESSING OPTIONS Command

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

2.4 READ RECORD

2.4.1 Definition and Scope

The READ RECORD command reads a file record in a linear file. The card's response consists of the data contained within the specified record.

2.4.2 Command Message

The READ RECORD command message is coded according to Table 14.

Table 14—READ RECORD Command Message

Code	Value
CLA	'00'
INS	'B2'
P1	Record Number
P2	See Table 15
Lc	Not present
Data	Not present
Le	'00'

Table 15—P2 of READ RECORD Command

b8-b4	b3-b1	Definition
xxxxxb		SFI
	100b	P1 is a record number

2.4.3 Data Field Returned in the Response Message

The data field of the response message contains the contents of 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 16.

Table 16—Read Record Response Message Data Field

'70'	Length	Record Template
------	--------	-----------------

2.4.4 Status Bytes for READ RECORD Command

The status bytes that may be returned in response to the READ RECORD command are listed in Table 17.

Table 17—Status Bytes for READ RECORD Command

SW1	SW2	Description
'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

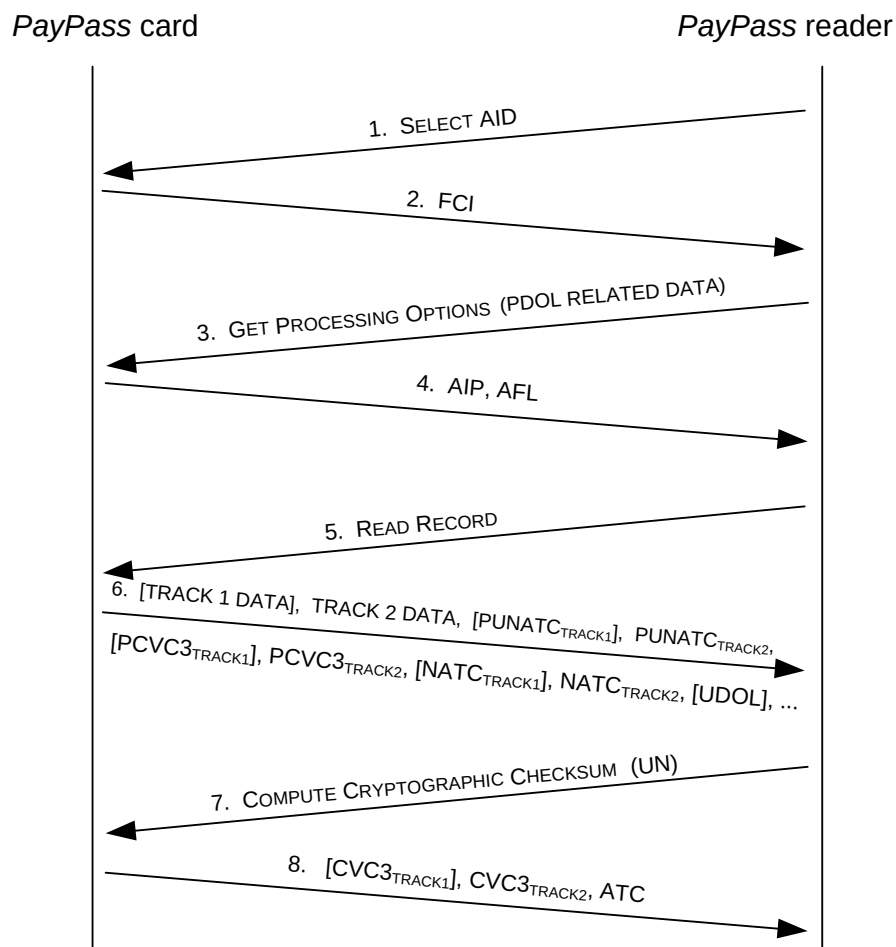
3 PayPass Reader Transaction Processing

This chapter specifies the interaction between the *PayPass* card and the *PayPass* reader after application selection as specified in Chapter 1 of PART II.

3.1 Overview of Command – Response APDU Flow

Figure 2 illustrates the Command – Response APDU flow of a *PayPass* – Mag Stripe transaction including the SELECT AID command.

Figure 2—Command – Response APDU Flow

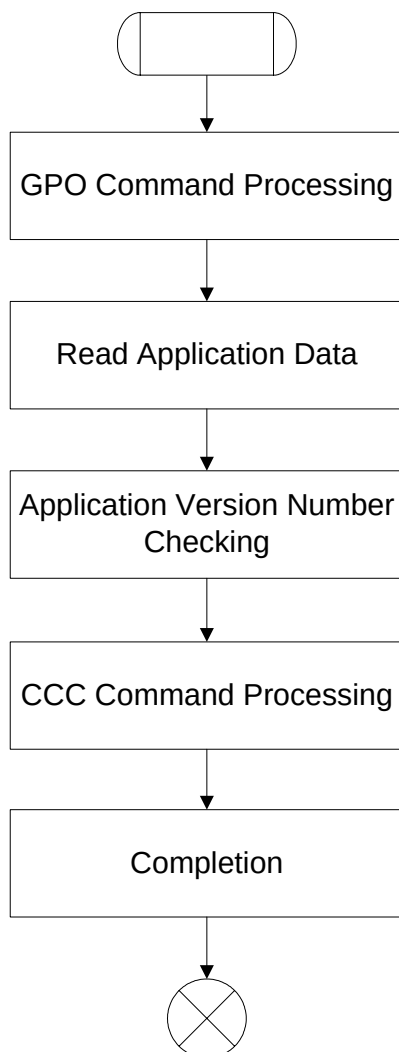


1. The *PayPass* reader issues the SELECT AID command to select the *PayPass* – Mag Stripe application.
2. The card responds with the *FCI*. The *FCI* may contain the *PDOL*. The *PDOL* is a list of tags and lengths of reader-resident data objects needed by the *PayPass* card in the GET PROCESSING OPTIONS command.
3. The *PayPass* reader issues the GET PROCESSING OPTIONS command. If there is no *PDOL* in the *FCI*, then the *PayPass* reader uses the command data field ‘8300’. Otherwise the command data field contains a *Command Template* (tag ‘83’) comprising the concatenated list of data objects resulting from processing the *PDOL*.
4. The card returns the *AIP* and the *AFL*. The *AFL* has a fixed value indicating that all data to be read by the *PayPass* reader are included in record 1 of the file with SFI 1.
5. The *PayPass* reader issues a READ RECORD command to retrieve the static data from the card.
6. The card returns the appropriate data.
7. The *PayPass* reader issues the COMPUTE CRYPTOGRAPHIC CHECKSUM command using a data field comprising the concatenated list of data objects resulting from processing the *UDOL*. If the card did not return a *UDOL* in the response to the READ RECORD command, then the *PayPass* reader uses the *Default UDOL*, which contains as its only entry the tag and length of the *UN* generated by the *PayPass* reader.
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 *UN* sent by the *PayPass* reader, the *ATC* of the card and the relevant secret key stored in the card.
8. The card responds with the dynamic *CVC3_{TRACK2}*, the *CVC3_{TRACK1}* (optional) and the *ATC*. The *PayPass* reader 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 *PCVC3_{TRACK2}*. The *PayPass* reader also copies the relevant digits of the *UN* into the discretionary data field of the *Track 2 Data*.
The *PUNATC_{TRACK2}* indicates where the *PayPass* reader 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 *PayPass* reader 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 in the *PUNATC_{TRACK2}*. After copying the *CVC3_{TRACK2}*, *UN* and *ATC* digits, the *PayPass* reader copies the number of *UN* digits (*n_{UN}*) in the least significant digit of the discretionary data field.
A similar procedure is carried out for the *Track 1 Data*, if the card returned *Track 1 Data* in the response to the READ RECORD command. For the *Track 1 Data* the *PayPass* reader converts the data returned by the card into ASCII encoded characters before copying them into the discretionary data field of the *Track 1 Data*.

3.2 *PayPass* – Mag Stripe Transaction Flow

Section 3.4 specifies the *PayPass* – reader processing on the level of application logical functions. The functions described in Section 3.4 begin after application selection has taken place. This section specifies the order in which these functions must be performed.

Figure 3—*PayPass* – Mag Stripe Logical Functions



- 3.2.1.1 The *PayPass* reader must perform the following logical functions in the order shown:
1. Get Processing Options Processing as specified in Section 3.4.1 of PART II.
 2. Read Application Data as specified in Section 3.4.2 of PART II.
 3. Application Version Number Checking as specified in Section 3.4.3 of PART II.
 4. Compute Cryptographic Checksum Processing as specified in Section 3.4.4 of PART II.
 5. Completion as specified in Section 3.4.5 of PART II.

3.3 Exception Processing

This section specifies exceptions to normal processing that cause termination of the normal transaction flow.

3.3.1 Data Objects

Data objects returned by the card must be checked by the *PayPass* reader in the following way:

- 3.3.1.1 Data objects in the card are classified in Section 5.3 of PART II as either mandatory or optional. If a mandatory data object is missing, then the *PayPass* reader must terminate the transaction.
- 3.3.1.2 If in the course of normal processing the *PayPass* reader recognizes that data is incorrectly formatted, then the *PayPass* reader must terminate 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
 - Multiple occurrences of a data object that must only appear once
 - An *AFL* with invalid syntax
- 3.3.1.3 The *PayPass* reader must not validate the individual data objects returned in the *Track 1 Data* and *Track 2 Data*. Specifically, validation of the values 2 and 6 in the first digit of the service code present in *Track 1 Data* or *Track 2 Data* to determine if a contact chip transaction is required must not be performed by the *PayPass* reader or by the terminal. Any other existing data validation carried out to support individual payment products is outside the scope of this specification. However, if in the course of copying the dynamic data, the *PayPass* reader is not able to locate the discretionary data field due to one or more format errors in the *Track 1 Data* or *Track 2 Data* (e.g. missing separator), the *PayPass* reader must terminate the transaction.

3.3.2 Status Bytes

- 3.3.2.1 Any SW1-SW2 returned by the card other than '9000' must cause termination of the transaction.



Note Requirements 3.3.1.1, 3.3.1.2 and 3.3.2.1 do not apply during application selection.

3.3.3 Compute Cryptographic Checksum

Specific exception processing is required for the COMPUTE CRYPTOGRAPHIC CHECKSUM command.

- 3.3.3.1 If the *PayPass* reader does not receive a valid response message from the card (i.e. no response message or an invalid response message) in response to the COMPUTE CRYPTOGRAPHIC CHECKSUM command, then the *PayPass* reader must wait for 300 ms before terminating the transaction. 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 *PayPass* reader must wait $2 * 300$ ms before terminating the transaction. In general, if this is the j^{th} ($j=1, 2, 3, \dots$) consecutive transaction for which no valid response message from the card for the COMPUTE CRYPTOGRAPHIC CHECKSUM command is received, then the *PayPass* reader must wait $2^m * 300$ ms ($m = \min(j-1, 5)$) before terminating the transaction.

3.4 Functions Used in Transaction Processing

The functions specified in this section begin after application selection and have to be executed in the order as specified in Section 3.2.

3.4.1 Get Processing Options Command Processing

The *PayPass* reader issues the GET PROCESSING OPTIONS command to initiate the transaction on the card.

- 3.4.1.1 The *PayPass* reader must format the GET PROCESSING OPTIONS command as specified in Section 2.3.2 of PART II.
- 3.4.1.2 If the *PDOL* is not present, then the *PayPass* reader must use a command data field of '8300'.
- 3.4.1.3 If the *PDOL* is present, then the *PayPass* reader must use the *PDOL* to create a concatenated list of data objects without tags or lengths following the rules specified in Section 5.2 of PART II. The *PayPass* reader must verify if the tags in the *PDOL* belong to reader resident data objects. If the tag of any data object identified in the *PDOL* does not belong to a reader resident data object, then the *PayPass* reader must provide a data object with the length specified and a value of all hexadecimal zeros. The *PayPass* reader must use the concatenated list as value field of the *Command Template* (tag '83').
- 3.4.1.4 The *PayPass* reader must verify if the response message of the GET PROCESSING OPTIONS command is correctly formatted, as specified in Section 2.3.3 of PART II. If this is not the case, then the *PayPass* reader must terminate the transaction.
- 3.4.1.5 The *PayPass* reader must verify if both the *AIP* (tag '82') and the *AFL* (tag '94') are included in the *Response Message Template* (tag '77'). If they are not both included, then the *PayPass* reader must terminate the transaction.
- 3.4.1.6 The *PayPass* reader must retrieve the *AFL* (tag '94') from the *Response Message Template* (tag '77') to use it during Read Application Data processing.

3.4.2 Read Application Data

Data contained in the files of the *PayPass* card are required by the *PayPass* reader to complete the Compute Cryptographic Checksum Command Processing. The *PayPass* reader uses the READ RECORD command to read the files and records indicated in the *AFL*.

- 3.4.2.1 If the value of the four most significant bytes of the *AFL* is different from the value listed in requirement 4.9.1.1 of PART II, then the *PayPass* reader must process the *AFL* as specified in Section 10.2 of [EMV 4.1 BOOK 3].
- 3.4.2.2 If the value of the four most significant bytes of the *AFL* is the same as the value listed in requirement 4.9.1.1 of PART II, then the *PayPass* reader must not interpret the *AFL* and only read the first record in the file with SFI 1.

- 3.4.2.3 The *PayPass* reader must 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 *PayPass* reader (that is, their tags are unknown by the *PayPass* reader) must not be stored.
- 3.4.2.4 If the *PayPass* reader encounters more than one occurrence of a single primitive data object during read application data processing, the transaction must be terminated.
- 3.4.2.5 All mandatory data objects must be present in the card. If any mandatory data objects are not present, the *PayPass* reader must terminate the transaction. The mandatory data objects are listed in Table 18.

Table 18—Mandatory Data Objects

Tag	Value
'9F6B'	<i>Track 2 Data</i>
'9F66'	<i>PUNATC_{TRACK2}</i>
'9F65'	<i>PCVC3_{TRACK2}</i>
'9F67'	<i>NATC_{TRACK2}</i>

- 3.4.2.6 The *PayPass* reader must copy the discretionary data field of the *Track 1 Data* (if present) into *DD_{CARD,TRACK1}*. The *PayPass* reader must copy the discretionary data field of the *Track 2 Data* into *DD_{CARD,TRACK2}*.

3.4.3 Application Version Number Checking

The application within both the card and *PayPass* reader maintain an *Application Version Number* assigned by the payment system. The *PayPass* reader must verify the compatibility of the application in the *PayPass* reader with the application in the card.

- 3.4.3.1 The *PayPass* reader must use the version number (*Application Version Number (Card)*) in the card to ensure compatibility. If the *Application Version Number (Card)* is not present in the card, then the *PayPass* reader must assume that the reader and card application versions are compatible.
- 3.4.3.2 If the *Application Version Number (Card)* is present in the card and the *PayPass* reader supports the application version of the card, then the *PayPass* reader must use the appropriate code and/or commands to perform the transaction with the card. If the *PayPass* reader does not recognize the application version of the card, then the *PayPass* reader must use its most recent version to perform the transaction.

3.4.4 Compute Cryptographic Checksum Command Processing

The *PayPass* reader issues the COMPUTE CRYPTOGRAPHIC CHECKSUM command to the card to obtain the $CVC3_{TRACK2}$, the $CVC3_{TRACK1}$ (optional) and the *ATC* of the card.

- 3.4.4.1 The *PayPass* reader must verify that the number of non-zero 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 *PayPass* reader must terminate the transaction. Otherwise, the *PayPass* reader must set n_{UN} equal to $k_{TRACK2} - t_{TRACK2}$.
- 3.4.4.2 The *PayPass* reader must verify that n_{UN} is less than or equal to 8. If n_{UN} is greater than 8, then the *PayPass* reader must terminate the transaction.
- 3.4.4.3 The *PayPass* reader must verify that the number of non-zero 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 *PayPass* reader must terminate the transaction.
- 3.4.4.4 If *Track 1 Data* is returned by the card, then the *PayPass* reader must verify that also $PCVC3_{TRACK1}$, $PUNATC_{TRACK1}$ and $NATC_{TRACK1}$ are returned. If at least one of these data objects is not available, then the *PayPass* reader must terminate the transaction.
- 3.4.4.5 If *Track 1 Data* is returned by the card, then the *PayPass* reader must verify that the number of non-zero 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 *PayPass* reader must terminate the transaction.
- 3.4.4.6 If *Track 1 Data* is returned by the card, then the *PayPass* reader must verify that $k_{TRACK1} - t_{TRACK1}$ is equal to n_{UN} . If this is not the case, then the *PayPass* reader must terminate the transaction.
- 3.4.4.7 If *Track 1 Data* is returned by the card, then the *PayPass* reader must verify that the number of non-zero 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 *PayPass* reader must terminate the transaction.
- 3.4.4.8 The *PayPass* reader must retrieve from the *Track 2 Data* the *PAN* and *Expiration Date*. If *Track 1 Data* is returned by the card, then the *PayPass* reader must verify that the *PAN* and *Expiration Date* included in the *Track 1 Data* are the same as the *PAN* and *Expiration Date* included in the *Track 2 Data*. If this is not the case, then the *PayPass* reader must terminate the transaction.
- 3.4.4.9 The *PayPass* reader must generate *UN* of length 8 digits of which the $8 - n_{UN}$ most significant digits are set equal to 0.
- 3.4.4.10 The *PayPass* reader must format the COMPUTE CRYPTOGRAPHIC CHECKSUM command, as specified in Section 2.2.2 of PART II.
- 3.4.4.11 If the *UDOL* is returned by the card in the response to the READ RECORD command, then the *PayPass* reader must create a concatenated list of data objects without tags or lengths, following the rules specified in Section 5.2 of PART II.

- 3.4.4.12 If the *UDOL* is not returned by the card in the response of the READ RECORD command, then the *PayPass* reader must use the *Default UDOL* to construct the data field of the command message.
- 3.4.4.13 The *PayPass* reader must 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 *PayPass* reader must terminate the transaction as indicated in 3.3.3.1.
- 3.4.4.14 The *PayPass* reader must 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 *PayPass* reader must terminate the transaction as indicated in 3.3.3.1.
- 3.4.4.15 The *PayPass* reader must convert the binary encoded $CVC3_{TRACK2}$ to the BCD encoding of the corresponding number expressed in base 10. The *PayPass* reader must 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 $PCVC3_{TRACK2}$.
- 3.4.4.16 The *PayPass* reader must 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 $PUNATC_{TRACK2}$.
- 3.4.4.17 If $t_{TRACK2} \neq 0$, then the *PayPass* reader must convert the *ATC* to the BCD encoding of the corresponding number expressed in base 10. The *PayPass* reader must 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 $PUNATC_{TRACK2}$.
- 3.4.4.18 The *PayPass* reader must copy n_{UN} in the least significant digit of the discretionary data field of the *Track 2 Data*.
- 3.4.4.19 If *Track 1 Data* is returned by the card, then the *PayPass* reader must retrieve the $CVC3_{TRACK1}$ from the *Response Message Template* (tag '77'). If *Track 1 Data* is returned by the card and the $CVC3_{TRACK1}$ is not available, then the *PayPass* reader must terminate the transaction as indicated in 3.3.3.1.
- 3.4.4.20 If *Track 1 Data* is returned by the card, then the *PayPass* reader must convert the binary encoded $CVC3_{TRACK1}$ to the BCD encoding of the corresponding number expressed in base 10. The *PayPass* reader must 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 $PCVC3_{TRACK1}$.
- 3.4.4.21 If *Track 1 Data* is returned by the card, then the *PayPass* reader must 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 $PUNATC_{TRACK1}$.

- 3.4.4.22 If *Track 1 Data* is returned by the card and $t_{\text{TRACK1}} \neq 0$, then the *PayPass* reader must convert the *ATC* to the BCD encoding of the corresponding number expressed in base 10. The *PayPass* reader must convert the t_{TRACK1} least significant digits of the *ATC* into the ASCII format. The *PayPass* reader must 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 $PUNATC_{\text{TRACK1}}$.
- 3.4.4.23 If *Track 1 Data* is returned by the card, then the *PayPass* reader must 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*.
- 3.4.4.24 The *PayPass* reader must execute the requirements 3.4.4.15, 3.4.4.16, 3.4.4.17 and 3.4.4.18 and the requirements 3.4.4.20, 3.4.4.21, 3.4.4.22 and 3.4.4.23 in the order as specified above.

3.4.5 Completion

With the Completion function the *PayPass* reader prepares the Data Record and hands over control to the terminal.

- 3.4.5.1 The *PayPass* reader must add the following data objects to the Data Record⁴:

- *Track 1 Data* (if present) and *Track 2 Data*
- $DD_{\text{CARD,TRACK1}}$ (if *Track 1 Data* present) and $DD_{\text{CARD,TRACK2}}$
- *PayPass Third Party Data* (if present)
- *Application Label* (if present), *Application Preferred Name* (if present), *Issuer Code Table Index* (if present)
- *Language Preference* (if present)

The *PayPass* reader must hand over control to the terminal.



Note Depending on the functionality supported by the terminal, not all the data objects in the Data Record have to be transferred to the terminal. The *Track 1 Data* (if present) and *Track 2 Data* are mandatory, the other data objects are optional. The *PayPass* reader must however be able to provide the full Data Record during sample testing.

⁴ The format of the Data Record is implementation dependent. If the *PayPass* reader uses the TLV format, then implementation specific values may be used for the tags of $DD_{\text{CARD,TRACK1}}$ and $DD_{\text{CARD,TRACK2}}$.

4 *Card Interoperability Requirements*

This chapter contains a list of card requirements to achieve interoperability at the application layer for *PayPass – Mag Stripe*.

4.1 DOL Handling

A *PayPass – Mag Stripe* card may include Data Object Lists in the response messages to the *PayPass* reader.

- 4.1.1.1 The card must allow the *PayPass* reader to fill the requested data fields with hexadecimal zeros.
- 4.1.1.2 If the card returns a *UDOL*, then it must always include the *UN* entry in the *UDOL*.
- 4.1.1.3 If the card returns a *PDOL*, then it must include only tags belonging to data objects identified in Chapter 5 of PART II as having the reader as source.

4.2 Exception Processing

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

4.3 ATC

The *PayPass – Mag Stripe* card must maintain an *ATC*.

- 4.3.1.1 The *ATC* must be binary coded on two bytes.
- 4.3.1.2 The card must increment the *ATC* by 1 when it receives a valid GET PROCESSING OPTIONS command.
- 4.3.1.3 When the *ATC* reaches the value 'FFFF', the *PayPass – Mag Stripe* application must be disabled. In this case the card must return the status bytes '6985' in response to the GET PROCESSING OPTIONS command.

4.4 Select PPSE Command Processing

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

- 4.4.1.1 All *PayPass* cards must provide support for the PPSE.
- 4.4.1.2 The card must include in the *FCI Issuer Discretionary Data* of the *FCI* of the PPSE all the *PayPass* applications supported by the card as specified in Table 3.
- 4.4.1.3 If the card is blocked, then the card must return the status bytes '6A81' (Function not supported) to the *PayPass* reader.
- 4.4.1.4 If the PPSE is blocked, then the card must return the status bytes '6283' (Selected file invalidated) to the *PayPass* reader.

4.5 Select AID Command Processing

This section specifies the behavior of the *PayPass – Mag Stripe* card when it receives a SELECT command with a file name different from the PPSE.

- 4.5.1.1 The *PayPass – Mag Stripe* application must support the SELECT command, as specified in Section 1.2.1 of PART II.
- 4.5.1.2 If *Lc* is different from the actual length of the *AID* included in the value field of the command message, then the card must return the status bytes '6700' (Wrong length).
- 4.5.1.3 The card must retrieve the *AID* from the data field of the command message. If the *AID* is not supported by the card, then the card must return the status bytes '6A82' (File not found).
- 4.5.1.4 If the card is blocked, then the card must return the status bytes '6A81' (Function not supported).
- 4.5.1.5 If the selected application is blocked, then the card must return the status bytes '6283' (Selected file invalidated).
- 4.5.1.6 The card must return the *FCI* in the response message, as specified in Section 1.2.2 of PART II.

4.6 Get Processing Options Command Processing

- 4.6.1.1 The card must support the GET PROCESSING OPTIONS command as specified in Section 2.3.2 of PART II.
- 4.6.1.2 If the card did not provide a *PDOL*, then the card must check that the data field contains the value '8300'. If this is not the case, then the card must return the status bytes '6985' (Conditions of use not satisfied).
- 4.6.1.3 If the card provided a *PDOL*, then the card must check that 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 must return the status bytes '6700' (Wrong length).
- 4.6.1.4 The card must update the *ATC* as specified in Section 4.3 of PART II.
- 4.6.1.5 The card must return the *Response Message Template* (tag '77') in the response message, as specified in Section 2.3.3 of PART II.

4.7 Read Record Command Processing

- 4.7.1.1 The card must support the READ RECORD command as specified in Section 2.4.2 of PART II. The card may support other coding formats for P1-P2.
- 4.7.1.2 All records referenced in the *AFL* must be readable without conditions by the READ RECORD command coded as specified in Section 2.4.2 of PART II.
- 4.7.1.3 For all records referenced in the *AFL*, the response message must follow the coding as specified in Section 2.4.3 of PART II.
- 4.7.1.4 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 2.4.2 of PART II and if the response message follows the coding as specified in Section 2.4.3 of PART II.

4.8 Compute Cryptographic Checksum Command Processing

- 4.8.1.1 The card must support the COMPUTE CRYPTOGRAPHIC CHECKSUM command as specified in Section 2.2.2 of PART II.
- 4.8.1.2 The card must verify if this is the first time the COMPUTE CRYPTOGRAPHIC CHECKSUM command has been received after a successful GET PROCESSING OPTIONS command. If this is not the case, then the card must return the status bytes '6985' (Conditions of use not satisfied).
- 4.8.1.3 If the card did not provide a UDOL, then the card must verify if the length of the command message data field is four bytes. If this is not the case, then the card must return the status bytes '6700' (Wrong length).
- 4.8.1.4 If the card provided a UDOL to the PayPass reader, then the card must 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 must return the status bytes '6700' (Wrong length).
- 4.8.1.5 The card must return in the response message the *Response Message Template* (tag '77') as specified in Section 2.2.3 of PART II.
- 4.8.1.6 The card must include the CVC3_{TRACK2} (tag '9F61') and the ATC (tag '9F36') in the *Response Message Template* (tag '77').
- 4.8.1.7 If the card provided the Track 1 Data (tag '56') to the PayPass reader in the response to the READ RECORD command, then the card must include the CVC3_{TRACK1} (tag '9F60') in the *Response Message Template* (tag '77').

4.9 Card Personalization Requirements

This section lists some specific personalization requirements for *PayPass – Mag Stripe* cards.

- 4.9.1.1 The *AFL* must be personalized with the value '08010100'.
- 4.9.1.2 The *AIP* must be personalized with the value '0000'.
- 4.9.1.3 If the card includes the *Track 1 Data* in record 1 of SFI 1, then the card must also include $PCVC_{TRACK1}$, $PUNATC_{TRACK1}$ and $NATC_{TRACK1}$.
- 4.9.1.4 If present, then $PCVC3_{TRACK1}$ and $PUNATC_{TRACK1}$ must only have non-zero bits that refer to available positions in the discretionary data field of *Track 1 Data*. All other bits must be set to zero.
- 4.9.1.5 $PCVC3_{TRACK2}$ and $PUNATC_{TRACK2}$ must only have non-zero bits that refer to available positions in the discretionary data field of *Track 2 Data*. All other bits must be set to zero.
- 4.9.1.6 If $NATC_{TRACK1}$ and $PUNATC_{TRACK1}$ are present, then t_{TRACK1} must be less than or equal to k_{TRACK1} .
- 4.9.1.7 t_{TRACK2} must be less than or equal to k_{TRACK2} .
- 4.9.1.8 If $PCVC3_{TRACK1}$ is present, then q_{TRACK1} must be greater than or equal to 3.
- 4.9.1.9 q_{TRACK2} must be greater than or equal to 3.
- 4.9.1.10 $k_{TRACK2} - t_{TRACK2}$ (i.e. n_{UN}) must be less than or equal to 8.
- 4.9.1.11 If $NATC_{TRACK1}$ and $PUNATC_{TRACK1}$ are present, then $k_{TRACK1} - t_{TRACK1}$ must be equal to $k_{TRACK2} - t_{TRACK2}$.

5 *Data Objects*

This chapter describes the data objects used in the *PayPass* – Mag Stripe application.

5.1 Data Object Format

Data objects moved from the *PayPass* card to the *PayPass* reader are TLV encoded. Data objects moved from the *PayPass* reader to the *PayPass* card are identified either by a Data Object List (DOL) sent to the *PayPass* reader by the *PayPass* card, or by the definition of the command message.

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

Data objects 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), must 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.

5.2 DOL Handling

To minimize processing in the card, the data fields of the command messages are not TLV encoded. The application on the card indicates the requested data, including format and length, by sending a Data Object List (DOL) to the *PayPass* reader. DOLs used in this specification include the *PDOL* used with the GET PROCESSING OPTIONS command and the *UDOL* used with the COMPUTE CRYPTOGRAPHIC CHECKSUM command.

The following rules apply for the concatenation of data objects listed in the DOL:

- 5.2.1.1 If the tag of any data object identified in the DOL is unknown to the *PayPass* reader, the *PayPass* reader must provide a data object with the length specified and a value of all hexadecimal zeros.
- 5.2.1.2 If the tag of any data object identified in the DOL is known to the *PayPass* reader, but the data object represents optional static data that is not present in the card, then the *PayPass* reader must provide a data object with the length specified and a value of all hexadecimal zeros.
- 5.2.1.3 The *PayPass* reader must concatenate the list of data objects in the sequence in which the corresponding data objects appear in the DOL.
- 5.2.1.4 If the length specified in the DOL entry is less than the length of the actual data object, then the leftmost bytes of the data object must be truncated if the data object has numeric (n) format, or the rightmost bytes of the data must be truncated for any other format.
- 5.2.1.5 If the length specified in the DOL entry is greater than the length of the actual data, then the *PayPass* reader must pad the actual data:
 - With leading hexadecimal zeroes if the data has numeric (n) format
 - With trailing hexadecimal 'FF's if the data has compressed numeric (cn) format
 - With trailing hexadecimal zeroes for any other format (an or ans)

5.3 Bitmap

PayPass – Mag Stripe uses bitmaps to indicate positions in the discretionary data field. These bitmaps are used when the *PayPass* reader needs to put data into the discretionary data: the bits indicate which positions certain data must be loaded into.

Figure 4 indicates the numbering of the different positions in the discretionary data. In this example there are m positions within the discretionary data field, labeled p_1 to p_m .

Figure 4—Numbering of Positions within the 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 bitmap refers to a position in the discretionary data. The least significant bit of the bitmap, i.e. the rightmost bit b_1 , corresponds to position p_1 as indicated in Figure 5.

Figure 5—Relation between Discretionary Data and Bitmap

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

b_q	b_{q-1}	b_{q-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
Bitmap														

The bitmap is composed of a number of bytes, and therefore the number of bits in the bitmap is always a multiple of 8. To accommodate all the positions in a field, the number of bytes in the bitmap will normally contain more bits than the number of positions. If the number of bits in the bitmap is denoted by q , then

$$q = (r + 1) * 8, \text{ where } r \text{ is the integer quotient of } (m-1)/8.$$

For *Track 2 Data* m_{TRACK2} is a maximum of 13 digits, resulting in a bitmap 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.

An example is given in Figure 6, for $m_{\text{TRACK2}}=13$, $t_{\text{TRACK2}}=2$ and $PUNATC_{\text{TRACK2}} = '031A'$, referring to position $p_{10}p_9p_5p_4p_2$. Based on this, k_{TRACK2} equals 5 and n_{UN} equals 3.

Figure 6—Example $PUNATC_{TRACK2} = '031A'$

Discretionary Data															
p ₁₃	p ₁₂	p ₁₁	p₁₀	p₉	p ₈	p ₇	p ₆	p₅	p₄	p ₃	p₂	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’			
Bitmap = ‘031A’															

5.4 Data Object Dictionary

This section lists the data objects interchanged between card and *PayPass* reader during a *PayPass* – Mag Stripe transaction.

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

5.4.1 ADF Name

Tag: '4F'

Source: Card

Presence: Mandatory

Format: b, 5–16 bytes

Description: The *ADF Name* identifies the *PayPass* application in the card. The value of the *ADF Name* is the *AID* of the *PayPass* application in the card as specified in [ISO/IEC 7816-5].

The *AID* is comprised of the following two parts:

- The Registered Application Provider Identifier (RID) of 5 bytes. ISO assigns this number.
- A Proprietary Application Identifier Extension (PIX) of up to 11 (hexadecimal) bytes. The *RID* owner assigns this number.

The *ADF Name* is included in an *Application Template* (tag '61') in the *FCI Issuer Discretionary Data* (tag 'BF0C') of the PPSE. The value of the *ADF Name* must be the same as the value of the *DF Name* (tag '84') included in the *FCI* of the corresponding application.

5.4.2 AFL

<i>Tag:</i>	'94'
<i>Source:</i>	Card
<i>Presence:</i>	Mandatory
<i>Format:</i>	b, var. up to 252 bytes
<i>Description:</i>	The <i>AFL</i> identifies the files and records containing the data to be used for the transaction. The <i>AFL</i> is returned by the card in the response message of the GET PROCESSING OPTIONS command.

5.4.3 AID (Reader)

<i>Tag:</i>	'9F06'
<i>Source:</i>	Reader
<i>Presence:</i>	Mandatory
<i>Format:</i>	b, 5–16 bytes
<i>Description:</i>	The <i>AID</i> with tag '9F06' identifies the <i>PayPass</i> application in the <i>PayPass</i> reader as specified in [ISO/IEC 7816-5].

5.4.4 AIP

<i>Tag:</i>	'82'
<i>Source:</i>	Card
<i>Presence:</i>	Mandatory
<i>Format:</i>	b, 2 bytes
<i>Description:</i>	The <i>AIP</i> indicates the ability of the card to support specific functions in the application. The <i>AIP</i> is returned by the card in the response message of the GET PROCESSING OPTIONS command.

5.4.5 Application Label

<i>Tag:</i>	'50'
<i>Source:</i>	Card
<i>Presence:</i>	Optional
<i>Format:</i>	an, 1–16 bytes
<i>Description:</i>	The <i>Application Label</i> is the mnemonic associated with the <i>AID</i> according to [ISO/IEC 7816-5]. If present, then the <i>Application Label</i> is included in the <i>FCI</i> returned by the card in the response to the <i>SELECT AID</i> command.

5.4.6 Application Preferred Name

<i>Tag:</i>	'9F12'
<i>Source:</i>	Card
<i>Presence:</i>	Optional
<i>Format:</i>	an, 1–16 bytes
<i>Description:</i>	The <i>Application Preferred Name</i> is the preferred mnemonic associated with the <i>AID</i> . If present, then the <i>Application Preferred Name</i> is included in the <i>FCI</i> returned by the card in the response to the <i>SELECT AID</i> command.

5.4.7 Application Priority Indicator

<i>Tag:</i>	'87'
<i>Source:</i>	Card
<i>Presence:</i>	Mandatory
<i>Format:</i>	b, 1 byte
<i>Description:</i>	The <i>Application Priority Indicator</i> indicates the priority of the associated application in a directory. Refer to Table 5 of PART II for the coding of the <i>Application Priority Indicator</i> . The <i>Application Priority Indicator</i> is included in the <i>Application Template</i> (tag '61'), in the <i>FCI Issuer Discretionary Data</i> (tag 'BF0C') of the PPSE and optionally in the <i>FCI</i> of the corresponding application.

5.4.8 Application Version Number (Card)

Tag: '9F6C'

Source: Card

Presence: Optional

Format: b, 2 bytes

Description: The *Application Version Number (Card)* is the version number assigned by the payment system for the specific *PayPass – Mag Stripe* application in the card. If present, then the *Application Version Number (Card)* must be present in the file read using the READ RECORD command.

5.4.9 Application Version Number (Reader)

Tag: '9F6D'

Source: Reader

Presence: Mandatory

Format: b, 2 bytes

Description: The *Application Version Number (Reader)* is the version number assigned by the payment system for a specific *PayPass – Mag Stripe* application in the *PayPass* reader.

5.4.10 ATC

Tag: '9F36'

Source: Card

Presence: Mandatory

Format: b, 2 bytes

Description: The *ATC* is a dynamic counter maintained by the *PayPass* application on the card. The *ATC* is used by the card to calculate the dynamic CVC. The *ATC* is returned by the card in the response message of the COMPUTE CRYPTOGRAPHIC CHECKSUM command.

5.4.11 CVC3_{TRACK1}

<i>Tag:</i>	'9F60'
<i>Source:</i>	Card
<i>Presence:</i>	Conditional (If the <i>Track 1 Data</i> is present, then the CVC3 _{TRACK1} must also be present.)
<i>Format:</i>	b, 2 bytes
<i>Description:</i>	CVC3 _{TRACK1} is a 2 byte cryptogram returned by the card in the response to the COMPUTE CRYPTOGRAPHIC CHECKSUM command.

5.4.12 CVC3_{TRACK2}

<i>Tag:</i>	'9F61'
<i>Source:</i>	Card
<i>Presence:</i>	Mandatory
<i>Format:</i>	b, 2 bytes
<i>Description:</i>	CVC3 _{TRACK2} is a 2 byte cryptogram returned by the card in the response to the COMPUTE CRYPTOGRAPHIC CHECKSUM command.

5.4.13 DD_{CARD,TRACK1}

<i>Tag:</i>	–
<i>Source:</i>	Reader
<i>Presence:</i>	Mandatory
<i>Format:</i>	ans, var. up to 56 bytes
<i>Description:</i>	If <i>Track 1 Data</i> is present, then DD_{CARD,TRACK1} contains a copy of the discretionary data field of <i>Track 1 Data</i> as returned by the card in the file read using the READ RECORD command (i.e. without UN, ATC, CVC3_{TRACK1} and n_{UN} included). If <i>Track 1 Data</i> is not present, then the value of DD_{CARD,TRACK1} is undefined.

5.4.14 DD_{CARD,TRACK2}

Tag: –

Source: Reader

Presence: Mandatory

Format: n, var. up to 8 bytes

Description: DD_{CARD,TRACK2} contains a copy of the discretionary data field of *Track 2 Data* as returned by the card in the file read using the READ RECORD command (i.e. without UN, ATC, CVC3_{TRACK2} and n_{UN} included).

5.4.15 Default UDOL

Tag: --

Source: Reader

Presence: Mandatory

Format: b, 3 bytes

Description: The *Default UDOL* is the UDOL to be used for constructing the value field of the COMPUTE CRYPTOGRAPHIC CHECKSUM command if the UDOL on the card is not present. The *Default UDOL* must always be present and must contain as its only entry the tag and length of UN. The value of the *Default UDOL* must therefore be: '9F6A04'.

5.4.16 DF Name

Tag: '84'

Source: Card

Presence: Mandatory

Format: b, 5–16 bytes

Description: The value of the *DF Name* (tag '84') identifies the name of the DF as described in [ISO/IEC 7816-4]. The *DF Name* is included in the FCI returned by the card in the response to the SELECT AID command. The value of the *DF Name* must be the same as the value of the *ADF Name* (tag '4F') of the corresponding application.

5.4.17 Issuer Code Table Index

<i>Tag:</i>	'9F11'
<i>Source:</i>	Card
<i>Presence:</i>	Optional
<i>Format:</i>	n 2, 1 byte
<i>Description:</i>	The value of the <i>Issuer Code Table Index</i> indicates the code table (according to [ISO 8859]) used to display the <i>Application Preferred Name</i> . If present, the <i>Issuer Code Table Index</i> is included in the <i>FCI</i> returned by the card in the response to the SELECT AID command.

5.4.18 Language Preference

<i>Tag:</i>	'5F2D'
<i>Source:</i>	Card
<i>Presence:</i>	Optional
<i>Format:</i>	an 2, 2–8 bytes
<i>Description:</i>	The value of the <i>Language Preference</i> represents 1-4 languages stored in order of preference, each represented by 2 alphabetical characters as defined by [ISO 639]. If present, the <i>Language Preference</i> is included in the <i>FCI</i> returned by the card in the response to the SELECT AID command.

5.4.19 NATC_{TRACK1}

<i>Tag:</i>	'9F64'
<i>Source:</i>	Card
<i>Presence:</i>	Conditional (If <i>Track 1 Data</i> is present, then also NATC _{TRACK1} must be present)
<i>Format:</i>	b, 1 byte
<i>Description:</i>	The value of NATC _{TRACK1} represents the number of digits of the <i>ATC</i> to be included in the discretionary data field of the <i>Track 1 Data</i> . If present, then the NATC _{TRACK1} must be present in the file read using the READ RECORD command.

5.4.20 NATC_{TRACK2}

<i>Tag:</i>	'9F67'
<i>Source:</i>	Card
<i>Presence:</i>	Mandatory
<i>Format:</i>	b, 1 byte
<i>Description:</i>	The value of NATC _{TRACK2} represents the number of digits of the ATC to be included in the discretionary data field of the <i>Track 2 Data</i> . NATC _{TRACK2} must be present in the file read using the READ RECORD command.

5.4.21 PayPass Third Party Data

<i>Tag:</i>	'9F6E'
<i>Source:</i>	Card
<i>Presence:</i>	Optional
<i>Format:</i>	b, 5–32 bytes
	The value field must be coded as follows:
	Country code according to [ISO 3166] n 3, 2 bytes
	Unique identifier assigned by MasterCard b, 2 bytes
	Proprietary data b, 1–28 bytes
<i>Description:</i>	The <i>PayPass Third Party Data</i> contains proprietary information from a third party. If present, the <i>PayPass Third Party Data</i> must be present in the file read using the READ RECORD command.
	The value field of the <i>PayPass Proprietary Data</i> is not interpreted by the <i>PayPass</i> reader. If returned by the card, the <i>PayPass Proprietary Data</i> is stored in the Data Record to be sent to the terminal.

5.4.22 PCVC3_{TRACK1}

<i>Tag:</i>	'9F62'
<i>Source:</i>	Card
<i>Presence:</i>	Conditional (If <i>Track 1 Data</i> is present, then <i>PCVC3_{TRACK1}</i> must also be present)
<i>Format:</i>	b, 6 bytes
<i>Description:</i>	<i>PCVC3_{TRACK1}</i> indicates to the <i>PayPass</i> reader the positions in the discretionary data field of the <i>Track 1 Data</i> where the <i>q_{TRACK1} CVC3_{TRACK1}</i> digits must be copied. If present, the <i>PCVC3_{TRACK1}</i> must be present in the file read using the READ RECORD command.

5.4.23 PCVC3_{TRACK2}

<i>Tag:</i>	'9F65'
<i>Source:</i>	Card
<i>Presence:</i>	Mandatory
<i>Format:</i>	b, 2 bytes
<i>Description:</i>	<i>PCVC3_{TRACK2}</i> indicates to the <i>PayPass</i> reader the positions in the discretionary data field of the <i>Track 2 Data</i> where the <i>q_{TRACK2} CVC3_{TRACK2}</i> digits have to be copied. The <i>PCVC3_{TRACK2}</i> must be present in the file read using the READ RECORD command.

5.4.24 PDOL

<i>Tag:</i>	'9F38'
<i>Source:</i>	Card
<i>Presence:</i>	Optional
<i>Format:</i>	b, var. up to 252 bytes
<i>Description:</i>	The <i>PDOL</i> is the Data Object List that specifies the reader resident data objects to be included in the data field of the GET PROCESSING OPTIONS command. If present, the <i>PDOL</i> is included in the <i>FCI</i> returned by the card to the <i>PayPass</i> reader in the response to the SELECT AID command.

5.4.25 PUNATC_{TRACK1}

<i>Tag:</i>	‘9F63’
<i>Source:</i>	Card
<i>Presence:</i>	Conditional (If <i>Track 1 Data</i> is present, then <i>PUNATC_{TRACK1}</i> must also be present)
<i>Format:</i>	b, 6 bytes
<i>Description:</i>	<i>PUNATC_{TRACK1}</i> indicates to the <i>PayPass</i> reader the positions in the discretionary data field of the <i>Track 1 Data</i> where the <i>n_{UN}</i> <i>UN</i> digits and <i>t_{TRACK1}</i> <i>ATC</i> digits have to be copied. If present, then the <i>PUNATC_{TRACK1}</i> must be present in the file read using the READ RECORD command.

5.4.26 PUNATC_{TRACK2}

<i>Tag:</i>	‘9F66’
<i>Source:</i>	Card
<i>Presence:</i>	Mandatory
<i>Format:</i>	b, 2 bytes
<i>Description:</i>	<i>PUNATC_{TRACK2}</i> indicates to the <i>PayPass</i> reader the positions in the discretionary data field of the <i>Track 2 Data</i> where the <i>n_{UN}</i> <i>UN</i> digits and <i>t_{TRACK2}</i> <i>ATC</i> digits have to be copied. The <i>PUNATC_{TRACK2}</i> must be present in the file read using the READ RECORD command.

5.4.27 Track 1 Data

Tag	'56'
Source:	Card
Presence:	Optional
Format:	ans, var. up to 76 bytes
	The <i>Track 1 Data</i> contains the data objects of the track 1 according to [ISO/IEC 7813] Structure B, excluding start sentinel, end sentinel and LRC.
	Format Code ('42' (B))1 byte
	Identification Number (PAN)var. up to 19 bytes
	Field Separator ('5E' (^))1 byte
	Name (see [ISO/IEC 7813])2 to 26 bytes
	Field Separator ('5E' (^))1 byte
	Expiration Date (YYMM)4 bytes
	Service Code3 bytes
	Discretionary Databalance of available bytes
Description:	The <i>Track 1 Data</i> may be present in the file read using the READ RECORD command. The <i>PayPass</i> reader copies the required digits of the <i>UN</i> , <i>CVC3_{TRACK1}</i> , <i>ATC</i> and <i>n_{UN}</i> into the discretionary data field of the <i>Track 1 Data</i> and stores the modified <i>Track 1 Data</i> in the Data Record to be sent to the terminal.

5.4.28 Track 2 Data

Tag: '9F6B'

Source: Card

Presence: Mandatory

Format: b, var. up to 19 bytes

The *Track 2 Data* contains the data objects 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 ('D')	b
Expiration Date (YYMM)	n 4
Service Code	n 3
Discretionary Data	n, balance of available digits
Padded with 'F' to ensure whole bytes.	

Description: The *Track 2 Data* must be present in the file read using the READ RECORD command. The *PayPass* reader copies the required digits of the *UN*, *CVC3_{TRACK2}*, *ATC* and *n_{UN}* into the discretionary data field of the *Track 2 Data* and stores the modified *Track 2 Data* in the Data Record to be sent to the terminal.

5.4.29 UDOL

Tag: '9F69'

Source: Card

Presence: Optional

Format: b, var. up to 252 bytes

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 tag and length of *UN*. The *UDOL* is not mandatory for the card. There will always be a *Default UDOL*, containing as its only entry the tag and length of *UN*. If the card has its own *UDOL*, then it must be present in the file read using the READ RECORD command.

5.4.30 UN

Tag: '9F6A'

Source: Reader

Presence: Mandatory

Format: n 8 (the 8-n_{UN} most significant digits must be set to zero)

Description: Unpredictable number generated by the *PayPass* reader. *UN* is passed to the card in the data field of the COMPUTE CRYPTOGRAPHIC CHECKSUM command.

Ideally *UN* should be truly unpredictable (even given access to all previous such numbers generated by the *PayPass* reader) and it should be infeasible for an attacker to control the next *UN* that the *PayPass* reader generates.

If the *PayPass* reader does not have access to a high quality random number generator, then one way of achieving the above objective is to make each *UN* a function of previous CVC3s. This function could be, in descending order of preference, a keyed encipherment function⁵, a hash function, or an exclusive-OR.

⁵ Keys used for this dedicated purpose need not be stored in a security module if the reader does not have access to such a module. Ideally keys should be different for each reader.

PART III – Card Specification

PART III describes the implementation of the PPSE and *PayPass* – Mag Stripe application on a *PayPass* card.

1	Introduction.....	59
1.1	Support for the PPSE	59
1.2	Personalization Process.....	60
1.3	State Machine	60
1.4	Functional Overview.....	62
1.5	Structure of the Document.....	63
2	PPSE Application.....	65
2.1	Introduction.....	65
2.2	Application State Machine.....	65
2.3	Command Processing.....	66
2.3.1	C-APDU Recognition	66
2.3.2	C-APDU Acceptance	67
2.3.3	SELECT PPSE	68
2.3.4	LOOP BACK	70
3	<i>PayPass</i> – Mag Stripe Application.....	73
3.1	Application State Machine.....	73
3.2	Variables	76
3.2.1	Internal Flags in Non Volatile Memory	76
3.2.2	ATC	76
3.3	C-APDU Processing	76
3.3.1	C-APDU Recognition	76
3.3.2	C-APDU Acceptance	77
3.3.3	C-APDU Processing	78
3.4	SELECT.....	79
3.4.1	Command Message.....	79
3.4.2	Response Message	79
3.4.3	Processing	79
3.4.4	Destination States	80
3.5	GET PROCESSING OPTIONS	81
3.5.1	Command Message.....	81
3.5.2	Response Message	81
3.5.3	Processing	82
3.5.4	Destination States	84
3.6	READ RECORD.....	84
3.6.1	Command Message.....	84
3.6.2	Response Message	85

3.6.3	Processing	85
3.6.4	Destination States.....	86
3.7	COMPUTE CRYPTOGRAPHIC CHECKSUM.....	87
3.7.1	Command Message	87
3.7.2	Data Field Returned in the Response Message	87
3.7.3	Processing	88
3.7.4	Destination States.....	89
3.8	Dynamic CVC3.....	90
3.8.1	KD _{CVC3}	90
3.8.2	Dynamic CVC3 Generation	91
3.8.3	IVCVC3 Generation	91
3.9	Data Objects Dictionary	92
3.9.1	IVCVC3 _{TRACK1}	92
3.9.2	IVCVC3 _{TRACK2}	92
4	PayPass Application Personalization.....	93
4.1	Personalization Processing Flow.....	93
4.2	Personalization State Machine	94
4.2.1	Personalization States.....	94
4.2.2	State Diagram.....	95
4.3	Personalization Command Processing	96
4.3.1	C-APDU Recognition	96
4.3.2	C-APDU Acceptance	97
4.3.3	SELECT Command.....	97
4.3.4	GET DATA Command.....	98
4.3.5	STORE DATA Command.....	100
4.4	Personalization Data.....	104
4.4.1	DGI '0101' – Data Objects Referenced in the AFL.....	104
4.4.2	DGI 'A001' – Persistent Data Objects for CVC3 Generation	105
4.4.3	DGI 'A002' – Secret Key	105
4.5	Cryptography for Personalization	106
4.5.1	KD _{PERSO}	106
4.5.2	MAC for STORE DATA.....	106
4.5.3	Encryption/Decryption.....	106
5	PayPass Application Pre-Personalization	107
5.1	Application Selection Data Objects	107
5.2	GET PROCESSING OPTIONS Response	108
5.3	Secret Key Information	108
5.4	KD _{PERSO}	108
5.5	ATC.....	108
Annex A	Example.....	109
A.1	Pre-Personalization Data	109
A.2	Personalization Data.....	110
A.3	Personalization Commands	111

A.4	Transaction Processing	112
Annex B	MAC Algorithm.....	115
Annex C	Multi-Application Manager	117
C.1	Introduction.....	117
C.2	The Multi-Application Manager	117
C.3	Processing of a C-APDU	118
C.4	Processing of the SELECT Command	119

1 Introduction

The *PayPass* – Mag Stripe card specification aims to provide a definition of the behavior of a contactless card containing the *PayPass* – Mag Stripe application with support for the PPSE. This document is generic in that it does not intend to include or exclude any particular platform.

1.1 Support for the PPSE

This specification views support for the PPSE as another application on the card. Therefore, the *PayPass* – Mag Stripe card contains two applications: the PPSE application and the *PayPass* – Mag Stripe 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.

Annex C provides a detailed description of the multi-application model used by the *PayPass* – Mag Stripe card. The following chapters contain the specification of the individual PPSE and *PayPass* – Mag Stripe applications assuming the multi-application model specified in Annex C.

The PPSE application supports two commands: the SELECT command with file name equal to “2PAY.SYS.DDF01” and the LOOP BACK command. In the response to the SELECT command, the card returns the list of applications supported over the contactless interface.

In addition to the SELECT command, the PPSE application also provides support for the loop-back functionality by means of the LOOP BACK command. Refer to Chapter 2 of PART III for more information about the loop-back functionality.

1.2 Personalization Process

The *PayPass* – Mag Stripe card implementation specification also includes the personalization process of the *PayPass* – Mag Stripe application. The personalization process described in this document assumes the *PayPass* – Mag Stripe cards are pre-personalized. The pre-personalization process loads the *PayPass* – Mag Stripe and PSSE applications and establishes the file and data structure. In addition, certain data which have the same value for all cards in a personalization batch must be placed onto the *PayPass* – Mag Stripe card.

Chapter 4 of Part PART III specifies the personalization process of the *PayPass* – Mag Stripe application. The PPSE application of the *PayPass* – Mag Stripe card does not need to be personalized. The *FCI* of the PPSE application is stored in the card during pre-personalization.

Chapter 5 of PART III lists the pre-personalization requirements, but does not include the description of the pre-personalization process, which is implementation dependent.

1.3 State Machine

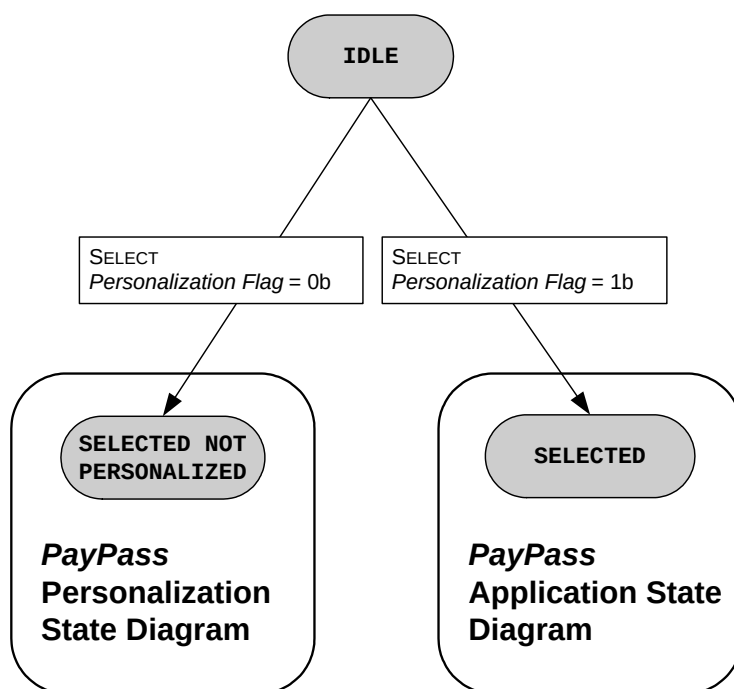
The state machine of the *PayPass* – Mag Stripe application differs depending on whether it is already personalized or not. Therefore, the *PayPass* – Mag Stripe application must know if it has been personalized, or not. To implement this feature, the application uses the *Personalization Flag* – a persistent data object consisting of only one bit, interpreted as described in Table 19.

Table 19—*Personalization Flag*

Personalization Status	<i>Personalization Flag</i>
<i>PayPass</i> application not personalized	0b
<i>PayPass</i> application personalized	1b

The processing of the SELECT command depends on the value of the *Personalization Flag*. When the application is not yet personalized, the *PayPass* application will go to the **SELECTED NOT PERSONALIZED** state after processing the SELECT command. When the application is personalized, the *PayPass* application will go to the **SELECTED** state. The processing of the SELECT command is shown in Figure 7.

Figure 7—Processing of SELECT Command



The *Personalization Flag* is managed by the *PayPass* application as follows:

- When the application is not yet personalized, the *Personalization Flag* has value 0b and, as a consequence, the personalization state machine will apply. Having successfully processed the SELECT C-APDU in the state **IDLE**, the application will go to the **SELECTED NOT PERSONALIZED** state.
- The last personalization C-APDU processed by the *PayPass* – Mag Stripe application, i.e. the STORE DATA, sets the *Personalization Flag* to 1b.
- When the application is already personalized, the *Personalization Flag* has value 1b and, as a consequence, the application state machine will apply. Having successfully processed the SELECT C-APDU in the state **IDLE**, the *PayPass* – Mag Stripe application will go to the **SELECTED** state.



Note

The *PayPass* – Mag Stripe application never resets the *Personalization Flag*. The last STORE DATA processed by the application irreversibly sets the *Personalization Flag* to 1b, disabling the personalization commands. Refer to Section 4.3.5 of PART III for the description of the STORE DATA command.



Note

The *Personalization Flag* is an internal flag that has no external visibility. The functionality achieved with the *Personalization Flag* may be implemented in another way, providing that the external behavior of the *PayPass* – Mag Stripe application is indistinguishable from the behavior defined in this document.



Note

For this version of the document, no personalization requirements are defined for the PPSE application.

1.4 Functional Overview

This section provides an overview of the functionality of the *PayPass – Mag Stripe* card implementation proposed by MasterCard and conforming to the requirements listed in PART II.

1. The *PayPass – Mag Stripe* card contains only one payment application. When ordering *PayPass – Mag Stripe* cards it must be specified for which product, as the *AID* is loaded during the pre-personalization phase.
2. The PPSE application of the *PayPass – Mag Stripe* card must support the loop-back functionality. The loop-back functionality facilitates compliance testing of the *PayPass – Mag Stripe* card with *[EMV PROTOCOL]*.
3. *PayPass – Mag Stripe* cards must be personalized as specified in Chapter 4 of PART III. The personalization process is based on the EMV Card Personalization Specification (refer to *[EMV PERSON]*) but optimized for the *PayPass – Mag Stripe* application.
4. The personalization of the *PayPass – Mag Stripe* card must be preceded by a pre-personalization process. The data that is assumed to be present prior to the personalization is specified in Chapter 5 of PART III.
5. *PDOL* is not supported.
6. *UDOL* is not supported.
7. The *PayPass – Mag Stripe* card always returns *CVC3TRACK1*.⁶

⁶ The *CVC3TRACK1* will only be taken into account by the *PayPass* reader if the *PayPass – Mag Stripe* card also returns the *Track 1 Data* in the response to the *READ RECORD* command.

1.5 Structure of the Document

The remainder of PART III is organized as follows:

- Chapter 2 specifies the behavior of the PPSE application.
- Chapter 3 contains the specification of the *PayPass* – Mag Stripe card application after it has been personalized (i.e. *Personalization Flag* = 1b).
- Chapter 4 describes the personalization process of the *PayPass* – Mag Stripe card application (i.e. *Personalization Flag* = 0b).
- Chapter 5 lists the pre-personalization requirements.
- Annex A contains an example that explains:
 - How the *PayPass* – Mag Stripe card is personalized.
 - How the dynamic *CVC3_{TRACK1}* and *CVC3_{TRACK2}* are calculated.
 - How the APDU commands and responses are formatted.
- Annex B specifies the MAC algorithm used by the *PayPass* – Mag Stripe application.
- Annex C introduces the PPSE and *PayPass* – Mag Stripe application in a multi-application environment.

2 PPSE Application

2.1 Introduction

This chapter specifies the behavior of the *PayPass* – Mag Stripe 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 mapped on a Dedicated File (DF) – which may or may not be the Master File (MF) – of an ISO 7816-4 compatible file structure.

In addition to the directory function, the PPSE application of the *PayPass* – Mag Stripe card provides support for the loop-back functionality. The 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. The loop-back functionality is used during the compliance testing of the *PayPass* – Mag Stripe card with [EMV PROTOCOL].

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 20.

Table 20—Application States of the PPSE Application

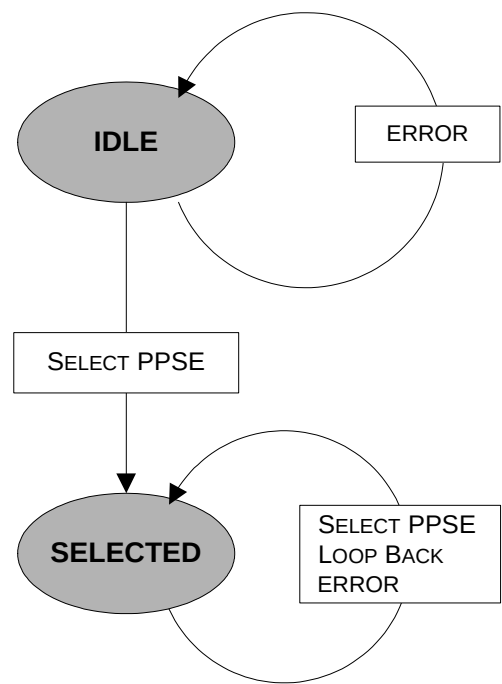
State	Description
IDLE	Application is not currently selected
SELECTED	Application is selected

The PPSE application is in the **IDLE** state 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 **SELECTED** state. The PPSE application remains in the **SELECTED** state until the PPSE application is de-selected (i.e. the *PayPass* – Mag Stripe application is selected or the card is powered-off).

The PPSE application does not change state if an error occurs. An error results in a command response with status bytes other than ‘9000’.

Figure 8 graphically represents the state machine of the PPSE application.

Figure 8—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 *PayPass* reader to the PPSE application. The recognition is based on the CLA and INS byte. Table 21 specifies the CLA and INS coding for the PPSE application.

Table 21—C-APDU Recognition of the PPSE Application

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

The C-APDU recognition procedure takes as input the CLA and INS bytes and produces as output one of the literals specified in the C-APDU column of Table 21.

If the CLA byte of the C-APDU is not one of those listed in Table 21, then the C-APDU recognition procedure returns status bytes ‘6E00’. If the CLA byte of the C-APDU is one of those listed in Table 21, but the combination of the CLA and INS byte of the C-APDU is not one of those listed in Table 21, then the C-APDU recognition procedure returns status bytes ‘6D00’.

When the *PayPass* application has recognized the C-APDU, it performs a validity check on the length fields:

- Consistency between Lc and the length of data sent
- Le (‘00’ or not present)

When the validity check detects an error on the lengths, then the C-APDU recognition procedure returns status bytes ‘6700’.

When the C-APDU is rejected, then 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 22.

Table 22—Acceptance Matrix of the PPSE Application

	IDLE	SELECTED
SELECT PPSE	Accept	Accept
LOOP BACK	Reject	Accept

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

When the C-APDU is rejected, then the PPSE application returns the status bytes ‘6985’ and remains in the current state.

2.3.3 SELECT PPSE

Command Message

The SELECT command message for the PPSE is coded according to Table 23.

Table 23—SELECT PPSE Command Message

Code	Value
CLA	'00'
INS	'A4'
P1	'04'
P2	'00'
Lc	'0E'
Data	'325041592E5359532E4444463031'
Le	'00'

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

Response Message

The data field of the response message contains the *FCI* and is coded according to Table 24.

Table 24—FCI Returned in Response Message of SELECT PPSE

Data Object	Tag	Length
<i>FCI Template</i>	'6F'	33–44
<i>DF Name</i> ('325041592E5359532E4444463031')	'84'	14
<i>FCI Proprietary Template</i>	'A5'	15–26
<i>FCI Issuer Discretionary Data</i>	'BF0C'	12–23
<i>Directory Entry</i>	'61'	10–21
<i>ADF Name</i>	'4F'	5–16
<i>Application Priority Indicator</i> ('01')	'87'	1

The *FCI* of the PPSE is stored in the card during the pre-personalization phase.

Processing

Figure 9 specifies the processing of the SELECT PPSE command.

Symbol 0

If $P1 \neq '04'$ or $P2 \neq '00'$, then the C-APDU is rejected (SW1-SW2 = '6A86').

Symbol 1

If $Lc \neq 14$, then the C-APDU is rejected (SW1-SW2 = '6700').

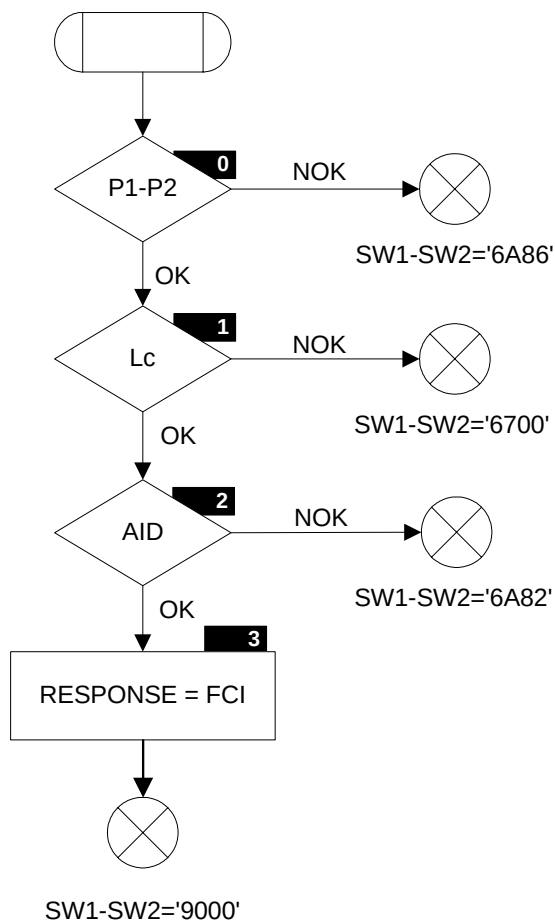
Symbol 2

If the value of 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 3

The PPSE application builds the response message as specified in Table 24.

Figure 9—SELECT PPSE Processing



Destination States

The destination states for the SELECT PPSE command are listed in Table 25.

Table 25—Destination States for SELECT PPSE Command

SW1	SW2	IDLE	SELECTED
'67'	'00'	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 26.

Table 26—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, defining the number of bytes included in the Test Data, may range from 1 to 250. The data field of the command message contains the Test Data that will 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 10 specifies the processing of the LOOP BACK command.

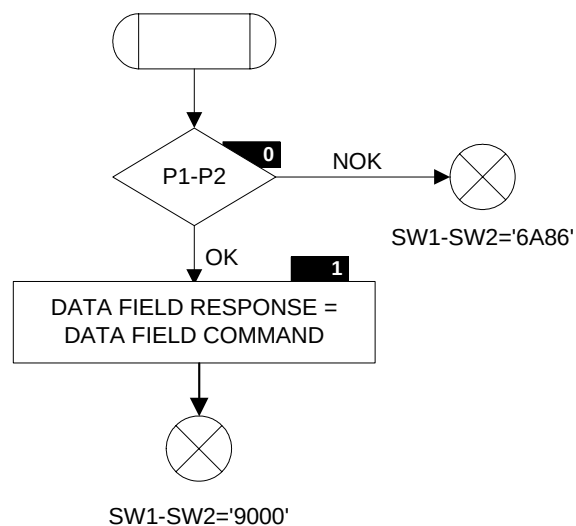
Symbol 0

If $P1 \neq '00'$ or $P2 \neq '00'$, 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.

Figure 10—LOOP BACK Processing



Destination States

The destination states for the LOOP BACK command are listed in Table 27.

Table 27—Destination States for LOOP BACK Command

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

3 PayPass – Mag Stripe Application

This chapter specifies the behavior of the *PayPass* – Mag Stripe card application. The specification aims to provide an unambiguous definition of the behavior of the *PayPass* – Mag Stripe card application so that once personalized all implementations behave in exactly the same way.

For the remainder of the document, the term “*PayPass* application” is used to mean “*PayPass* – Mag Stripe card application”.

3.1 Application State Machine

When the *PayPass* application is in the operational phase (i.e. personalized), its behavior is specified by the application state machine. The application states used in this description are given in Table 28.

Table 28—Application States of the *PayPass* Application

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

The *PayPass* application is in the **IDLE** state if it is not currently activated. There is only one C-APDU which is handled in this state: the SELECT C-APDU which activates the application. The application remains in the **SELECTED** state after the processing of the SELECT command.

The *PayPass* application goes to the **SELECTED** state:

- If it is just activated (i.e. the first SELECT C-APDU).
- When it has already been activated and a new SELECT command is received (i.e. the SELECT C-APDU).
- When a transaction is aborted.

In this state, all the transient data objects are reset (i.e. all the transient information about the previous transaction are lost).

Every transaction starts in the **SELECTED** state. There are three C-APDUs handled in this state:

- READ RECORD
- GET PROCESSING OPTIONS
- SELECT

The READ RECORD C-APDU does not modify the application state. The GET PROCESSING OPTIONS C-APDU initiates a new transaction. The application goes to the **INITIATED** state after the successful processing of the GET PROCESSING OPTIONS C-APDU. The application goes to the **SELECTED** state after the processing of the SELECT command.

A new transaction has been initiated when the application is in the **INITIATED** state. There are three C-APDUs handled in this state:

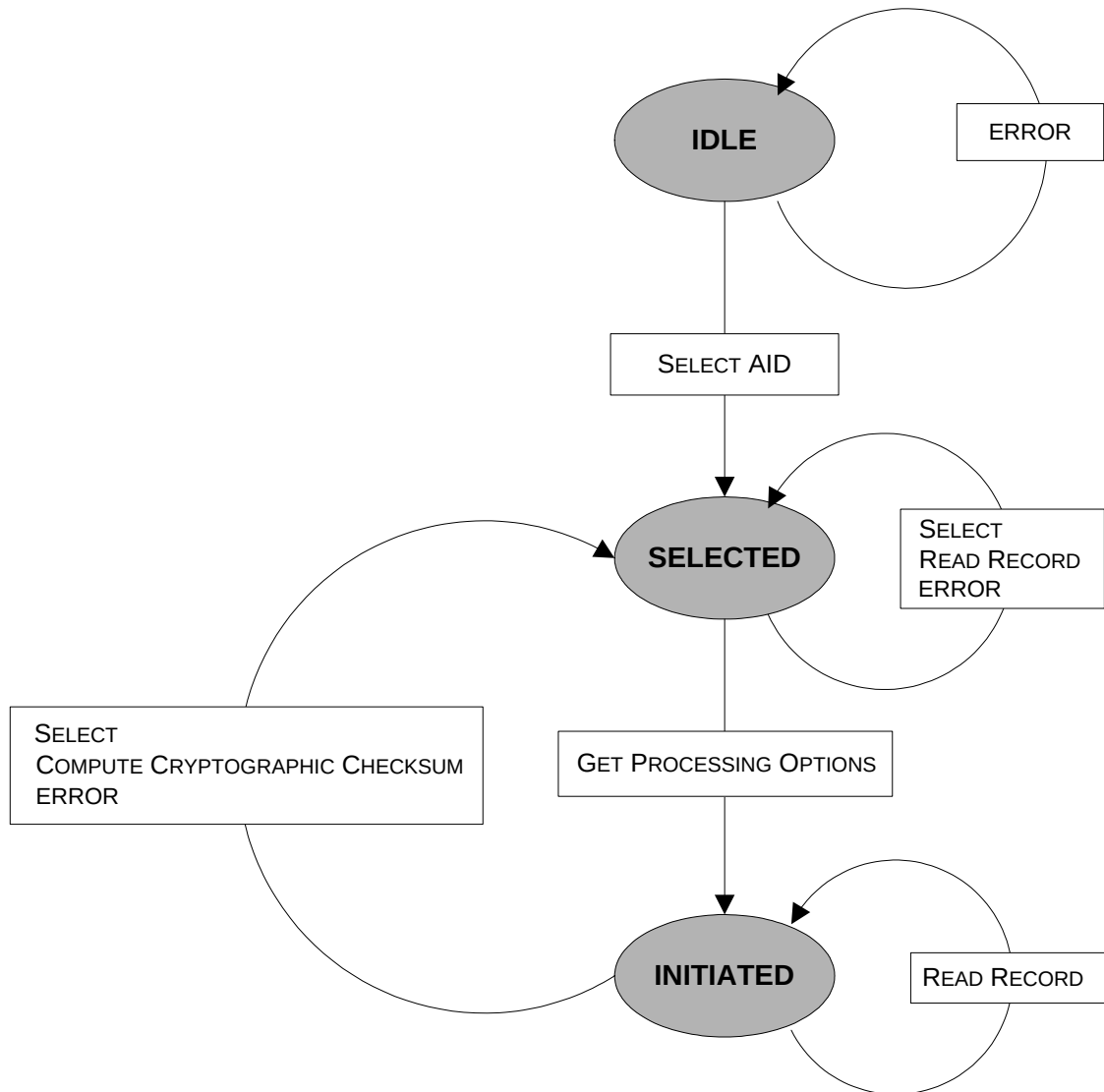
- READ RECORD
- COMPUTE CRYPTOGRAPHIC CHECKSUM
- SELECT

The READ RECORD C-APDU does not modify the application state. The application goes to the **SELECTED** state after the processing of the COMPUTE CRYPTOGRAPHIC CHECKSUM or SELECT command.

The *PayPass* application returns to the **SELECTED** state whenever an error occurs in a state different from the **IDLE** state. The *PayPass* application remains in the **IDLE** state if an error occurs in the **IDLE** state. An error means a command response with status bytes different from '9000' (this includes reception of a command not supported in the current state).

Figure 11 graphically represents the state machine of the *PayPass* application.

Figure 11—State Machine of the PayPass Application



3.2 Variables

3.2.1 Internal Flags in Non Volatile Memory

The *PayPass* application manages the following internal flags in non-volatile memory.

Table 29—Internal Flags – Non Volatile Memory

Flag Name	Meaning
<i>Personalization Flag</i>	The flag is set (1b) if the application is personalized.

This chapter describes the behavior of the *PayPass* application when the *Personalization Flag* is set.

3.2.2 ATC

The *ATC* is a 2-byte binary counter maintained by the *PayPass* application in non-volatile memory. The *ATC* is initialized with '0000' during the pre-personalization of the *PayPass* application. The *ATC* is incremented by 1 each time the GET PROCESSING OPTIONS command is executed successfully.

3.3 C-APDU Processing

This section specifies:

- How the card application identifies the C-APDU transmitted by the transport layer to the application layer.
- How the C-APDU is accepted or rejected in function of the application state.
- How the card processes the C-APDU after it passed the pre-processing functions.

3.3.1 C-APDU Recognition

C-APDU recognition is specified as a procedure that identifies the C-APDU transmitted by the *PayPass* reader to the *PayPass* application. The recognition is based on the CLA and INS byte. Table 30 specifies the CLA and INS coding.

Table 30—C-APDU Recognition of the PayPass Application

CLA	INS	C-APDU
'00'	'A4'	SELECT
'80'	'A8'	GET PROCESSING OPTIONS
'00'	'B2'	READ RECORD
'80'	'2A'	COMPUTE CRYPTOGRAPHIC CHECKSUM

The C-APDU recognition procedure takes as input the CLA and INS bytes and produces as output one of the literals specified in the C-APDU column of Table 30.

If the CLA byte of the C-APDU is not one of those listed in Table 30, then the C-APDU recognition procedure returns status bytes '6E00'. If the CLA byte of the C-APDU is one of those listed in Table 30, but the combination of the CLA and INS byte of the C-APDU is not one of those listed in Table 30, then the C-APDU recognition procedure returns status bytes '6D00'.

When the PayPass application has recognized the C-APDU, it performs a validity check on the length fields:

- Consistency between Lc and the length of data sent
- Le ('00' or not present)

When the validity check detects an error on the lengths, then the C-APDU recognition procedure returns status bytes '6700'.

When the C-APDU is rejected, then the PayPass application resets all transient data and returns to the **SELECTED** state whenever the error occurs in a state different from the **IDLE** state. The PayPass application remains in the **IDLE** state if the error occurs in the **IDLE** state.

3.3.2 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 is specified in Table 31. The C-APDU acceptance procedure takes as input the output literal of the C-APDU recognition procedure.

Table 31—Acceptance Matrix of the PayPass Application

	IDLE	SELECTED	INITIATED
SELECT	Accept	Accept	Accept
GET PROCESSING OPTIONS	Reject	Accept	Reject
READ RECORD	Reject	Accept	Accept
COMPUTE CRYPTOGRAPHIC CHECKSUM	Reject	Reject	Accept

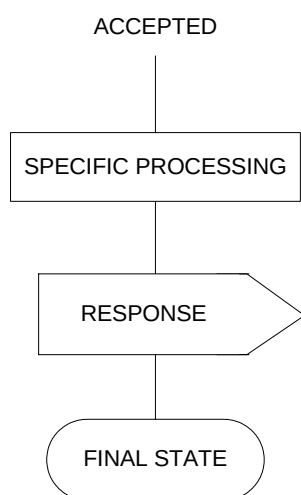
If the C-APDU is accepted in the current application state then the C-APDU is processed as specified in Section 3.3.3 of PART III.

When the C-APDU is rejected, then the *PayPass* application returns the status bytes ‘6985’, resets the transient data and returns to the **SELECTED** state whenever the error occurs in a state different from the **IDLE** state. The *PayPass* application remains in the **IDLE** state if the error occurs in the **IDLE** state.

3.3.3 C-APDU Processing

Figure 12 illustrates the actions taken by the *PayPass* application when a C-APDU is processed.

Figure 12—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 the section dedicated to the C-APDU.

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 used 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.

Errors in the processing of the C-APDU and not specified in the section dedicated to the C-APDU must respect the following requirements:

- The value of SW1-SW2 is left to the implementation.
- The destination state must be **IDLE** if the current state is **IDLE** and **SELECTED** in all other cases.
- The transient data must be reset.

3.4 SELECT

3.4.1 Command Message

The SELECT command message is coded according to Table 32.

Table 32—SELECT Command Message

Code	Value
CLA	'00'
INS	'A4'
P1	'04'
P2	'00'
Lc	'05' – '10'
Data	<i>AID</i>
Le	'00'

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

3.4.2 Response Message

The data field of the response message contains the *FCI* and is coded according to Table 59.

3.4.3 Processing

Figure 13 specifies the processing of the SELECT command.

Symbol 0

If $P1 \neq '04'$ or $P2 \neq '00'$, then the C-APDU is rejected (SW1-SW2 = '6A86').

Symbol 1

If Lc is different from the length of the *AID*, then the C-APDU is rejected (SW1-SW2 = '6700').

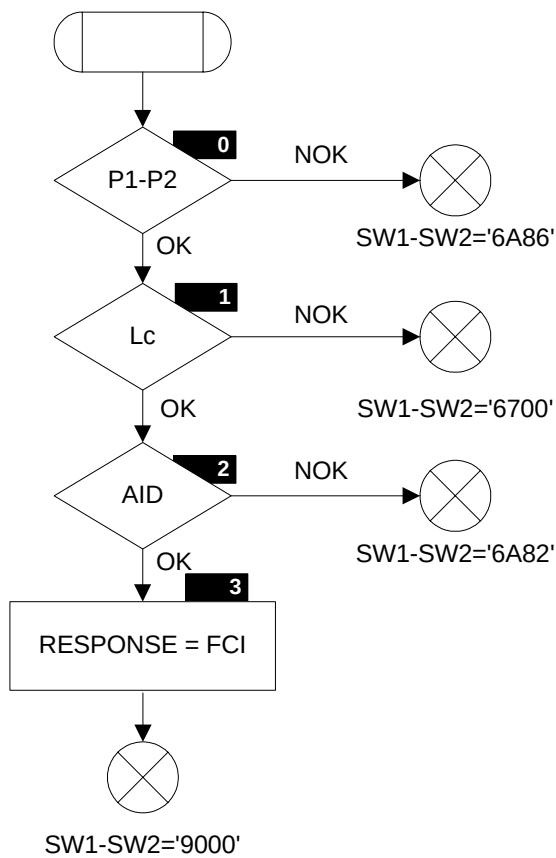
Symbol 2

If the *AID* in the command message data field is different from the *AID* of the *PayPass* application, then the C-APDU is rejected (SW1-SW2 = '6A82').

Symbol 3

The *PayPass* application builds the response message as specified in Section 3.4.2 of PART III.

Figure 13—SELECT Processing



3.4.4 Destination States

The destination states for the **SELECT** command are listed in Table 33.

Table 33—Destination States for SELECT Command

SW1	SW2	IDLE	SELECTED	INITIATED
'67'	'00'	IDLE	SELECTED	SELECTED
'6A'	'82'	IDLE	SELECTED	SELECTED
'6A'	'86'	IDLE	SELECTED	SELECTED
'90'	'00'	SELECTED	SELECTED	SELECTED
Other		IDLE	SELECTED	SELECTED

3.5 GET PROCESSING OPTIONS

3.5.1 Command Message

The GET PROCESSING OPTIONS command message is coded according to Table 34.

Table 34—GET PROCESSING OPTIONS Command Message

Code	Value
CLA	'80'
INS	'A8'
P1	'00'
P2	'00'
Lc	'02'
Data	'8300'
Le	'00'

The data field of the command message is a data object coded according to the *PDOL* provided by the *PayPass* application, introduced by the tag '83'. As the *PDOL* is not supported by the *PayPass* application, the length field of the template must be zero.

3.5.2 Response Message

The data field of the response message is a constructed data object with tag equal to '77'. The value field of the constructed data object includes the *AIP* (tag '82') and the *AFL* (tag '94'). The response message is specified in Table 35.

Table 35—GET PROCESSING OPTIONS Response Message Data Field

Data Object	Tag	Length
<i>Response Message Template</i>	'77'	'0A'
<i>AIP</i> ('0000')	'82'	'02'
<i>AFL</i> ('08010100')	'94'	'04'

3.5.3 Processing

Figure 14 specifies the flow of the GET PROCESSING OPTIONS command processing.

Symbol 0

If $P1 \neq '00'$ or $P2 \neq '00'$, then the C-APDU is rejected (SW1-SW2 = '6A86').

Symbol 1

If $Lc \neq 2$, then the C-APDU is rejected (SW1-SW2 = '6700').

Symbol 2

If the data field in the command message is different from '83 00', then the C-APDU is rejected (SW1-SW2 = '6985').

Symbol 3

If the *Application Transaction Counter (ATC)* has reached the limit (i.e. if $ATC = 'FFFF'$), then the C-APDU is rejected (SW1-SW2='6985').

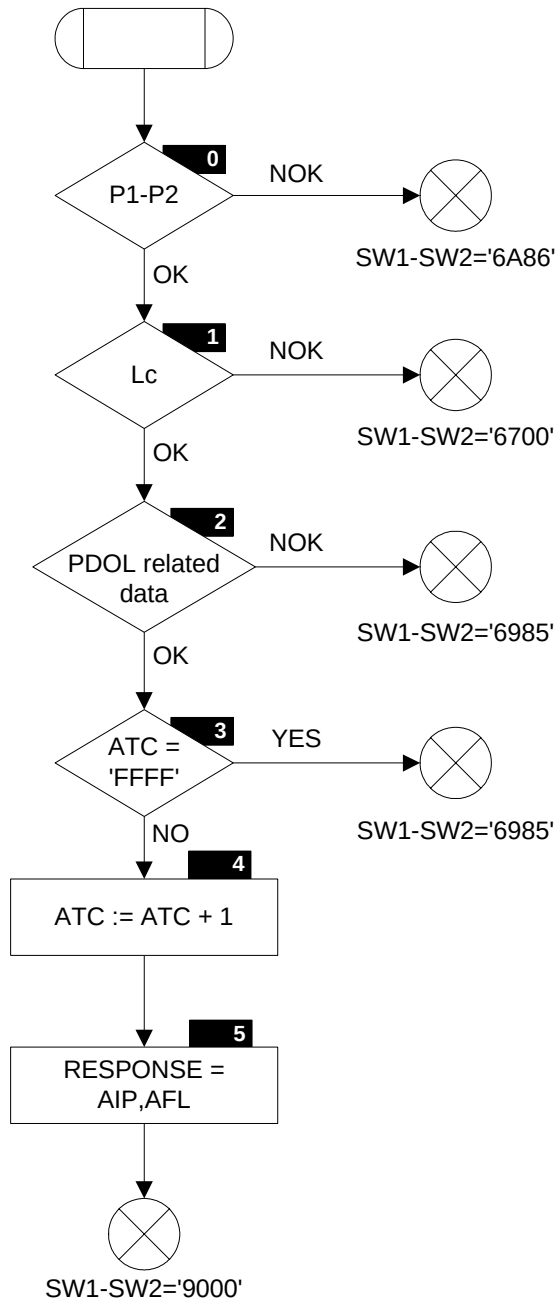
Symbol 4

The *PayPass* application increments the *ATC* and saves the new value in non-volatile memory.

Symbol 5

The *PayPass* application builds the response message as specified in Section 3.5.2 of PART III.

Figure 14—GET PROCESSING OPTIONS Processing



3.5.4 Destination States

The destination states for the GET PROCESSING OPTIONS command are listed in Table 36.

Table 36—Destination States for GET PROCESSING OPTIONS Command

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

3.6 READ RECORD

The *PayPass* application supports only one form of READ RECORD C-APDU: the EMV 4.1 READ RECORD.

3.6.1 Command Message

The READ RECORD command message is coded according to Table 37.

Table 37—READ RECORD Command Message

Code	Value
CLA	'00'
INS	'B2'
P1	Record Number
P2	Reference Control Parameter (see Table 38)
Lc	Not present
Data	Not present
Le	'00'

Table 38—Reference Control Parameter for the READ RECORD Command

b8-b4	b3-b1	Definition
xxxxxb		SFI
	100b	P1 is a record number

3.6.2 Response Message

The records referenced in the *AFL* are stored in files with *SFI* in the range 1-10. The records are BER-TLV constructed data objects with tag '70' as shown in Table 39.

Table 39—READ RECORD Response Message (SFI in Range 1-10)

'70'	Length	Record Template
------	--------	-----------------

Records stored in files with *SFI* not in the range 1 to 10 are outside the scope of these specifications and do not have to follow the coding as shown in Table 39.

3.6.3 Processing

Figure 15 specifies the flow of the READ RECORD command processing.

Symbol 0

The *PayPass* application checks that $P1 \neq '00'$ and $(P2 \ \& \ '07') = '04'$. If not, then the C-APDU is rejected (SW1-SW2 = '6A86').

Symbol 1

The *PayPass* application verifies that the *SFI* in the Reference Control Parameter corresponds to a supported record file. If the record file is not supported, then the C-APDU is rejected (SW1-SW2 = '6A82'). The way the *PayPass* application checks that the record file is supported is proprietary and left to the implementation.

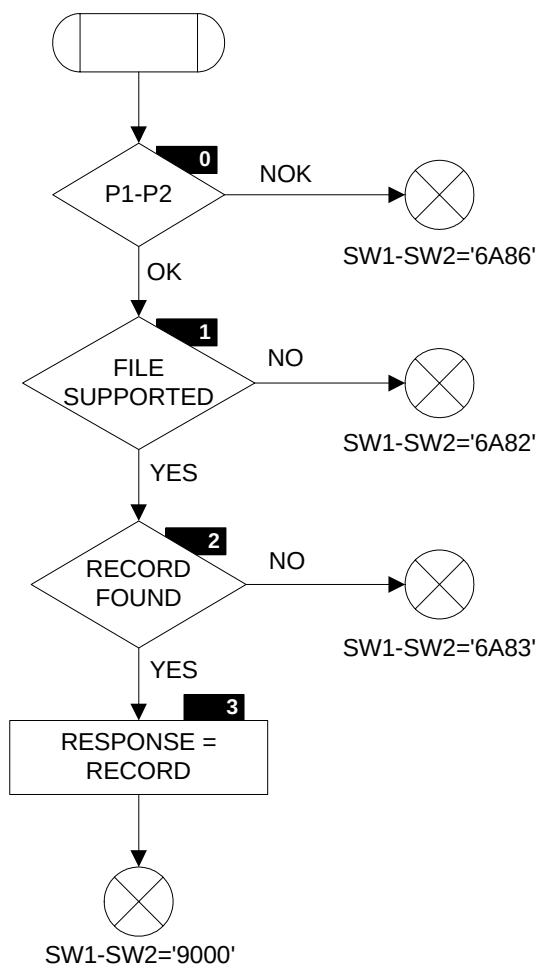
Symbol 2

The *PayPass* application verifies that there is a record corresponding to the Record Number, otherwise the C-APDU is rejected (SW1-SW2 = '6A83'). The way the *PayPass* application checks that the record is supported is proprietary and left to the implementation.

Symbol 3

The record encapsulated in the record template (tag '70') is sent in the response.

Figure 15—READ RECORD Processing



3.6.4 Destination States

The destination states for the READ RECORD command are listed in Table 40.

Table 40—Destination States for READ RECORD Command

SW1	SW2	SELECTED	INITIATED
'6A'	'82'	SELECTED	SELECTED
'6A'	'83'	SELECTED	SELECTED
'6A'	'86	SELECTED	SELECTED
'90'	'00'	SELECTED	INITIATED
Other		SELECTED	SELECTED

3.7 COMPUTE CRYPTOGRAPHIC CHECKSUM

3.7.1 Command Message

The COMPUTE CRYPTOGRAPHIC CHECKSUM command message is coded according to Table 41.

Table 41—COMPUTE CRYPTOGRAPHIC CHECKSUM Command Message

Code	Value
CLA	'80'
INS	'2A'
P1	'8E'
P2	'80'
Lc	'04'
Data	UN
Le	'00'

As the *UDOL* is not provided by the *PayPass* application, the data field of the command message is the value field of the *UN* data object.

3.7.2 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 42—COMPUTE CRYPTOGRAPHIC CHECKSUM Response Message

Data Object	Tag	Length
<i>Response Message Template</i>	'77'	15
<i>CVC3_{TRACK1}</i>	'9F60'	2
<i>CVC3_{TRACK2}</i>	'9F61'	2
<i>ATC</i>	'9F36'	2

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

3.7.3 Processing

Figure 16 specifies the flow of the COMPUTE CRYPTOGRAPHIC CHECKSUM command processing.

Symbol 0

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

Symbol 1

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

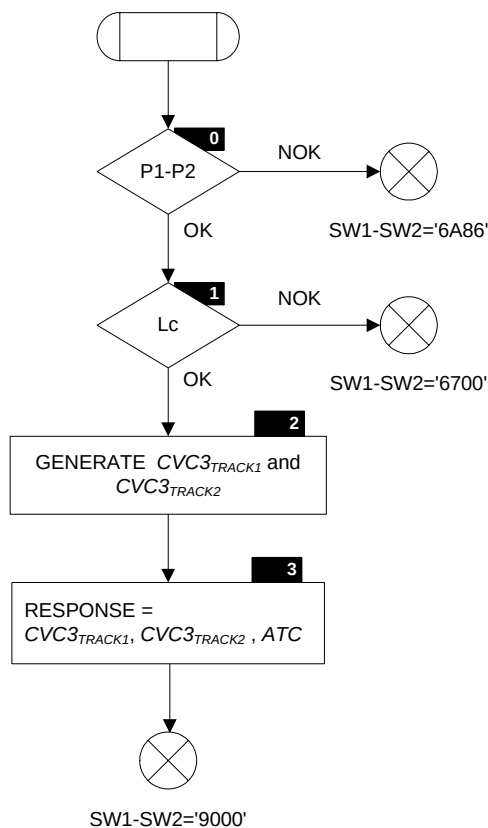
Symbol 2

The *PayPass* application generates $CVC3_{TRACK1}$ and $CVC3_{TRACK2}$ as specified in Section 3.8 of PART III.

Symbol 3

The *PayPass* application generates the response message template containing the $CVC3_{TRACK1}$, the $CVC3_{TRACK2}$ and the *ATC*.

Figure 16—COMPUTE CRYPTOGRAPHIC CHECKSUM Processing



3.7.4 Destination States

The destination states for the COMPUTE CRYPTOGRAPHIC CHECKSUM command are listed in Table 43.

Table 43—Destination States for COMPUTE CRYPTOGRAPHIC CHECKSUM Command

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

3.8 Dynamic CVC3

This section specifies how the *PayPass* application constructs the dynamic CVC3.

The *PayPass* 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 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 'F' padding) with the PAN sequence number⁷ (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:

$$Z_L := \text{DES3}(IMK_{CVC3})[Y]$$

$$Z_R := \text{DES3}(IMK_{CVC3})[Y \oplus ('FF' \| 'FF' \| 'FF' \| 'FF' \| 'FF' \| 'FF' \| 'FF' \| 'FF')]$$

and define :

$$KD_{CVC3} := (Z_L \| Z_R).$$

⁷ The PAN sequence number is sometimes also called the card sequence number. If the PAN sequence number is supported, then it is stored in the discretionary data of the *Track 1 Data* and *Track 2 Data* at a location proprietary to the issuer.

3.8.2 Dynamic CVC3 Generation

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

1. Concatenate the data listed in Table 44 in the order specified to obtain an 8 byte data block (D):

Table 44—Track 1 CVC3 Data Objects

Data Object	Length
$IVCVC3_{TRACK1}$	2 bytes
UN	4 bytes
ATC	2 bytes

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 objects that are used as input for the generation of the $CVC3_{TRACK1}$ and $CVC3_{TRACK2}$ cryptograms.

$IVCVC3_{TRACK1}$ is a MAC calculated over the *Track 1 Data*⁸ using KD_{CVC3} . $IVCVC3_{TRACK2}$ is a MAC calculated over the *Track 2 Data*⁸ also using KD_{CVC3} .

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

⁸ The *Track 1 Data* and *Track 2 Data* as stored in record 1 of SFI 1 during personalization (refer to Section 4.4.1 of PART III).

3.9 Data Objects Dictionary

This section provides a list of data object definitions, which are additional to the data objects listed in Section 5.3 of PART II. These definitions are for data objects that are internal to the *PayPass* application and which are not exchanged with the *PayPass* reader.

3.9.1 $IVCVC3_{TRACK1}$

Tag: --

Format: b, 2 bytes

Description: The $IVCVC3_{TRACK1}$ is an issuer proprietary static data object 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 $IVCVC3_{TRACK1}$.

3.9.2 $IVCVC3_{TRACK2}$

Tag: --

Format: b, 2 bytes

Description: The $IVCVC3_{TRACK2}$ is an issuer proprietary static data object 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 $IVCVC3_{TRACK2}$.

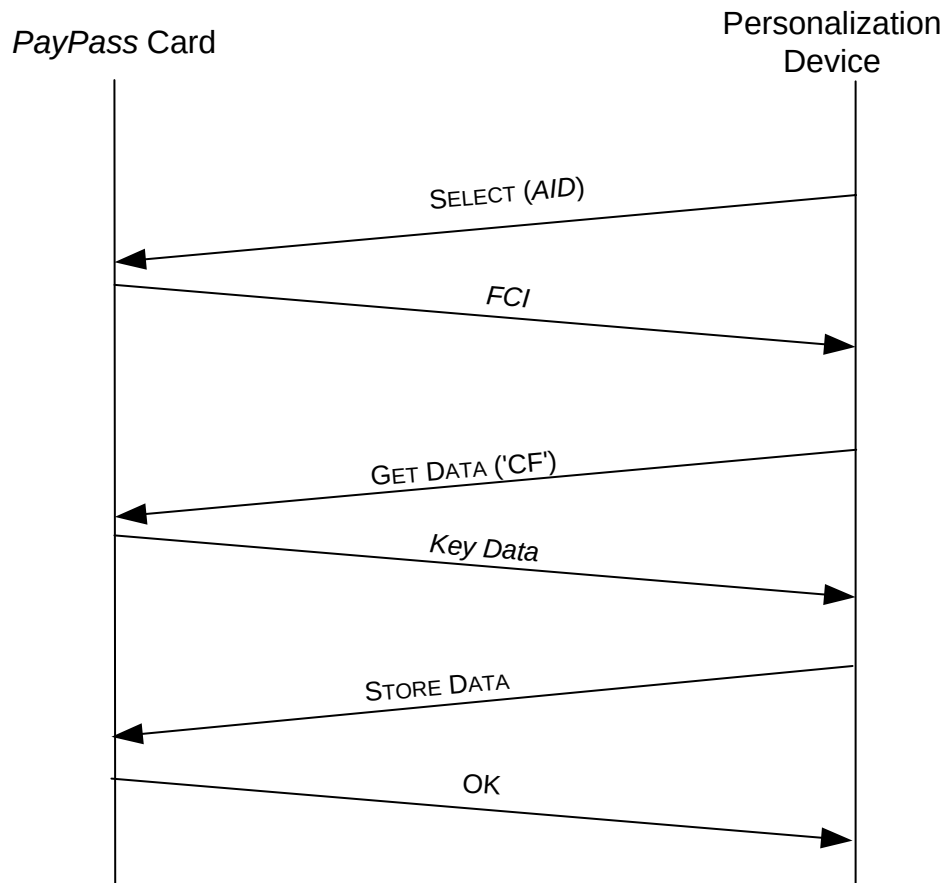
4 PayPass Application Personalization

This chapter specifies the personalization process of the PayPass application.

4.1 Personalization Processing Flow

The processing flow of messages between the personalization device and the *PayPass* application is shown in Figure 17.

Figure 17—Personalization Command Flow



The following commands are used:

- The **SELECT** command is used, with the *AID* stored in the card during the pre-personalization phase.
- The **GET DATA** command provides to the personalization device the identifier and version number of *KMC* and the derivation data for *KD_{PERSO}*.
- The *PayPass* application is personalized using one **STORE DATA** command that also concludes the personalization process of the application.

4.2 Personalization State Machine

This section describes the state machine of the *PayPass* application when it is not yet personalized (i.e. *Personalization Flag* = 0b).

4.2.1 Personalization States

When the *PayPass* application is not yet personalized, its behavior is specified by the personalization state machine. The personalization states used in this description are given in Table 45.

Table 45—Application States

State	Description
IDLE	Application is not currently selected
SELECTED NOT PERSONALIZED	Application is selected, but not personalized

The *PayPass* application is in the **IDLE** state if it is not currently activated. There is only one C-APDU which is handled in this state: the **SELECT** C-APDU which activates the application.

The *PayPass* application goes to the **SELECTED NOT PERSONALIZED** state:

- If it is just activated (i.e. the first **SELECT** C-APDU) and not yet personalized.
- When it has already been activated and a new **SELECT** command is received (i.e. the **SELECT** C-APDU).
- When a personalization session is aborted.

The personalization of the *PayPass* application starts in the **SELECTED NOT PERSONALIZED** state. There are two C-APDUs handled in this state:

- **GET DATA**
- **STORE DATA**

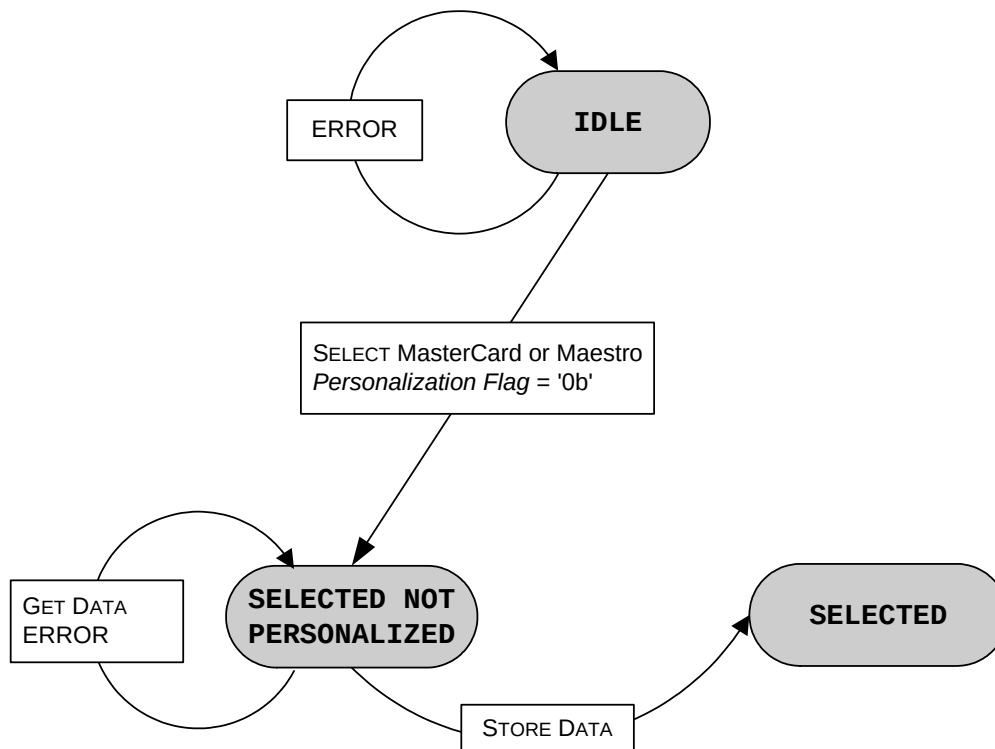
The GET DATA C-APDU does not modify the application state. The STORE DATA C-APDU personalizes the *PayPass* application. The application goes to the state **SELECTED** after the successful processing of the STORE DATA C-APDU.

The *PayPass* application returns to the **SELECTED NOT PERSONALIZED** state whenever an error occurs in a state different from the **IDLE** state. The *PayPass* application remains in the **IDLE** state if an error occurs in the **IDLE** state. An error means a command response with status bytes different from '9000'.

4.2.2 State Diagram

Figure 18 shows the personalization state diagram of the *PayPass* application.

Figure 18—*PayPass* Personalization State Diagram



4.3 Personalization Command Processing

This section specifies the command processing for the *PayPass* application during the personalization process.

4.3.1 C-APDU Recognition

C-APDU recognition is specified as a procedure that identifies the C-APDU transmitted by the *PayPass* reader to the *PayPass* application. The recognition is based on the CLA and INS byte. Table 46 specifies the CLA and INS coding for the *PayPass* application when the value of the *Personalization Flag* = 0b.

Table 46—C-APDU Recognition when *Personalization Flag* = 0b

CLA	INS	C-APDU
'00'	'A4'	SELECT
'80'	'CA'	GET DATA
'84'	'E2'	STORE DATA

The C-APDU recognition procedure takes as input the CLA and INS bytes and produces as output one of the literals specified in the C-APDU column of Table 46.

If the CLA byte of the C-APDU is not one of those listed in Table 46, then the C-APDU recognition procedure returns status bytes '6E00'. If the CLA byte of the C-APDU is one of those listed in Table 46, but the combination of the CLA and INS byte of the C-APDU is not one of those listed in Table 46, then the C-APDU recognition procedure returns status bytes '6D00'.

When the *PayPass* application has recognized the C-APDU, it performs a validity check on the length fields:

- Consistency between Lc and the length of data sent
- Le ('00' or not present)

When the validity check detects an error on the lengths, then the C-APDU recognition procedure returns status bytes '6700'.

When the C-APDU is rejected, then the *PayPass* application resets all transient data and returns to the **SELECTED NOT PERSONALIZED** state whenever the error occurs in a state different from the **IDLE** state. The *PayPass* application remains in the **IDLE** state if the error occurs in the **IDLE** state.

4.3.2 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 during the personalization process is specified in Table 47.

Table 47—Acceptance Matrix when Personalization Flag = 0b

	IDLE	SELECTED NOT PERSONALIZED
SELECT	Accept	Accept
GET DATA	Reject	Accept
STORE DATA	Reject	Accept

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.

When the C-APDU is rejected, then the *PayPass* application returns the status bytes ‘6985’, resets the transient data and remains in the current state (**IDLE** or **SELECTED NOT PERSONALIZED**).

4.3.3 SELECT Command

This section describes the SELECT command for a *PayPass* application that is not yet personalized (i.e. *Personalization Flag* = 0b).

Command Message

The SELECT command message is coded according to Table 32.

Response Message

The data field of the response message contains the *FCI* and is coded according to Table 59.

Processing

The internal processing of the SELECT command when the *Personalization Flag* = 0b, is identical to the internal processing of the SELECT command when the *Personalization Flag* = 1b as described in Section 3.4.3 of PART III.

Destination States

The destination states for the SELECT command when the value of the *Personalization Flag* = 0b, are listed in Table 48.

Table 48—Destination States for SELECT Command

SW1	SW2	IDLE	SELECTED NOT PERSONALIZED
'67'	'00'	IDLE	SELECTED NOT PERSONALIZED
'6A'	'82'	IDLE	SELECTED NOT PERSONALIZED
'6A'	'86'	IDLE	SELECTED NOT PERSONALIZED
'90'	'00'	SELECTED NOT PERSONALIZED	SELECTED NOT PERSONALIZED
Other		IDLE	SELECTED NOT PERSONALIZED

4.3.4 GET DATA Command

The *PayPass* application uses the GET DATA command during the personalization process to retrieve the *Key Data* (tag 'CF').

Command Message

The command message of the GET DATA command is coded as shown in Table 49.

Table 49—GET DATA Command Message

Code	Value
CLA	'80'
INS	'CA'
P1	'00'
P2	'CF'
Lc	Not present
Data	Not present
Le	'00'

Response Message

The data field of the response message contains the *Key Data* data object, TLV encoded.

Table 50—GET DATA Response Message Data Field

Tag	Length	Value
'CF'	'0B'	Value of <i>Key Data</i>

Processing

Figure 19 shows the processing flow for the GET DATA command.

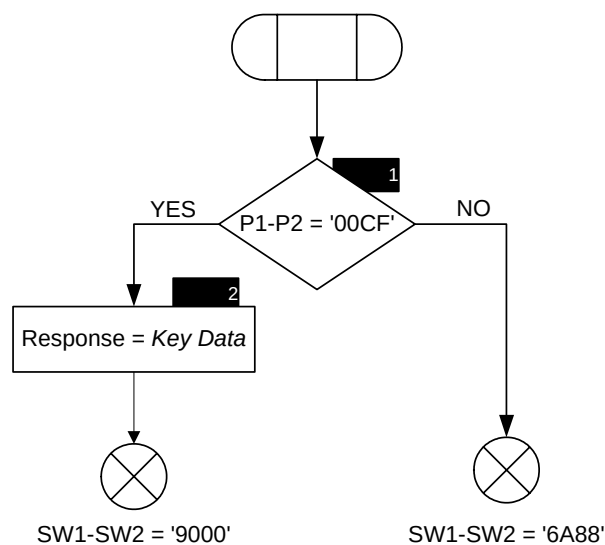
Symbol 1

If P1-P2 does not carry the tag of the *Key Data* (i.e. if P1-P2 ≠ '00CF'), then the application returns status bytes '6A88' (Referenced data not found).

Symbol 2

The *PayPass* application builds the response message as specified in Table 50.

Figure 19—GET DATA Processing



Destination States

The destination states for the GET DATA command when the value of the *Personalization Flag* = 0b, are listed in Table 51.

Table 51—Destination States for GET DATA Command

SW1	SW2	SELECTED NOT PERSONALIZED
'6A'	'88'	SELECTED NOT PERSONALIZED
'90'	'00'	SELECTED NOT PERSONALIZED
Other		SELECTED NOT PERSONALIZED

4.3.5 STORE DATA Command

The STORE DATA command is used to send the personalization data to the *PayPass* application.

Command Message

The command message of the STORE DATA command is coded as shown in Table 52.

Table 52—STORE DATA Command Message

Code	Value
CLA	‘84’
INS	‘E2’
P1	‘A0’
P2	‘00’
Lc	Length of command data
Data	DGI ₁ Length ₁ Data ₁ DGI ₂ Length ₂ Data ₂ DGI ₃ Length ₃ Data ₃ MAC
Le	Not present

The data field of the STORE DATA contains three DGIs. The coding of the data field of the STORE DATA command message is detailed in Table 53. The order in which the DGIs are included in the command data field is fixed and must be as specified in Table 53.

Table 53—STORE DATA Command Message Data Field

Field	Length	Value
DGI ₁	2	‘0101’
Length ₁	1	var
Value ₁	var	Content of record 1 (see Section 4.4.1 of PART III)
DGI ₂	2	‘A001’
Length ₂	1	‘0B’
Value ₂	11	Data objects for CVC3 generation (see Section 4.4.2 of PART III)
DGI ₃	2	‘A002’
Length ₃	1	‘10’
Value ₃	16	<i>KD_{CVC3}</i> (encrypted)
MAC	8	8 byte MAC that must be verified by the <i>PayPass</i> application prior to accepting the command (see Section 4.5.2 of PART III).



Note The length field of a DGI (Length_{*i*}) is coded on one byte with the binary format. Therefore it does not follow the BER-TLV encoding.

Response Message

The data field of the response message contains only SW1-SW2.

Processing

Figure 20 shows the processing flow for the STORE DATA command.

Symbol 0

The *PayPass* application checks if P1 = 'A0' and P2 = '00'. If this is not the case, then the C-APDU is rejected ('6A86').

Symbol 1

The *PayPass* application checks if $Lc - 44$ (i.e. Length₁, the length of the record 1 personalization data) $\leq$ reserved length for record. If this is not the case, then the C-APDU is rejected ('6700').

Symbol 2

The MAC is retrieved from the data field of the C-APDU (Lc-7 to Lc). The personalization data is retrieved from the data field of the C-APDU (1 to Lc-8).

Symbol 3

The *PayPass* application verifies the MAC. Refer to Section 4.5.2 of PART III. If the MAC verification is unsuccessful, the C-APDU is rejected ('6982').

Symbol 4

The *PayPass* application checks if the DGIs included in the personalization data are those listed in Table 53. If one or more DGIs in the personalization data are not known by the *PayPass* application or one or more DGIs are missing, the C-APDU is rejected ('6A80').

Symbol 5

The *PayPass* application stores the record data.

Symbol 6

The *PayPass* application stores the data for CVC3 generation.

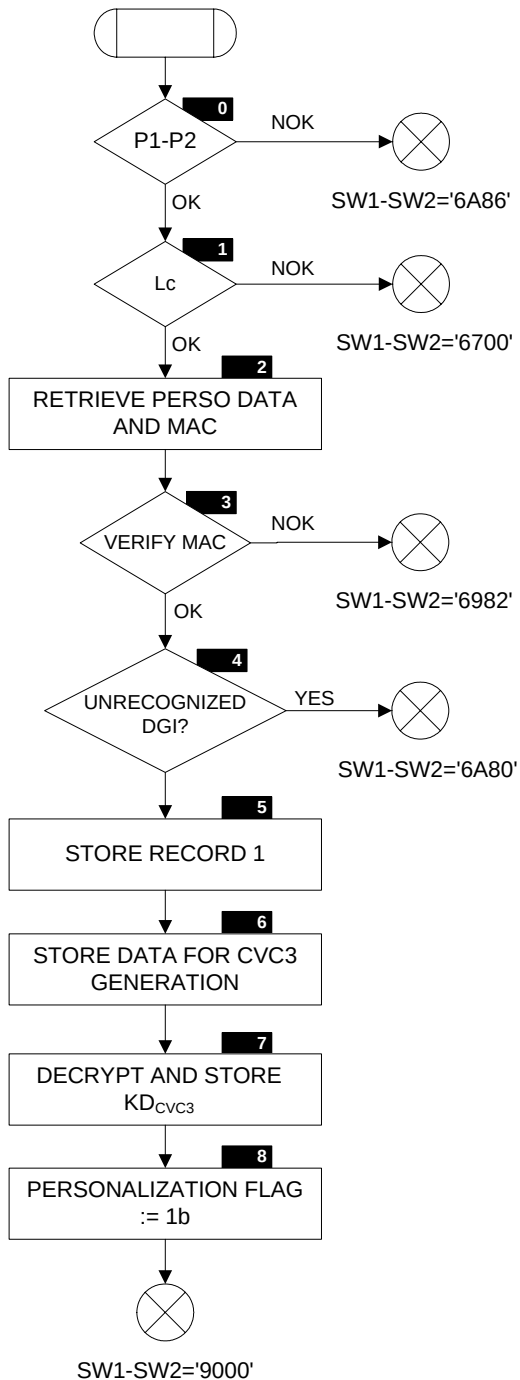
Symbol 7

The *PayPass* application decrypts and stores KD_{CVC3} .

Symbol 8

The *PayPass* application sets the *Personalization Flag* to 1b.

Figure 20—STORE DATA Processing



Destination State

The destination states for the STORE DATA command are listed in Table 54.

Table 54—Destination States for STORE DATA Command

SW1	SW2	SELECTED NOT PERSONALIZED
'67'	'00'	SELECTED NOT PERSONALIZED
'69'	'82'	SELECTED NOT PERSONALIZED
'6A'	'80'	SELECTED NOT PERSONALIZED
'6A'	'86'	SELECTED NOT PERSONALIZED
'90'	'00'	SELECTED
Other		SELECTED NOT PERSONALIZED

4.4 Personalization Data

This section specifies the data objects that are available to the issuer for personalization.

All data objects available for personalization are stored in persistent memory of the card.

4.4.1 DGI ‘0101’ – Data Objects Referenced in the AFL

Record 1 of the record file with SFI 1 must be able to store the TLV representation of the data objects listed in Table 55. As this file is transparent to the *PayPass* application, the order and presence of the data objects is irrelevant for the *PayPass* application⁹.

Table 55—Data Objects Referenced in the AFL

Tag	Name	Length (bytes)
‘9F6C’	<i>Application Version Number (Card)</i>	2
‘9F62’	<i>PCVC3_{TRACK1}</i>	6
‘9F63’	<i>PUNATC_{TRACK1}</i>	6
‘56’	<i>Track 1 Data</i>	var. up to 76
‘9F64’	<i>NATC_{TRACK1}</i>	1
‘9F65’	<i>PCVC3_{TRACK2}</i>	2
‘9F66’	<i>PUNATC_{TRACK2}</i>	2
‘9F6B’	<i>Track 2 Data</i>	var. up to 19
‘9F67’	<i>NATC_{TRACK2}</i>	1
‘9F6E’	<i>PayPass Third Party Data</i>	var. up to 32



Note The tag and length of the record template (tag ‘70’) are not included in the value field of DGI ‘0101’.

⁹ The presence of data objects is defined by the product rules and the interoperability requirements listed in Section 4.9 of PART II.

4.4.2 DGI ‘A001’ – Persistent Data Objects for CVC3 Generation

Table 56 lists the persistent card data objects used during the generation of $CVC3_{TRACK1}$ and $CVC3_{TRACK2}$. The data objects must be included in DGI ‘A001’ in the order as specified in Table 56.

Table 56—Data Objects for CVC3 Generation

Data Object	Length (bytes)
'00004000000000'	7
$IVCVC3_{TRACK1}$	2
$IVCVC3_{TRACK2}$	2

4.4.3 DGI ‘A002’ – Secret Key

DGI ‘A002’ includes the encrypted value of KD_{CVC3} as shown in Table 57. Encryption of KD_{CVC3} must be done as specified in Section 4.5.3 of PART III.

Table 57—Triple DES Key for CVC3 Generation

Name	Length (bytes)
KD_{CVC3} (encrypted)	16

4.5 Cryptography for Personalization

This section specifies the cryptography used in the key zone between the data personalization device and the *PayPass* application. This zone makes use of the *ICC Derived Key for Personalization* (KD_{PERSO}) derived from the *Master Key for Personalization* (KMC) shared by the *PayPass* application provider and personalizer.

4.5.1 KD_{PERSO}

A derived key (KD_{PERSO}) must be generated for each *PayPass* card and placed into the application during the pre-personalization phase. This key is used to generate and verify the MAC in the STORE DATA command. This key is also used to encrypt KD_{CVC3} when stored in DGI 'A002'. Furthermore, the application needs this key to decrypt KD_{CVC3} before storing it in the card.

KD_{PERSO} is a 16 byte (112 bits plus parity) DES key derived in the following way:

$$KD_{PERSO} := \text{DES3}(KMC)[\text{Six least significant bytes of the Key Data} \parallel \text{'F0'} \parallel \text{'01'}] \parallel \\ \text{DES3}(KMC)[\text{Six least significant bytes of the Key Data} \parallel \text{'0F'} \parallel \text{'01'}].$$

4.5.2 MAC for STORE DATA

The MAC must be calculated as specified in Annex B with KD_{PERSO} as key. The message M is constructed by concatenating the command header (CLA INS P1 P2 Lc) with the command data (excluding the MAC itself). The value of Lc in the data to compute the MAC must reflect the presence of the MAC, i.e. Lc = length of personalization data + 8.

4.5.3 Encryption/Decryption

This section describes the encryption and decryption of KD_{CVC3} during the personalization.

KD_{CVC3} must be encrypted and decrypted with DES3 in ECB mode as defined in [ISO/IEC 10116] using KD_{PERSO} as key.

5 PayPass Application Pre-Personalization

The *PayPass* application personalization is preceded by a pre-personalization process. This section describes the data that is assumed to be present in the *PayPass* application prior to personalization.

5.1 Application Selection Data Objects

Table 58 specifies the data objects that are used during the application selection process and that are assumed to be present prior to personalization.

Table 58—Data Objects for Application Selection

Name	Length (bytes)	Value
<i>AID</i>	5–16	Any binary value
<i>FCI</i>	var. up to 38	See Table 59

The *FCI* of the *PayPass* application is coded as specified in Table 59.

Table 59—FCI

Data Object	Tag	Length (bytes)
<i>FCI Template</i>	‘6F’	var. up to 38
<i>DF Name (= AID)</i>	‘84’	5–16
<i>FCI Proprietary Template</i>	‘A5’	var. up to 18
<i>Application Label</i>	‘50’	1–16



Note When ordering the *PayPass* card, the *AID* and *Application Label* must be specified.

5.2 GET PROCESSING OPTIONS Response

Table 60 lists the data objects used for the GET PROCESSING OPTIONS response. The *AFL* and *AIP* must be present prior to personalization.

Table 60—Data Objects for GET PROCESSING OPTIONS Response

Tag	Data Object	Length (bytes)	Format	Value
'94'	<i>AFL</i>	4	Binary	'08010100'
'82'	<i>AIP</i>	2	Binary	'0000'

5.3 Secret Key Information

Prior to the personalization process the *Key Data* (tag 'CF') data object must be placed onto the card. The *Key Data* contains the data used to derive KD_{PERSO} from *KMC*. *Key Data* contains the version number of *KMC* and the 6 byte master key identifier KMC_{ID} concatenated with the 4 byte Chip Serial Number as shown in Table 61.

Table 61—Contents of Key Data (tag 'CF')

Field	Length	Format
Version Number of <i>KMC</i> (VER_{KMC})	1	Binary
KMC_{ID}	6	Binary
Chip Serial Number (CSN)	4	Binary

KMC_{ID} is the data object required by the personalization device for identification of *KMC*.

The CSN uniquely identifies each chip from a specific chip/card manufacturer.

5.4 KD_{PERSO}

A derived key (KD_{PERSO}) must be generated for each card and placed into the *PayPass* application (refer to Section 4.5.1 of PART III for the generation of KD_{PERSO}). This key is used to check the MAC cryptogram in the data field of the STORE DATA command. This key is also used to decrypt in ECB mode KD_{CV3} when transferred in the data field of the STORE DATA command.

5.5 ATC

The *ATC* must be initialized to the value '0000'.

Annex A Example

This appendix contains an example that explains how a *PayPass – Mag Stripe* card is personalized, how the dynamic $CVC3_{TRACK1}$ and $CVC3_{TRACK2}$ are calculated and how the APDU commands and responses are formatted.

A.1 Pre-Personalization Data

This section lists the pre-personalization data stored in the card prior to the personalization.

FCI (PPSE):	6F 23 84 0E 32 50 41 59 2E 53 59 53 2E 44 44 46 30 31 A5 11 BF 0C 0E 61 0C 4F 07 A0 00 00 00 04 10 10 87 01 01
AID:	A0 00 00 00 04 10 10
FCI:	6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 0A 4D 61 73 74 65 72 43 61 72 64
AFL:	08 01 01 00
AIP:	00 00
KMC¹⁰	01 23 45 67 89 0A BC DE ED CB A0 98 76 54 32 10
VER_{KMC}:	01
KMC_{ID}:	54 13 12 FF FF FF
CSN:	89 AA 7F 00
KD_{PERSO}:	DES3(KMC)[FF FF 89 AA 7F 00 F0 01] DES3(KMC)[FF FF 89 AA 7F 00 0F 01]
KD_{PERSO}:	A8 6A 3D 06 CA E7 04 6A 10 63 58 D5 B8 23 9C BE
ATC:	00 00

¹⁰ *KMC* is not stored in the card. Its value is only listed to illustrate the derivation of KD_{PERSO} .

A.2 Personalization Data

This section lists the values of the data objects for personalization.

```

AVN (Card)      00 01

Track 1 Data:    42 35 34 31 33 31 32 33 34 35 36 37 38 34 38 30
                  30 5E 53 55 50 50 4C 49 45 44 2F 4E 4F 54 5E 30
                  39 30 36 31 30 31 33 33 30 30 30 33 33 33 30 30
                  30 32 32 32 32 32 30 30 30 31 31 31 31 30
                  (ASCII dump: B5413123456784800^SUPPLIED/NOT^0906101
                               330003330002222200011110)

PCVC3TRACK1:    00 00 00 38 00 00
PUNATCTRACK1: 00 00 00 00 E0 E0
NATCTRACK1:      03

Track 2 Data:    54 13 12 34 56 78 48 00 D0 90 61 01 90 00 99 00
                  00 00 0F

PCVC3TRACK2:    00 0E
PUNATCTRACK2: 0E 70
NATCTRACK2:      03

IMKCVC311:      01 23 45 67 89 98 76 54 32 10 01 23 45 67 89 98
Y:               13 12 34 56 78 48 00 00
=> ZL:          DES3(IMKCVC3)[Y]
=> ZR:          DES3(IMKCVC3)[Y ⊕ (FF FF FF FF FF FF FF FF)]
=> ZL:          DES3(IMKCVC3)[13 12 34 56 78 48 00 00]
=> ZR:          DES3(IMKCVC3)[EC ED CB A9 87 B7 FF FF]
=> KDCVC3:      A3 D0 BC 42 0A 76 66 11 F7 BC 19 E7 F2 E2 21 AF

IVCVC3TRACK1: Two least sign. bytes of MAC(KDCVC3)[Track 1 Data]
=>              MAC(KDCVC3)[42 35 34 31 33 31 32 33 34 35 36 37 38
                               34 38 30 30 5E 53 55 50 50 4C 49 45 44
                               2F 4E 4F 54 5E 30 39 30 36 31 30 31 33
                               33 30 30 30 33 33 33 30 30 30 32 32 32
                               32 32 30 30 30 31 31 31 31 30]

=>              8A 75 A3 EB 10 0B 77 80
IVCVC3TRACK1: 77 80

IVCVC3TRACK2: Two least sign. bytes of MAC(KDCVC3)[Track 2 Data]
=>              MAC(KDCVC3)[54 13 12 34 56 78 48 00 D0 90 61 01 90
                               00 99 00 00 00 0F]

=>              2D 2C A5 91 0A 50 99 D3
IVCVC3TRACK2: 99 D3

```

¹¹ IMK_{CVC3} is not stored in the card. Its value is only listed to illustrate the derivation of KD_{CVC3}.

A.3 Personalization Commands

- **SELECT Command**

```
>CARD: 00 A4 04 00 07 A0 00 00 00 04 10 10 00
CARD>: 6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 0A 4D
      61 73 74 65 72 43 61 72 64 90 00
```

- **GET DATA Command**

```
>CARD: 80 CA 00 CF 00
CARD>: CF 0B 01 54 13 12 FF FF FF 89 AA 7F 00 90 00
```

- **STORE DATA Command**

```
>CARD: 84 E2 A0 00 AB 01 01 7F 9F 6C 02 00 01 56 3E 42
      35 34 31 33 31 32 33 34 35 36 37 38 34 38 30 30
      5E 53 55 50 50 4C 49 45 44 2F 4E 4F 54 5E 30 39
      30 36 31 30 31 33 33 30 30 30 33 33 33 30 30 30
      32 32 32 32 32 30 30 30 31 31 31 31 30 9F 64 01
      03 9F 62 06 00 00 00 38 00 00 9F 63 06 00 00 00
      00 E0 E0 9F 65 02 00 0E 9F 66 02 0E 70 9F 6B 13
      54 13 12 34 56 78 48 00 D0 90 61 01 90 00 99 00
      00 00 0F 9F 67 01 03 A0 01 0B 00 00 40 00 00 00
      00 77 80 99 D3 A0 02 10 52 29 A2 B1 82 0F 32 13
      CA F2 24 3C B1 9C 5D F7 DE 65 E2 9F 48 C7 F2 12
CARD>: 90 00
```

A.4 Transaction Processing

```

nUN:      3
UN:        00 00 08 99
ATC:       00 5E
CVC3TRACK1: 2 least significant bytes of
              DES3(KDCVC3)[77 80 00 00 08 99 00 5E]
CVC3TRACK1: FB C7
CVC3TRACK2: 2 least significant bytes of
              DES3(KDCVC3)[99 D3 00 00 08 99 00 5E]
CVC3TRACK2: B8 92

• SELECT PPSE Command
  >CARD: 00 A4 04 00 0E 32 50 41 59 2E 53 59 53 2E 44 44
         46 30 31 00
  CARD>: 6F 23 84 0E 32 50 41 59 2E 53 59 53 2E 44 44 46
         30 31 A5 11 BF 0C 0E 61 0C 4F 07 A0 00 00 00 04
         10 10 87 01 01 90 00

• SELECT AID Command
  >CARD: 00 A4 04 00 07 A0 00 00 00 04 10 10 00
  CARD>: 6F 17 84 07 A0 00 00 00 04 10 10 A5 0C 50 0A 4D
         61 73 74 65 72 43 61 72 64 90 00

• GET PROCESSING OPTIONS Command
  >CARD: 80 A8 00 00 02 83 00 00
  CARD>: 77 0A 82 02 00 00 94 04 08 01 01 00 90 00

• READ RECORD Command
  >CARD: 00 B2 01 0C 00
  CARD>: 70 7F 9F 6C 02 00 01 56 3E 42 35 34 31 33 31 32
         33 34 35 36 37 38 34 38 30 30 5E 53 55 50 50 4C
         49 45 44 2F 4E 4F 54 5E 30 39 30 36 31 30 31 33
         33 30 30 30 33 33 33 30 30 30 32 32 32 32 32 30
         30 30 31 31 31 31 30 9F 64 01 03 9F 62 06 00 00
         00 38 00 00 9F 63 06 00 00 00 00 E0 E0 9F 65 02
         00 0E 9F 66 02 0E 70 9F 6B 13 54 13 12 34 56 78
         48 00 D0 90 61 01 90 00 99 00 00 00 0F 9F 67 01
         03 90 00

• COMPUTE CRYPTOGRAPHIC CHECKSUM Command
  >CARD: 80 2A 8E 80 04 00 00 08 99 00
  CARD>: 77 0F 9F 61 02 B8 92 9F 60 02 FB C7 9F 36 02 00
         5E 90 00

```

The *PayPass* reader generates the track 2 data to be included in the authorization message as follows:

$CVC3_{TRACK2}$ in base 10: 47250

ATC in base 10: 94

=> track 2 data: 54 13 12 34 56 78 48 00 D0 90 61 01
 9 **094** 99 **899 250** **3** F
 ↑↑↑ ↑↑↑ ↑↑↑ ↑
 ATC UN $CVC3_{TRACK2}$ n_{UN}

The *PayPass* reader generates the track 1 data to be included in the authorization message as follows:

$CVC3_{TRACK1}$ in base 10: 64455

ATC in base 10: 94

=> track 1 data: 42 35 34 31 33 31 32 33 34 35 36 37 38 34 38 30
 30 5E 53 55 50 50 4C 49 45 44 2F 4E 4F 54 5E 30
 39 30 36 31 30 31 33 33 34 35 35 33 33 33 30 39
 34 32 32 32 32 32 38 39 39 31 31 31 31 33

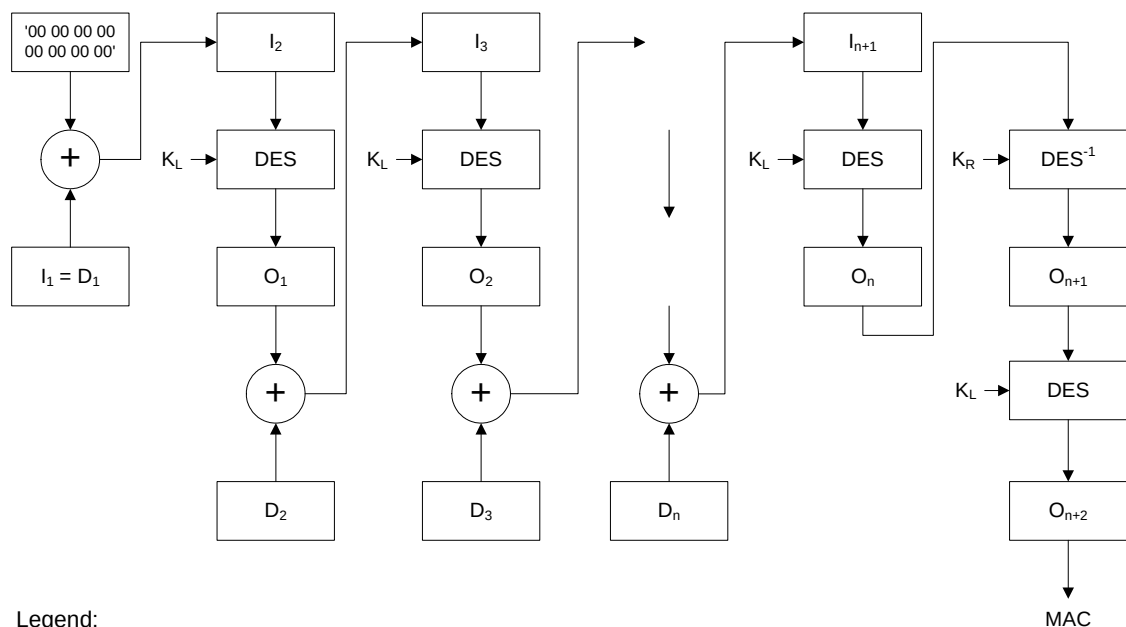
ASCII dump: B5413123456784800^SUPPLIED/NOT^0906101
 33**455333094**22222**899**1111**3**
 ↑↑↑ ↑↑↑ ↑↑↑ ↑
 $CVC3_{TRACK1}$ ATC UN n_{UN}

Annex B 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: '8000000000000000'. 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 '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 21. Figure 21 assumes that after the padding there are n data blocks (D).

Figure 21—MAC Algorithm



Legend:

$K = K_L || K_R$

DES indicates single DES encryption

DES^{-1} indicates single DES decryption

Annex C Multi-Application Manager

C.1 Introduction

The card specification included in PART III of the *PayPass – Mag Stripe Technical Specifications*, views support for the PPSE as another application on the card. Therefore, the *PayPass – Mag Stripe* card contains two applications: the PPSE application and the *PayPass – Mag Stripe* application.

The card specification specifies how the SELECT command is handled by each of the applications individually, but does not specify how the card itself processes the SELECT command. This annex introduces the PPSE and *PayPass – Mag Stripe* application in a multi-application environment. It is assumed that the multi-application aspects are managed by an entity referred to as the multi-application manager.

**Note**

These principles are used in order to present the multi-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.

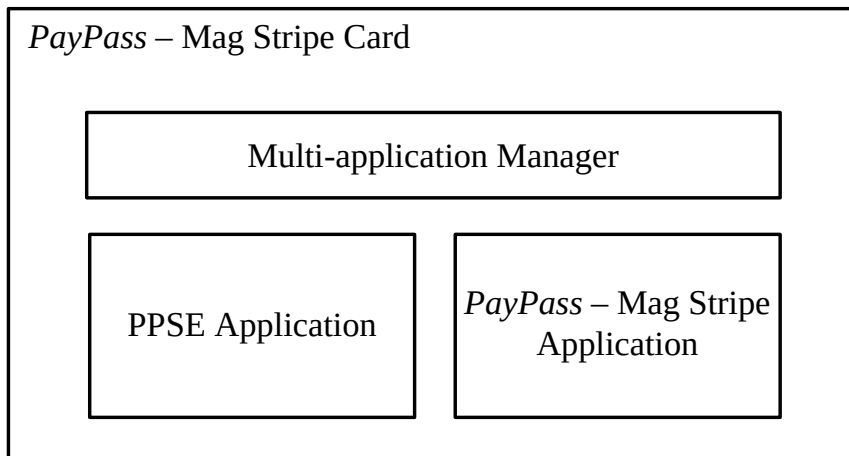
The multi-application manager described in this document is specific for the *PayPass – Mag Stripe* card containing the PPSE and *PayPass – Mag Stripe* application. It is not the description of a multi-application manager of a generic multi-application card.

C.2 The Multi-Application Manager

The card specification assumes that the dispatching of C-APDU commands between the *PayPass – Mag Stripe* application and the PPSE application is managed by the multi-application manager. Figure 22 illustrates the *PayPass – Mag Stripe* card containing the multi-application manager, the PPSE application and the *PayPass – Mag Stripe* application.

For the multi-application manager, an application may be either active or inactive. The active application is the application that is currently selected. An application that is inactive is in the **IDLE** state.

Figure 22—PayPass – Mag Stripe Card



All C-APDU's send to the card are first handled by the multi-application manager before they are eventually forwarded to the active application. The multi-application manager treats the SELECT C-APDU different from any other C-APDU. Refer to Section C.4 for the processing of the SELECT C-APDU. A C-APDU different from the SELECT C-APDU is always forwarded to the active application. If no application is active, and a C-APDU different from the SELECT C-APDU is received, then the multi-application manager returns status bytes '6895' (Conditions of use not satisfied).

Only one of the two applications can be active at the same time. The multi-application manager deactivates the current active application if a successful SELECT command for the other application is received.

C.3 Processing of a C-APDU

Figure 23 specifies the processing of the multi-application manager when a C-APDU is received.

Symbol 0

The multi-application manager verifies if the C-APDU is a SELECT C-APDU (CLA='00' and INS='A4').

Symbol 1

The multi-application manager processes the SELECT C-APDU as specified in Section C.4.

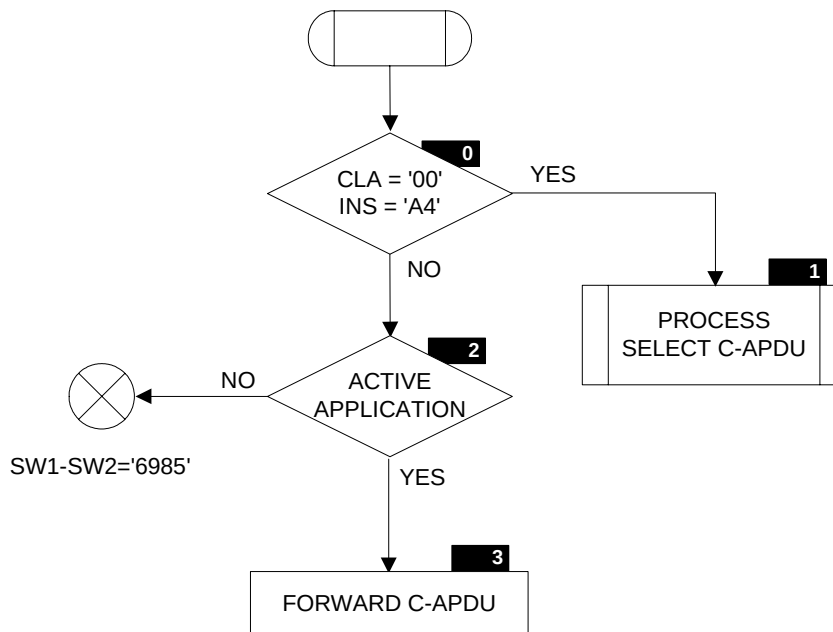
Symbol 2

The multi-application manager verifies if one of the applications is active. If this is not the case, then the multi-application manager returns status bytes '6985' (Conditions of use not satisfied).

Symbol 3

The multi-application manager forwards the C-APDU to the active application.

Figure 23—C-APDU Processing



C.4 Processing of the SELECT Command

Figure 24 specifies the processing of the multi-application manager when a SELECT C-APDU is received.

Symbol 0

If $P1 \neq '04'$ or $P2 \neq '00'$, then the C-APDU is rejected (SW1-SW2 = '6A86').

Symbol 1

If $Lc < 5$ or $Lc > 16$, then the C-APDU is rejected (SW1-SW2 = '6700').

Symbol 2

The multi-application manager verifies if the *AID* in the command message data field is equal to "2PAY.SYS.DDF01".

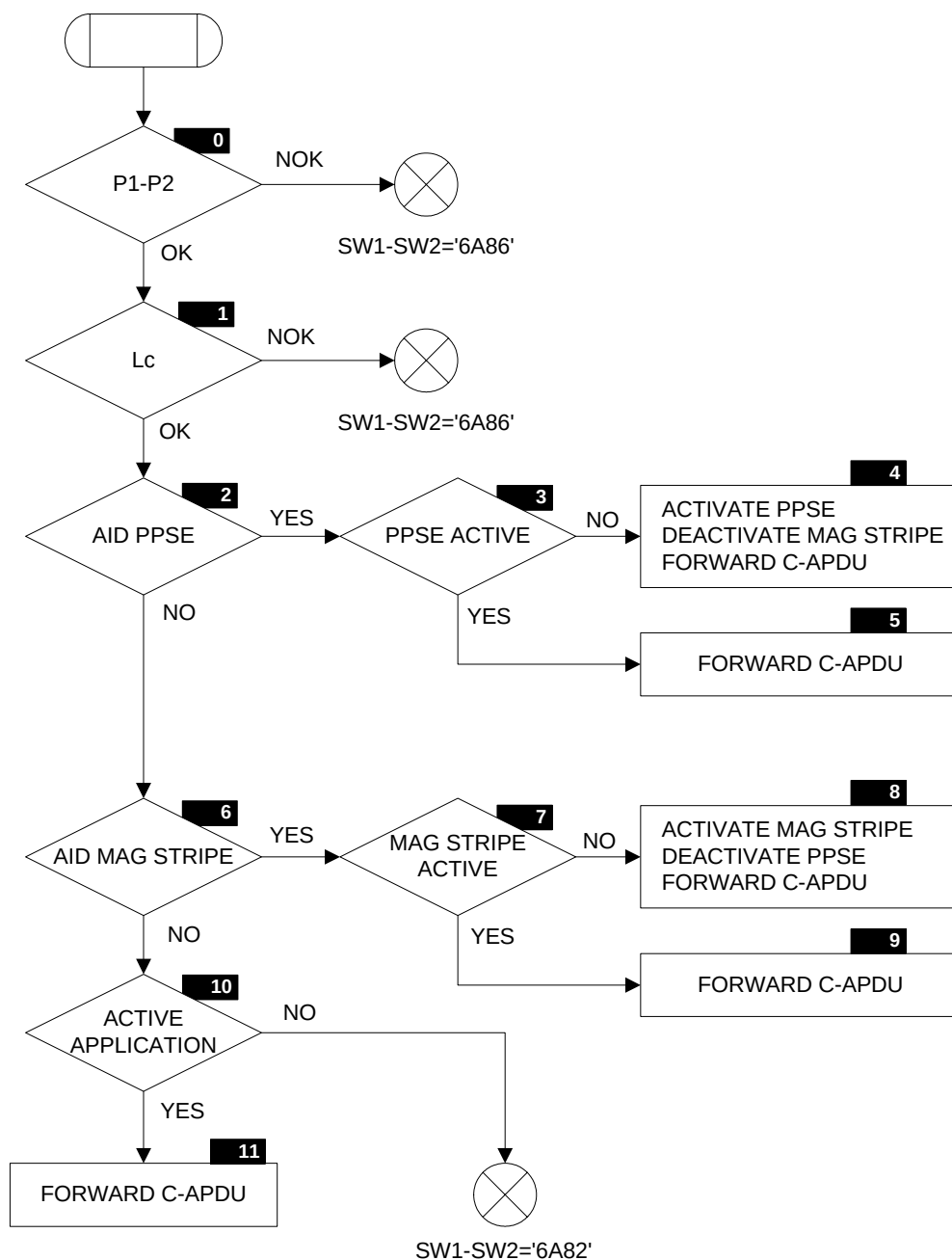
Symbol 3

The multi-application manager verifies if the PPSE application is active.

Symbol 4

The multi-application manager activates the PPSE application, deactivates the *PayPass – Mag Stripe* application and forwards the SELECT C-APDU to the PPSE application. (Note that it is the PPSE application that stores the FCI and status bytes in the card output buffer.)

Figure 24—Processing of SELECT C-APDU

**Symbol 5**

The multi-application manager forwards the SELECT C-APDU to the PPSE application.

Symbol 6

The multi-application manager verifies if the *AID* in the command message data field is equal to the *AID* of the *PayPass – Mag Stripe* application.

Symbol 7

The multi-application manager verifies if the *PayPass* – Mag Stripe application is active.

Symbol 8

The multi-application manager activates the *PayPass* – Mag Stripe application, deactivates the PPSE application and forwards the SELECT C-APDU to the *PayPass* – Mag Stripe application. (Note: It is the *PayPass* – Mag Stripe application that stores the FCI and status bytes in the card output buffer.)

Symbol 9

The multi-application manager forwards the SELECT C-APDU to the *PayPass* – Mag Stripe application.

Symbol 10

The multi-application manager verifies if there is an active application. If this is not the case, then the multi-application manager returns status bytes '6A82' (File not found).

Symbol 11

The multi-application manager forwards the SELECT C-APDU to the active application.

***** END OF DOCUMENT *****