Terminal Architecture for PSAM Applications (TAPA)

Application Architecture Specification

Version 3.0

October 2013

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Revision Log

Version	Date	Affects	Brief Description of Change
2.0	4/00	All	Initial Publication
2.1	1/01	All	Editorial revisions
		Tables 16, 46, 52	Change description of SP_{MAC} field.
		6.1.1, 6.1.2	Change example data for magnetic stripe read and write
		6.1.2	Restrict writing to the magnetic strip to Track 3 only.
		Table 66	Change label and description of LPKM field.
		Tables 72, 73, 74, 81	Fix lengths of fields
		Table 85	Fix Destination address, add SP_{MAC} field
		Table 98	Add Synchronize PSAM/PIN Pad command
		Table 105	Permit 0 (zero) AIDs.
		Table 123	Clarify use of Pointer Orientation field
		11.2.6.1	Clarify use of the Update Record command
		Table 130	Add LEN_{SKEY} field
		Table 149	Allow event type '01' to be originated by the PSAM Handler
		Figure 11	Add flow arrow for error case
		Table 166	Set minimum key lengths to 1024 bits. Fix maxima for two keys. Add warning regarding use of shorter keys.
		Tables 58 and 59 7.2.1.2, 14.5.4.3, 14.6.2, 14.6.2.1, 14.6.3, 14.6.3.1	Allow for multiple PIN Pad key versions
		Tables 60 and 167	Add response codes for PIN Pad processing.
		7.2.5.3	Require that derived session keys have odd parity

Version	Date	Affects	Brief Description of Change
		14.5.3.4	Require that the RSA and padding operations be performed within the protected devices
		14.6.10.1	Acquirers are responsible for ensuring that expired CA public keys are no longer used after they expire.
3.0	5/12	All	Editorial revisions and functional enhancements. Replace Tamper Evident Device with Secure Device. Remove reference to Common Electronic Purse (CEP) Updated figures and fonts
		2.5.2	Replaced ISO/IEC 14443 with EMV Contactless Specifications for Payment Systems – Entry Point Specification Updated references to current versions
		3.2	Compliance awareness with PCI Data Security Standards added
		4.2.4	Add encrypted response for handler
		4.3	Add encrypted commands
		4.4	Specify keys to use for encrypted data.
		6.1.1	Add enciphered magnetic stripe responses
		6.2.2	Add enciphered commands
		6.4	Reference to EMVCo contactless interface specification included
		7.2.1.2	Clarifying which public key version to select if more than one match
		7.2.1.3	Reference to table corrected
		Table 60	Response codes removed
		7.2.1.5	New requirement added
		Table 68	Response codes added
		7.2.5	$KSES_{CDP}$ added
		7.2.4.2	Reference in step 5 corrected
			New session key $KSES_{DATA}$ added
		8.1	A new Get Amount Command (get Amount 2) added to include EMV field Amount Other
		9	Note on discrepancies between ISO and EMV added
		Table 109	Response codes added
		14.1	Introduce PIN Pad less Secure Devices

Version	Date	Affects	Brief Description of Change
		14.3.1.5	SD requirements changed to allow for key entry of non-PIN data outside payment application
		14.6.2	Changed to reference the PCI PTS requirements, only
		14.6.3	Changed to reference the PCI PTS requirements; redundant requirements removed
		14.8	Add Secure Devices in terminals without any PIN Pad
		Table 176	New response code added: Transaction interrupt request
		17	Add acronyms CDP and SD
3.0	10/13	all	Limit copyright to Nets Denmark A/S
3.0		many	Change Secure Device to Secure Cryptographic Device and acronym SD to SCD

2. Document Overview

2.1 Purpose

The purpose of this document is to provide the information necessary for a POS device manufacturer or application developer to gain an understanding of the Terminal Architecture for PSAM Applications (TAPA) for purposes of creating multi-function applications.

2.2 Intended Audience

This document is intended for use by all technical staff involved in the development, testing, operation, and maintenance of one or more structural components of the Terminal Architecture for PSAM Applications.

2.3 Included in this Document

Included in this document are:

- Overview of the Terminal Architecture for PSAM Applications and a description of the individual structural components comprising that architecture.
- Description of the general functional requirements to be performed by each structural component.
- Description of the message formatting, addressing, exchanging of messages between structural components, and command sets used by these components.
- Description of the general message and error handling procedures to be conducted within the terminal application.
- Description of the terminal and PSAM initialization procedures.

2.4 Not Included in this Document

Not included in this document are:

- Specifications already available in other documents, such as the EMV specifications and ISO standards.
- Scheme specific information related to business requirements, design options, or implementation details – including supported command sets, transaction data flows, or user interface requirements.

- Specific message formats or online communication protocols used.

2.5 Reference Information

2.5.1 Requirement Numbering

Requirements in this specification are uniquely numbered with the number appearing next to each requirement.

A requirement can 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.

2.5.2 References

The following documents are referenced in this specification:

1. Terminal Architecture for PSAM Applications, Overview, version 2.0, April 2000.
2. ISO/IEC 7816-3: 2006, "Identification cards - Integrated circuit cards with contacts - Part 3: Electrical interface and transmission protocols".
3. ISO/IEC 7816-4: 2005, "Identification cards – Integrated circuit cards with contacts - Part 4: Organization, security and commands for interchange".
4. EMV Contactless Specifications for Payment Systems – V2.1, March 2011
5. ISO/IEC 9797-1:2011 "Information technology - Security techniques - Message Authentication Codes (MACs) - Part 1: Mechanisms using a block cipher"
6. EMV, version 4.3, November 2011 "Integrated Circuit Card Specification for Payment Systems" including later bulletins
7. ISO/IEC 646: 1991, "Information technology - ISO 7-bit coded character set for information interchange"
8. ISO 8859-15: 1999, "Information technology – 8-bit single-byte coded graphic character sets – Part 15: Latin alphabet No.9"
9. ISO/IEC 7813: 2006, "Information technology – Identification Cards - Financial Transaction Cards"

10. ISO/IEC 4909: 2006 “Identification cards – Financial transaction cards – Magnetic stripe data content for track 3”

11. PCI SSC PTS, version 3.1, June 2011, “PIN Transaction Security”

2.5.3 Command and Response Format Conventions

This specification adopts the following conventions for specifying the commands and responses exchanged between the terminal and PSAM applications. Commands and responses have been extended to include plaintext as well as enciphered commands.

PSAM Commands

For commands sent to the PSAM by the MAD-Handler, this specification documents the entire set of ICC Command Terminal Messages (Messages Type ‘42’, ‘46’and ‘47’), including the embedded Command APDU, which is being forwarded to the PSAM. To aid the PSAM developer, the C-APDU portion is shaded. The document will, where applicable, show the enciphered as well as the plaintext versions.

Commands requesting an enciphered response are identified by the most significant bit of the sub handler address being set, i.e. that 1000 0000b is added to the “normal” sub handler address.

For commands sent to the PSAM that are generated by the PSAM Handler, this specification documents the C-APDU, which is generated by the PSAM Handler.

PSAM Responses

This specification defines the format and contents of the Response Terminal Message (Message Type ‘FF’) received by the MAD-Handler in response to a command sent to the PSAM. The entire response message, excluding the Response Code (RC) is sent to the PSAM Handler by the PSAM within a response APDU. To aid the PSAM developer, the portion of the response generated by the PSAM is shaded.

A detailed description of the message handling performed by the PSAM Handler can be found in section 9.1. A description of the PSAM application requirements is in section 10.

2.5.4 Notational Conventions

Hexadecimal Notation

Values expressed in hexadecimal form are enclosed in single quotes (e.g., '_'). For example, 27509 decimal is expressed in hexadecimal digits as '6B75'.

Letters used to express constant hexadecimal values are always upper case ('A' - 'F'). Where lower case is used, the letters have a different meaning explained in the text.

Binary Notation

Values expressed in binary form are followed by a lower case "b". For example, '08' hexadecimal is expressed in binary as 00001000b.

Document Word Usage

The following words are used often in this document and have specific meanings:

- Must

Defines a product or system capability that is required, compelled, or mandatory.

- Should

Defines a product or system capability that is highly recommended.

- May

Defines a product or system capability that is optional.

Notation used in the PIN Pad Cryptography section

⊕ represents the bitwise XOR function

SHA-1 (X) := the SHA-1 hash of X

SHA (X,n) := leftmost n bytes of SHA-1(X)

2.6 Document Organization

The document is organized as follows:

- Section 1 is the revision log.
- Section 2 provides an overview of this document.
- Section 3 provides an architectural overview of the Terminal Architecture for PSAM Applications (TAPA).
- Section 4 provides information concerning the general

functional requirements of TAPA architectural components.

- Section 5 describes the processing requirements of the Multi-Application Driver Handler.
- Section 6 describes the processing requirements of the Card Handlers.
- Section 7 describes the processing requirements of the User Interface Handlers.
- Section 8 describes the processing requirements of the Merchant Application Handlers.
- Section 9 describes the processing requirements of the PSAM Handlers.
- Section 10 describes the requirements for TAPA-compliant PSAM applications
- Section 11 describes the processing requirements of the Data Store Handler.
- Section 12 describes the processing requirements of the Communication Handler.
- Section 13 describes the processing requirements for the Event Handler.
- Section 14 provides the requirements for the use of Secure Cryptographic Devices and PIN Processing when a PSAM is used in conjunction with a PIN Pad or with a Secure Cryptographic Device, only.
- Section 15 provides a listing of Response Codes that may be generated.
- Section 16 describes the referenced data elements.
- Section 17 provides a list of acronyms

3. Architectural Overview

3.1 Introduction

This section provides an overview of the Terminal Architecture for PSAM Applications (TAPA) and outlines the general functional and processing requirements that this architecture is intended to fulfil. This section will additionally identify and describe the various structural components comprising the overall TAPA architecture.

3.2 General Requirements

The TAPA application architecture defines a generic Terminal - PSAM interface such that a terminal application may function using PSAM's produced by different manufacturers.

The TAPA application architecture supports a multi-function point-of-sale terminal application that can support both EMV and other payment applications as well as additional proprietary applications such as loyalty programs.

TAPA is designed to be independent of the implementation strategy chosen and is independent of both terminal operating system and device architecture.

TAPA does not impose restrictions on the type of transmission protocol used to exchange messages between most structural components. However, communication with ICCs is assumed to conform to the command-response protocol defined in reference 3, ISO/IEC 7816-4.

TAPA does not restrict the physical configuration of the terminal. The terminal may exist as either a complete unit of fully integrated components (such as a conventional payment terminal) or may exist to varying degrees as a distributed system using shared components such as an Internet payment server. For a distributed system, the PCI Data Security Standard and the PCI PTS are to be complied with.

3.3 Terminal Application Architecture

The terminal application architecture consists of a set of logical components. These components include a Router, a set of Device Handlers, a Multi-Application Driver Handler, and one or more Purchase Secure Application Modules (PSAM). Each of these components is described in detail in subsequent sections of this document. Figure 1 provides an illustration of the various components, relative location, and relationships

within the TAPA architecture. (Note that the device handlers are grouped into categories according to their function.)

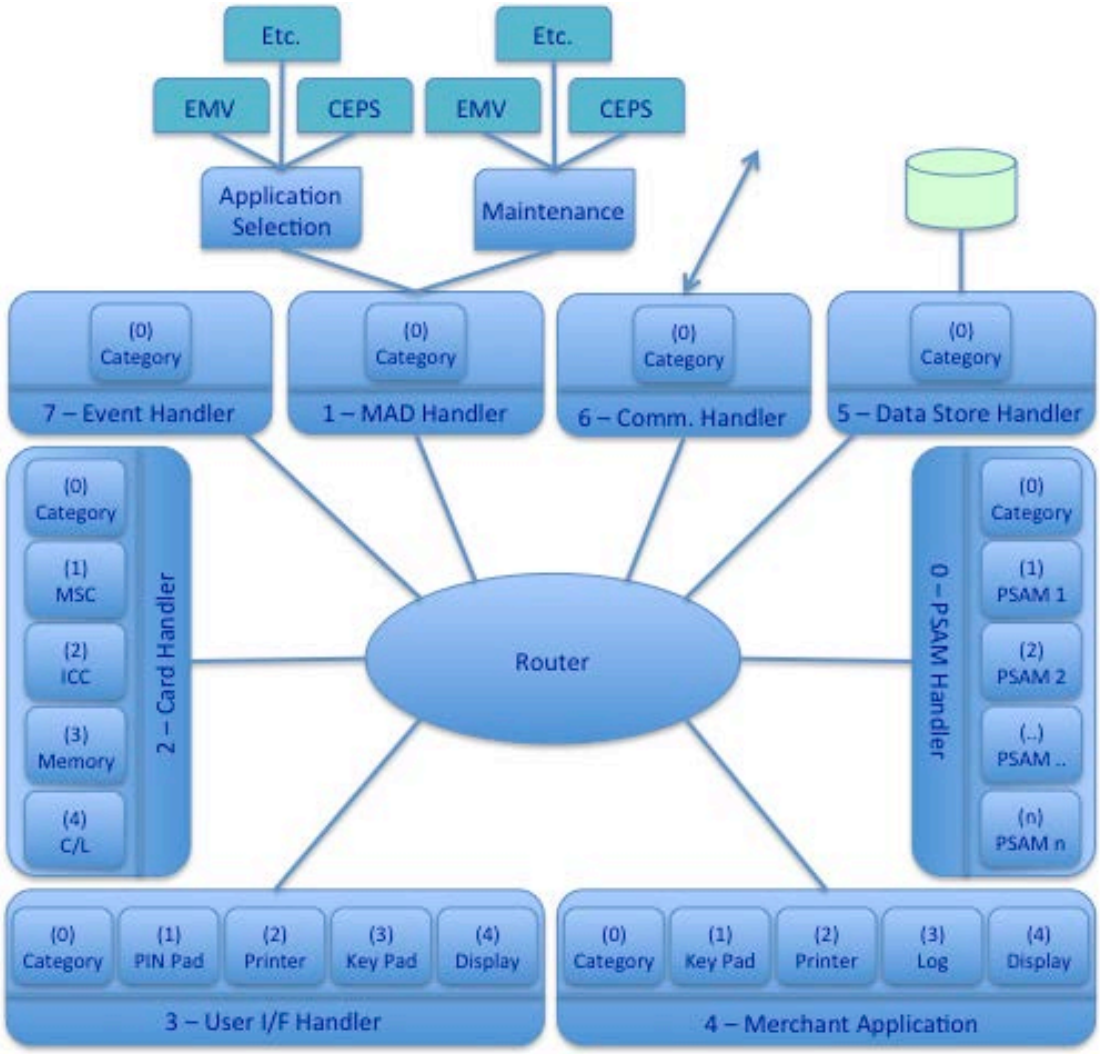

Figure 1: Logical Components of the Terminal Architecture for PSAM Applications

4. Functional Requirements

This section provides a detailed description of the various structural components comprising the TAPA architecture and their respective functional and processing requirements. For each component, the command and response sets are defined as well as required processing and possible Response Codes.

*Note: Certain requirements are only applicable if the terminal uses the PSAM controlled PIN Pad processing specified in section 14.4. Such requirements are prefaced with the phrase **"PIN Pad Requirement"**.*

4.1 The Router

The Router functions as a communication channel and is the central entity to which all Handlers are attached. The communication channel may be implemented on a variety of hardware and software protocols such as an internal bus (in the case of a stand-alone POS device), a LAN in a multi-lane controller environment, or TCP/IP if architectural components are remotely distributed on the Internet or an intranet. In order to accommodate all of the aforementioned configurations, the Router must function purely as a simple transport mechanism that is application and device independent.

The Router is primarily responsible for the transfer of messages (commands and responses) from one Handler to another. In this way, the Router functions as a pure transport mechanism -- ensuring that messages received from an origination Handler are delivered to the destination Handler as specified in message address fields. By means of transferring messages from one Handler to another, the Router also effectively passes application control from one Handler to another. Before a destination Handler responds to a message, it may initiate a series of message exchanges to Handlers other than the originator of the message.

Aside from validating the address fields of a message to ensure it can be delivered to the specified destination, the Router remains ignorant of the data content of the messages it conveys. This transparency allows the Router to remain application independent. It is the responsibility of the individual Handlers to validate, process, and respond to the contents of the received message.

4.1.1 Functional Requirements

4.1.1.1 The Router must validate the source and destination address and sub-address fields in messages received from an origination Handler to ensure they are defined in the specification and are also supported by the terminal application. This validation process will ensure that the message can be delivered to the appropriate destination Handler.

4.1.1.2 The Router must not intervene or prevent routable messages from being delivered to a destination Handler.

4.1.1.3 The Router should implement some error checking mechanism to ensure the data integrity of messages exchanged between handlers across physical interfaces. The mechanism implemented is left to the discretion of the terminal developer; however, LRC, CRC, or MAC checking is frequently used.

4.1.2 Error Handling

4.1.2.1 The Router must generate error messages if either the source or destination address is invalid.

Table 1 provides a listing of Router generated Response Codes.

Table 1: Router Response Codes

Response Code	Description
'FFF0'	*Invalid Source Address*: the source address does not match the originator of the message.
'FFF1'	*Invalid Destination Address*: the message cannot be delivered because it contains an invalid destination address.

4.2 The Handlers

Handlers are logical entities responsible for either managing the interface to a specific hardware component or peripheral device, or responsible for performing specific operational control functions. There are three types of Handler: Device

Handlers, the Multi-Application Driver Handler and the Event Handler.

4.2.1 Device Handlers

Device Handlers are logical entities responsible for managing the bi-directional interface to a hardware component or peripheral device such as a card reader, modem, or customer keypad. A device Handler may manage either a simple or sophisticated interface. A Device Handler may be a simple mechanism in the sense that it may only transport a message received from the Router to the device it operates and automatically respond to the Handler that originated the message. In the other extreme, it may be a very sophisticated piece of software that analyzes the content of incoming and outgoing messages in addition to manipulating the message addresses.

The existence of a Device Handler serves the purpose of shielding interface and implementation details for the specific device it supports from other components in the system. A Device Handler should therefore perform only those functions directly associated with support of the hardware or peripheral equipment it is intended to support -- thereby allowing Handlers to remain as application independent as possible. Each of these Handlers is described in greater detail in subsequent sections of this document. Note that additional Handlers may be identified and defined as necessary to accommodate a particular implementation or environment.

4.2.2 Multi-Application Driver Handler

The Multi-Application Driver Handler is the operative software component of the terminal application which, in addition to selecting and controlling transaction processing, performs a number of operational and maintenance functions. The Multi-Application Driver Handler is described in greater detail in Section 5.0.

4.2.3 Event Handler

The Event Handler is a logical entity responsible for receiving notification of asynchronous external events and providing notification of these events to the terminal's application processing.

The Device Handlers will forward messages to the Event Handler when events occur such as a card insertion or a key press is detected. A MAD Handler application, or an entity (such as a PSAM application) to which the MAD Handler has

delegated control, may retrieve the events and take appropriate actions.

4.2.4 General Characteristics

The following characteristics are common to all classes of Handler:

- Handlers are connected to and communicate directly with the Router. All messages either originating from or destined for other Handlers must be exchanged via the Router.
- Handlers are application independent in that they may be accessed by multiple applications resident in the terminal.
- Each Handler is assigned a unique logical address as specified in Table 2. The handler category is assigned sub-address '00', which permits it to be separately addressed as a logical component.
- A Handler may be able to respond with plaintext as well as enciphered data.

Additional Handler and device addresses may be assigned as needed.

Note: Secure communication between the (Secure) Card Reader and the PSAM to enable Card Data Protection (CDP) may be used for PCI compliance. For this, the response from the Handler to the PSAM is enciphered. Requesting an enciphered response is specified by setting the MSB of the Handler sub-address to '1'.

Table 2: Handler Address Assignments

Handler category	Address	Sub-address	Handler
PSAM	'00'	'00'	The PSAM Handler
		'01'	PSAM Handler 1
		'02'	PSAM Handler 2
		'03' - '7F'	PSAM Handler 3 – PSAM Handler 127
		'80' - 'FF'	PSAM Handler with enciphered response

Handler category	Address	Sub-address	Handler
MAD	'01'	'00'	The Multi-Application Driver Handler
		'80'	The Multi-Application Driver Handler, with enciphered response
		'01' - '7F' and '81'-'FF'	Reserved for Future Use
Card	'02'	'00'	The Card Handler
		'01'	Magnetic stripe Reader
		'02'	Processor Card Reader
		'03'	Memory Card Reader
		'04'	Contactless Card Reader
		'05' - '7F'	Reserved for Future Use
		'80'	The Card Handler with enciphered response
		'81'	The Magnetic stripe Reader with enciphered response
		'82'	The Processor Card Reader with enciphered response
		'84'	The Contactless Card Reader with enciphered response
		'85' – 'FF'	Reserved for Future Use
User Interface	'03'	'00'	The User Interface Handler
		'01'	PIN Pad
		'02'	Customer Printer
		'03'	Customer Key Pad
		'04'	Customer Display
		'05' - '80'	Reserved for Future Use
		'81'	PIN PAD with enciphered command/response
		'82' –'FF'	Reserved for Future Use

Handler category	Address	Sub-address	Handler
Merchant Application	'04'	'00'	The Merchant Application Handler
		'01'	Merchant Key Pad
		'02'	Merchant Printer
		'03'	Merchant Log
		'04'	Merchant Display
		'05' - 'EF'	Reserved for Future Use
		'F0' - 'FF'	Serial ports
Data Store	'05'	'00'	The Data Store Handler
		'01' - 'FF'	Reserved for Future Use
Communication	'06'	'00'	The Communication Handler
		'01' - 'FF'	Reserved for Future Use
Event	'07'	'00'	The Event Handler
		'01' - 'FF'	Reserved for Future Use
RFU	'08 - 'FF'	any	Reserved for Future Use

4.2.5 Functional Requirements

This section defines the functional requirements to be supported by all Handlers.

4.2.5.1 A Handler that receives a command must always respond to the originator of that command (except as noted in Section 13.2.1).

4.2.5.2 Prior to generating a response, a destination Handler must be permitted to issue commands to Handlers other than the originator of the initial command.

4.2.5.3 A Handler must only respond to the originator of the command after all required subsequent dialogue has been completed with other Handlers.

4.2.5.4 After sending a command, a Handler must not send another command to the same destination or for the same thread prior to receiving a

response (except as noted in Section 13.2.1). (See section 5.5 for a discussion of multi-threading).

4.2.5.5 When constructing a response, the responding Handler must use the source address and sub-address of the command message as the destination address and sub-address of the response. The Thread Identifier (ID_{THREAD}) from the original command message must be included in the response.

4.2.5.6 A Handler should be limited to performing only those functions as needed to either directly support a particular device or manage a particular operation.

4.2.5.7 When a Handler receives a command message from another terminal component, it must return a response to the requesting Handler.

A successful response must contain a Response Code of '0000'.

If the command has not been processed correctly, the handler must return the appropriate Response Code.

4.2.5.8 All Handlers must be able to receive and process messages with a message data length of at least 1024 bytes ($L_{DATA} \leq$ '0200'). (Note: terminal applications must only rely on the ability to send longer messages in a proprietary environment).

4.3 Message Handling

Command and informational data is exchanged between components of the application by use of Terminal messages as defined in Table 3.

Table 3: Terminal Message Format

Field	Length	Format	Contents
Destination Address	2	binary	Handler address || Handler Sub-address
Source Address	2	binary	Handler address || Handler Sub-address

Message Type	1	binary	The message type of the command message, or 'FF' to indicate that it is a response message.
ID_{THREAD}	1	binary	Thread Identifier assigned by the Multi Application Driver Handler.
L_{DATA}	2	binary	Length of the message data field
Message Data	var.	var.	Data passed between Handlers. This field may not always contain data.

Message types can be either *handler-dependent* or *handler-independent*. Handler-independent messages are messages that are common to and supported by multiple handlers within the TAPA architecture. Handler-dependent messages are those supported only by specific Handlers.

The following fields comprise the Handler to Handler message format:

- The Destination Address field contains the address and sub-address of the Handler to which the Router is requested to deliver the message. Setting most significant bit in the destination sub address requires that the response is encrypted.
- The Source Address field contains the address and sub-address of the Handler that generated the message.
- The Message Type field is used to identify the type of command or response being sent. Table 4, Table 5 and Table 6 provide a list of currently defined Message Types.
- The Thread Identifier field is supplied by the Multi-Application Driver and is used in systems where the PSAM Handler can manage several transactions concurrently. In single transaction systems, this field can be defaulted.
- The L_{DATA} field provides the length of the data contained in the Message Data field.
- The Message Data field contains the data being transferred between Handlers in command and response messages. Note that, in a response message, the data includes a 2-byte Response Code.

Table 4: Terminal Messages, Handlers 0-2

		Handler Category		
Message Type	**Command/Message Name**	**PSAM Handler (0)**	**MAD Handler (1)**	**Card Handler (2)**
'40'	Read Magnetic Stripe			√
'41'	Write Magnetic Stripe			√
'42'	ICC Command	√		√
'43'	ICC Power-On	√		√
'44'	ICC Power-Off	√		√
'45'	ICC Query	√		√
'46'	Verify Offline PIN			√
'47'	ICC Command partially encrypted	√		√
'48'	ICC Command fully encrypted	√		√
'F0'	Open Handler	√		√
'F1'	Close Handler	√		√
'F3'	Write Handler String	√		√
'F4'	Read Handler String	√		√
'F5'	Get Handler Addresses	√		√
'FF'	Response Message	√	√	√

Table 5: Terminal Messages, Handlers 3-5

		Handler Category		
Message Type	**Command/Message Name**	**User Interface Handler (3)**	**Merchant Application Handler (4)**	**Data Store Handler (5)**
'60'	Confirm Amount	√		
'61'	Display Message	√	√	
'63'	Print Message	√	√	
'64'	Purge Print Buffer	√	√	
‘65’	Get Key Check Value	√		
‘66’	Verify PSAM Public Key Certificate	√		
‘67’	Get PIN Pad Public Key Record	√		
‘68’	Submit Initial Key	√		
‘69’	Initiate PIN Entry	√		
‘6A’	Get PIN	√		
‘6C’	Terminate PIN Entry	√		
'80'	Get Amount	√	√	
'81'	Transaction Completed		√	
‘82’	Funds Available	√	√	
'90'	Create File			√
'91'	Delete File			√
'92'	Add File Record			√
'93'	Get File Record			√
'94'	Update File Record			√
'95'	Find and Get File Record			√

		Handler Category		
Message Type	**Command/Message Name**	**User Interface Handler (3)**	**Merchant Application Handler (4)**	**Data Store Handler (5)**
'96'	Delete File Record			√
'97'	Find and Delete File Record			√
'98'	Clear File			√
'F0'	Open Handler	√	√	√
'F1'	Close Handler	√	√	√
'F3'	Write Handler String	√	√	√
'F4'	Read Handler String	√	√	√
'F5'	Get Handler Addresses	√	√	√
'FF'	Response Message	√	√	√
'01' - '3F'	Reserved for Proprietary Use			
All Others	Reserved for Future Use			

Table 6: Terminal Messages, Handler 6-7

		Handler Category	
Message Type	**Command/Message Name**	**Comm. Handler (6)**	**Event Handler (7)**
'B0'	Initiate Communication Session	√	
'B1'	Terminate Communication Session	√	
'C0'	Add Event to Queue		√
'C1'	Get Event from Queue		√

		Handler Category	
Message Type	**Command/Message Name**	**Comm. Handler (6)**	**Event Handler (7)**
'C2'	Find Event on Queue		√
'C3'	Flush Event Queue		√
'F0'	Open Handler	√	√
'F1'	Close Handler	√	√
'F3'	Write Handler String	√	
'F4'	Read Handler String	√	
'F5'	Get Handler Addresses	√	√
'FF'	Response Message	√	√
'01' - '3F'	Reserved for Proprietary Use		
All Others	Reserved for Future Use		

4.3.1 Time-out Management

For most messages, the recipient is expected to perform the requested action and respond when the action is complete.

4.3.1.1 If the requested action cannot be performed, or the requested data is not available, then the recipient must respond with an error response.

4.3.1.2 If a message includes a *Timer Flag*, and the requested action cannot be performed, the recipient must wait either until the action can be performed, or until the maximum time, as indicated (in milliseconds) in the *Time* field, has passed. If all requested data is not available at the end of the wait period, the available data is returned.

For example, if the message is a Read request for 200 bytes of data from the Communications Handler – and at the end of the Wait time a 150 byte block of data is available – then the

available 150 bytes must be returned in the response.

4.3.1.3 If either the Timer Flag is not set, or if the Time field contains a value of binary zeros, a response is required either when action is complete or when it is known that it cannot be completed.

If there is no malfunctioning identified but the service cannot be provided, the requester may wait indefinitely.

4.3.2 Exception Handling

If a handler is not able to perform the requested function, it must respond to the sender with a Response Message containing only the appropriate Response Code in the message data. Specific Response Codes for each message type are defined in these specifications. The terminal developer may assign additional proprietary Response Codes as needed.

The recipient of a response message must handle all of the defined Response Codes appropriately. Any unknown Response Codes must be treated as an error.

4.4 Handler-Independent Messages

This section describes those commands that can be processed by any Handler defined within the TAPA architecture.

4.4.1 Get Handler Addresses

The Get Handler Addresses command is used to obtain a list of active and available addresses within a handler category.

4.4.1.1 The Get Handler command must conform to the format defined in Table 7.

Table 7: Get Handler Addresses command

Field	Value	Length
Destination Address	XX00 – this command is sent to a handler category (for example, the PSAM Handler category).	2
Source Address	Any	2
Message Type	'F5'	1

ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0000'	2

4.4.1.2 The Get Handler Addresses response must conform to the format defined in Table 8

Table 8: Response to Get Handler Addresses command

Field	**Value**	**Length**
Destination Address	Any	2
Source Address	XX00	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	3+ $CNT_{SUBADDRESS}$	2
$CNT_{SUBADDRESS}$	Number of sub-addresses returned	1
$SUBADDRESS_N$	Available sub-address N in category	1
Response Code	Response Code	2

4.4.1.3 The Response Codes applicable to the Get Handler Addresses command are listed in Table 9.

Table 9: Response Codes to Get Handler Addresses command

Response Code	**Description**
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.

4.4.2 Open Handler

The Open Handler command is used to initialize or activate a Handler. The initialization process may include the clearing of buffers or performing other maintenance procedures deemed necessary by the terminal developer.

4.4.2.1 All Handlers must be in the closed state before terminal start- up.

4.4.2.2 The Open Handler command must conform to the format defined in Table 10.

Table 10: Open Handler command

Field	Value	Length
Destination Address	Any	2
Source Address	Any	2
Message Type	'F0'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0000'	2

4.4.2.3 The Open Handler response must conform to the format defined in Table 11.

4.4.2.4 A response of "handler must be opened" must be returned if a Handler receives a terminal message prior to being opened.

4.4.2.5 A response of "handler already opened" must be returned if a Handler receives the Open Handler command while already in open status.

4.4.2.6 After successfully processing the Open Handler command, the handler must be capable of receiving and processing messages.

Note that the handler category is a separate logical unit from the device handlers within the category. Opening the handler category does not open other handlers within the category. (For example, opening the Card Handler category does not automatically open the Magnetic Stripe Reader.)

Table 11: Response to Open Handler command

Field	Value	Length
Destination Address	Any	2
Source Address	Any	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	2
Response Code	Response Code	2

4.4.2.7 The Response Codes applicable to the Open Handler command are defined in Table 12.

Table 12: Response Codes to Open Handler command

Response Code	Description
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF8'	*Handler is already open*
'FFFA'	*Handler cannot be opened*: an error indicating that the Handler cannot be opened.

4.4.3 Close Handler

The Close Handler command is used to close or deactivate a previously opened Handler. Once closed, the affected Handler is no longer accessible for processing additional commands, except for the Open Handler command. Note: if the Communication handler is closed while a communication session is active, that communication session must be immediately terminated.

4.4.3.1 The Close Handler command must conform to the format defined in Table 13.

Table 13: Close Handler command

Field	Value	Length
Destination Address	Any	2
Source Address	Any	2
Message Type	'F1'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0000'	2

4.4.3.2 The Close Handler response must conform to the format defined in Table 14.

4.4.3.3 A response of "handler already closed" must be returned when a destination Handler receives a Close Handler command while already in closed status.

4.4.3.4 A response of "Handler cannot be closed" must be returned when the physical device processing cannot be terminated. For example, the communication handler will return this response when the modem will not hang up.

Table 14: Response to Close Handler command

Field	Value	Length
Destination Address	Any	2
Source Address	Any	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1

L_{DATA}	'0002'	2
Response Code	Response Code	2

4.4.3.5 The Response Codes applicable to the Close Handler command are defined in Table 15.

Table 15: Response Codes to Close Handler command

Response Code	Description
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF9'	*Handler already closed*
'FFFC'	*Handler cannot be closed*: an error indicating that the Handler cannot be closed.

4.4.4 Write Handler String

The Write Handler String command is used to send a data string to a terminal device. For example, if sent to the User Display, the data contained in the message will be a displayable text string. Note that some handlers will not support this function.

4.4.4.1 The Write Handler String command must conform to the format defined in Table 16.

4.4.4.2 If the destination address is for either a Display or a Printer device, the data string must be coded as indicated in the Code Table Index.

4.4.4.3 **PIN Pad requirement:** If the Write Handler String command is sent to the User Interface Display Handler while the Secure Cryptographic Device (SCD) is in PIN Entry State, the command must include the SP_{MAC}. The SCD must authenticate the message using the $KSES_{MAC}$ of the PSAM that initiated the PIN Entry.

4.4.4.4 **SCD requirement:** If the Data String in the Write Handler String command is enciphered, the SCD must decipher the Data String using the $KSES_{DATA}$ of the PSAM that initiated the command.

Table 16: Write Handler String command

Field	Value	Length
Destination Address	Any	2
Source Address	Any	2
Message Type	'F3'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0006' + Length of Message Data + L' SP_{MAC}	2
Timer Flag	'00' = Not Timed '80' = Timed	1
Time	Time-out value in milliseconds	4
Code Table Index	If the destination is either a Display or a Printer device, this field is an index to the character set being used. If the destination is neither a Display nor a Printer device, this field is not used and may be set to zeros. If the originator of this command has enciphered the data string, the most significant bit of this byte is set to '1'.	1
Data String	Data to be sent to the Handler	var.
SP_{MAC}	MAC on Destination Address – Data String, computed using $KSES_{MAC}$. Only present if message must be MAC'ed	0 or 8

4.4.4.5 All Display and Printer device Handlers must support the Common Character set defined in reference 6, EMV Book 4, Annex B.

4.4.4.6 The Write Handler String response must conform to the format defined in Table 17.

4.4.4.7 **SCD requirement:** If the Data String in the Write Handler String command is required to be enciphered, the SCD must encipher the Data String using the $KSES_{DATA}$ of the PSAM that initiated the command.

4.4.4.8 If the Handler does not support this function, it must return a Response Code of *Unsupported Operation*.

Table 17: Response to Write Handler String command

Field	Value	Length
Destination Address	Any	2
Source Address	Any	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	1
Response Code	Response Code	2

4.4.4.9 The Response Codes applicable to the Write Handler String command are listed in Table 18.

Note*: The Response Codes defined in Table 18 are generic Response Codes and do not reflect handler-specific Response Codes (such as 'No Connection for the communication handler'), nor proprietary Response Codes that may exist for specific operating environments.*

Table 18: Response Codes to Write Handler String command

Response Code	Description
'FF35'	*Code Table not supported.*
'FF82'	*Authentication Error (MAC validation failed)*

Response Code	Description
'FFF2'	*Time-out*: the requested operation is valid, but some external event necessary for the proper execution failed to arrive in time.
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF4'	*Handler must be initialized*: the Handler cannot perform the requested action until it has been initialized.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

4.4.5 Read Handler String

The Read Handler String command is used to retrieve requested data from another Handler. For example, this message may be used to retrieve a block of data from the Communication Handler. Note that some handlers will not support this function.

4.4.5.1 The Read Handler String command must conform to the format defined in Table 19.

Table 19: Read Handler String command

Field	Value	Length
Destination Address	Any	2
Source Address	Any	2
Message Type	'F4'	1

ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0007'	2
Timer Flag	'00' = Not Timed '80' = Timed	1
Time	Time-out value in milliseconds	4
Code Table Index	If the destination of the message is a Key entry device, this field is an index to the character set in which the response data must be returned. If the destination is not a Key-entry device, this field is not used and may be set to zeros. If the originator of this command requires the response to be enciphered, the most significant bit of this byte is set to '1'.	1
Len	Number of bytes to read	2

4.4.5.2 The Read Handler String response must conform to the format defined in Table 20.

4.4.5.3 If the responding Handler is a key-entry device, the Returned Data String must be coded using the character set specified in the Code Table Index. If the character set is not supported, the Handler must respond with the appropriate Response Code ('FF35').

4.4.5.4 All key-entry device Handlers must support the Common Character set defined in reference 6, EMV Annex C.

4.4.5.5 If the Handler does not support this function, it must return a Response Code of *Unsupported Operation*.

Table 20: Response to Read Handler String command

Field	**Value**	**Length**
Destination Address	Any	2
Source Address	Any	2
Message Type	'FF'	1

ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	Len + 2	2
Returned String	Returned Data String	Len
Response Code	Response Code	2

4.4.5.6 The Response Codes applicable to the Read Handler String command are defined in Table 21.

Note: The Response Codes defined in Table 21 are generic Response Codes and do not reflect handler-specific Response Codes (such as 'No Connection for the communication handler'), nor proprietary Response Codes that may exist for specific operating environments.

Table 21: Response Codes to Read Handler String command

Response Code	Description
'FF35'	*Code Table not supported.*
'FFF2'	*Time-out*: the requested operation is valid, but some external event necessary for the proper execution failed to arrive in time.
'FFF4'	*Handler must be initialized*: the Handler cannot perform the requested action until it has been initialized.
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

4.4.6 Summary

4.4.6.1 The common Handler commands are listed in Table 22. Any handler must support those with a Destination address marked "any". The

specified handlers must support those with specific addresses.

Table 22: Handler-Independent commands

Destination Address	Message Type	Description
Any	'F0'	Open Handler
Any	'F1'	Close Handler
Any	'F3'	Write Handler String
Any	'F4'	Read Handler String

5. The Multi-Application Driver Handler

The MAD Handler is the interface to the application software. This software is responsible for running the transactions to be performed on the terminal.

The application software may also include functions such as Terminal initialization, Application Maintenance and Acquirer Processing.

5.1 Application Selection

The first step in transaction processing consists of Application Selection. This selection process must consider the type of media being presented such as processor cards, memory cards or magnetic stripe cards and therefore can be based on AID, ATR, insertions, a button pressed or other. The result is that a mutual application is agreed upon, often indicated by an AID.

During terminal initialization, the MAD-Handler has constructed a cross-reference table that associates each supported card application with the identity of the terminal application, which must be used for processing. The terminal application is denoted by the $ID_{PSAMAPP}$. After selecting the card application, the MAD-Handler uses this cross-reference to invoke the correct terminal application. (See section 15.1.35 for a description of the $ID_{PSAMAPP}$.)

5.2 Terminal Initialization

As part of normal terminal start-up, the terminal application must perform an initialization procedure to ensure that all logical components are present and functioning normally. The terminal initialization process must include the following procedures:

5.2.1.1 The MAD-Handler must open any necessary Device Handlers through the issuance of multiple Open Handler commands. The MAD-Handler can determine the occupied sub-addresses, if any, by using the Get Handler Addresses command.

5.2.1.2 All PSAMs must be reset by sending an ICC Power-On command to each occupied sub-address. In the response the MAD Handler

receives the ATR and, if present, the Historical Bytes.

5.2.1.3 The MAD Handler must issue the Start-up PSAM command to each application at each occupied PSAM sub-address.

5.2.1.4 The MAD Handler must issue the Get Supported AIDs command to each application at each occupied PSAM sub-address.

5.2.1.5 Prior to completing the configuration process, the MAD Handler may be required to send one or more application specific start-up commands to the PSAM. The format and content of these commands are outside the scope of this specification, and must be defined in the appropriate application specifications.

5.2.1.6 The terminal must successfully perform the initialization sequence prior to initiating any card transactions.

Figure 2 illustrates the PSAM Initialization process. The corresponding behaviour of the PSAM will be described in section 10.1. Section 10.3 contains details about the commands.

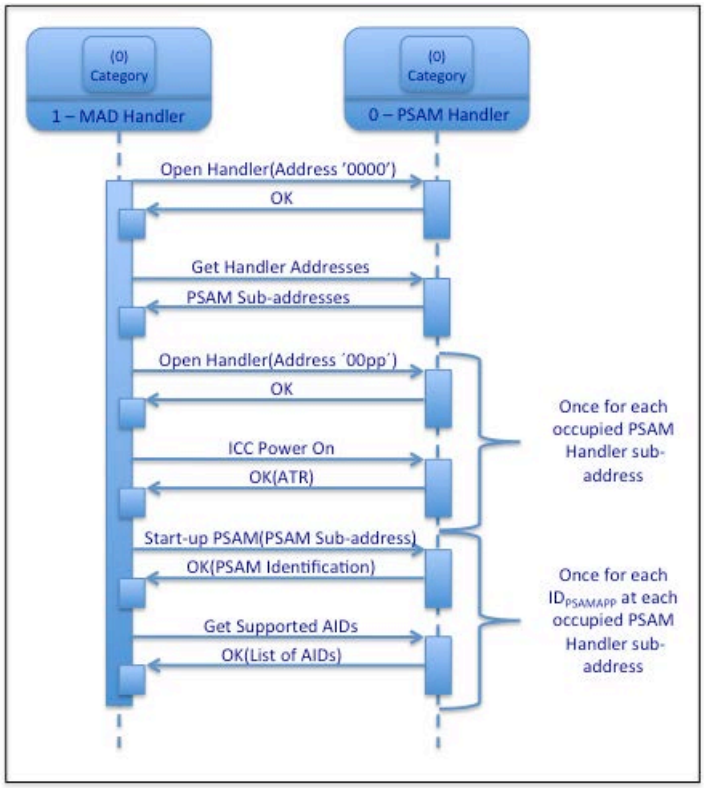

Figure 2: Terminal PSAM Initialisation

5.3 Terminal Shutdown

Certain terminals (in particular battery-operated devices) require the ability to withdraw power from the PSAM between cardholder transactions. When a card is inserted into the device, power is restored to the PSAM and processing commences.

Such terminals require that the PSAM be restarted very quickly, without a lengthy initialization.

5.3.1.1 In order to ensure that the PSAM application is able to save all outstanding data, and be easily restarted, the terminal must send a PSAM Shut-down command to each PSAM application, and receive a response, prior to withdrawing power.

5.4 Terminal Control

When the terminal is initialized, the MAD Handler Application software has control over the terminal's functioning. The MAD Handler applications may temporarily delegate control to application processing in other devices – in particular the PSAM. Control is transferred through an application-specific message or command sent from a terminal application.

The delegated component will then take over control of the terminal's functions – issuing messages to devices and receiving responses. When its processing is completed, it will send a response to the MAD Handler application, which resumes control.

5.5 Multi-Threading

The TAPA message structure incorporates support for "multi-threading", where multiple concurrent transactions may be in process, in varying stages of completion.

An example of an environment where this is required is a distributed POS environment, consisting of a "server" (which contains one or more PSAMs, the Data Store and the connection to the acquirer's host system) and multiple remote terminals, each of which contains a card reader and a user interface.

In such a system, the MAD Handler application functions are distributed between the remote terminal and the server. The messaging between the distributed and centralized parts of the MAD Handler applications is proprietary to the device developer and outside the scope of this specification. This messaging must however include identification of the remote terminal.

5.5.1.1 In order to support a multi-threading environment, the MAD Handler must assign a unique identifier (ID_{THREAD}) to each currently active transaction, which must be used in all Terminal Messages relating to that transaction. The ID_{THREAD} value may be reused after a transaction has been terminated.

5.6 Exception Handling

There are a variety of exception conditions that can occur during application processing in a POS device. This section addresses the exception conditions that affect the interface between the MAD Handler application and the PSAM application, and defines a set of functions that must be provided by the PSAM and terminal applications in order to allow the other component to attempt recovery.

Exception conditions fall broadly into the following categories:

- An inability to perform a particular transaction because of a problem detected during the dialogue with the consumer card. This is normally not a problem with the POS device,

and simply requires completing the transaction in a defined manner.

- A temporary problem with a particular device. This may particularly be the case in a distributed environment where some resources are shared. This sort of problem may possibly be overcome by retrying the failed function.
- A hardware or software problem with the POS device that prevents continued correct functioning. The resolution to this sort of problem is device and implementation dependent, and outside the scope of this specification.

Exception processing is specific to each TAPA application, i.e. $ID_{PSAMAPP}$.

6. The Card Handler

The Card Handler is responsible for managing the interface to an integrated or peripheral card reading device. Currently defined card reading devices include the magnetic stripe reader, IC card reader, memory card reader and contactless card reader. Each card reading device and its associated Card Handler sub-address is defined in Table 2.

6.1 Commands sent to the Magnetic Stripe Reader

6.1.1 Read Magnetic Stripe

The Read Magnetic Stripe command is used to read data from one or more ISO magnetic tracks. The command supports enciphered as well as clear text response.

6.1.1.1 The Read Magnetic Stripe command must conform to the format defined in Table 23.

Table 23: Read Magnetic Stripe command

Field	Value	Length
Destination Address	'0201' / '0281'	2
Source Address	Any	2
Message Type	'40'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0007'	2
Timer Flag	'00' = Not Timed '80' = Timed	1
Time	Time-out value in milliseconds	4
u	Track Identifier of ISO track(s) to be read as shown in Table 24	1
Len	Length of track data to be read	1

The parameter track *u* is the hex value of the ISO identifier of the magnetic stripe track(s) to be read (as illustrated in Table 24). See reference 9, ISO/IEC 7813 for a description of the

format of this data element. If the magnetic stripe track data is to be returned in enciphered form, the highest bit in *u* is set. A particular device is not required to support all of these possibilities.

Table 24: Track Assignment

Track	*u*	Tracks to be read
1	'01', '81'	ISO1
2	'02', '82'	ISO2
3	'03', '83'	ISO3
12	'0C', '8C'	ISO1 and ISO2
13	'0D', '8D'	ISO1 and ISO3
23	'17', '97'	ISO2 and ISO3
123	'7B', 'FB'	ISO1, ISO2 and ISO3

Len is its maximum length, in bytes. On return, *len* gives the actual length of the string read for each track. If more data is available, a Response Code of "output buffer overflow" is returned, together with the maximum length of data specified. See reference 9, ISO/IEC 7813 for a description of the track data formats.

This command returns when either track data is available or the time-out is reached. The operation may return tracks that have been read since the last execution of this command and stored in a buffer. If the Handler does not have track buffering capability, only one swipe of the card must be buffered and all the track buffers will be cleared after their contents have been returned by this command (even if all tracks were not requested).

The format returned by the Read Magnetic Stripe command depends on whether or not the data are encrypted.

For plaintext data each track is a track number (1, 2 or 3), a length byte and the data whose size is specified by the length byte. If multiple tracks are requested, there are multiple instances of the above structure concatenated in the order they were requested. The data is returned in ASCII format with STX/ETX and LRC delimiters removed. Non-BCD digits are

left unconverted. (Please see Figure 3 below for an example.)

For example:

BYTES FROM CARD ==> BYTES DELIVERED TO APPLICATION

B1 23 4D 78 5F xx => 31 32 33 34 0D 37 38 35

^ ^ ^LRC

STX ETX

Figure 3: Example of bytes read from Magnetic Stipe

6.1.1.2 A response of "unsupported operation" must be returned if the reader does not support one or more of the requested tracks. If the requested tracks are supported by the reader, but are not present on the card swiped, then a response of "successful operation" is returned and the message contains those tracks that are available on the card.

6.1.1.3 If an error occurs while reading one or more of the requested tracks, a Response Code of "transmission error" must be returned with the length field of the corresponding tracks in the returned message set to zero. In this case, the data of the tracks that were read successfully is available in the returned message.

6.1.1.4 The Read Magnetic Stripe clear text response must conform to the format defined in Table 25.

6.1.1.5 **SCD requirement:** The data of the track(s) must be enciphered using the $KSES_{CDP}$ of the PSAM that initiated the Read Magnetic Stripe command if the highest bit in *u* is set. The data shall be formatted as specified in Table 26.

Note: After a Read Magnetic Stripe function has been performed, the buffer containing the read data must be cleared after returning the response or prior to performing a subsequent read operation. Closing and re-opening the Magnetic Stripe Reader may clear the buffer.

Table 25: Clear text response to Read Magnetic Stripe command

Field	Value	Length
Destination Address	Any	2
Source Address	'0201'/'0281'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	Overall length of data that follows, var.	2
Magnetic Stripe Data	A set of data for each track read	
U	The identifier of the following track data (must be '01', '02', '03', '81', '82', or '83')	1
Len	Length of the data read from this track or length of enciphered track data	1
Data	Track Data read from the specified track or enciphered data.	var.
Response Code	Response Code	2

Table 26: Enciphered response to Read Magnetic Stripe command

Field	Value	Length
Seed	Any, in plain text	4
Enciphered data	Enciphered using $KSES_{CDP}$	
Random number	Any	4
Track data	Data read from the track	var
Padding	'80…', pad to n × 8 byte	var
Response Code	Response Code	2

6.1.1.6 The Magnetic Stripe Reader must be capable of generating the Response Codes to the Read Magnetic Stripe command as defined in Table 27.

Table 27: Response Codes to Read Magnetic Stripe command

Response Code	Description
'FF20'	*Read Error*
'FFF2'	*Time-out*: the requested operation is valid, but some external event necessary for the proper execution failed to arrive in time.
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

6.1.2 Write Magnetic Stripe

The Write Magnetic Stripe is an optional command used to write the entire track data to ISO track 3. (This command may be required by some proprietary applications).

6.1.2.1 The Write Magnetic Stripe command must conform to the format defined in Table 28.

6.1.2.2 **Secure Cryptographic Device requirement:** The data of the track must be deciphered using the $KSES_{CDP}$ of the PSAM that initiated the Read Magnetic Stripe command.

Table 28: Write Magnetic Stripe command

Field	Value	Length
Destination Address	'0201'	2
Source Address	Any	2
Message Type	'41'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0007' + Len	2
Timer Flag	'00' = Not Timed '80' = Timed	1
Time	Time-out value in milliseconds	4
u	Identifier of the track to be written (must be '03', or '83')	1
Len	Length of track data to be written or length of enciphered track data	1
Message Data	Enciphered track data or track data to be written to the magnetic stripe	Len

The parameter track *u* is the identifier of the track to write, as shown in Table 24. The data must be written to track 3 when the user swipes the card. If no card is swiped within the time-out period allowed, a response of "timeout" is returned. Clear text data are given in ASCII format. Enciphered data are formatted as specified in Table 26 and shall be deciphered before written.

The data is written to the card is in the format specified in ISO/IEC 4909: 2006 "Identification cards – Financial transaction cards – Magnetic stripe data content for track 3".

Note: After a Write Magnetic Stripe function has been performed; the buffer containing the written data must be cleared prior to performing a subsequent read operation. Closing and re-opening the Magnetic Stripe Reader may clear the buffer.

For example:

BYTES DELIVERED BY APPLICATION => BYTES TO CARD

31 32 33 34 0D 37 38 35 => B1 23 4D 78 5F xx

^ ^ ^LRC

STX ETX

Figure 4: example of bytes written to Track 3

6.1.2.3 The Write Magnetic Stripe response must conform to the format defined in Table 29.

Table 29: Response to Write Magnetic Stripe command

Field	**Value**	**Length**
Destination Address	Any	2
Source Address	'0201'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	2
Response Code	Response Code	2

6.1.2.4 The Magnetic stripe Reader must be capable of generating the Response Codes to the Write Magnetic Stripe command as defined in Table 30.

Table 30: Response Codes to Write Magnetic Stripe command

Response Code	Description
'FF20'	Unrecoverable Transmission error between reader and magnetic stripe
'FF21'	Output buffer overflow
'FF22'	Write operation failed
'FFF2'	*Time-out*: the requested operation is valid, but some external event necessary for the proper execution failed to arrive in time.
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

6.2 Commands sent to the Processor Card Reader

The interface between the Processor Card Reader and the ICC is a standard command-response protocol as defined in reference 3, ISO/IEC 7816-4 and reference 6, EMV, Book II, section 2.1. The technical interface must be as defined in reference 2, ISO/IEC 7816-3 and reference 6, EMV – Book I.

The Processor Card Reader is responsible for performing the required formatting between the command-response protocol and the Terminal Message formats used among terminal components. Commands are sent to the processor card from other terminal components using the ICC command, which contains within it the command APDU to be delivered to the card.

The response APDU from the processor card must then be sent to the handler that originated the command, embedded

within a Terminal response message.

Note that the Response Code contained in the response message only reflects whether the receiving handler was able to successfully process the ICC command, forward the C-APDU to the processor card, and receive a response. If a response is received from the processor card, then the Processor Card Reader has been able to successfully process the command message. The Status Words in the reply from the processor card will be contained in the Message Data field returned in the response to the ICC command. Figure 5 provides an illustration of the message flows occurring between the Processor Card Reader and the ICC.

6.2.1 Message Handling

Figure 5 illustrates the role of the Processor Card Reader in transmitting messages between the ICC and the terminal. Figure 6 shows the detailed message translation that must be performed.

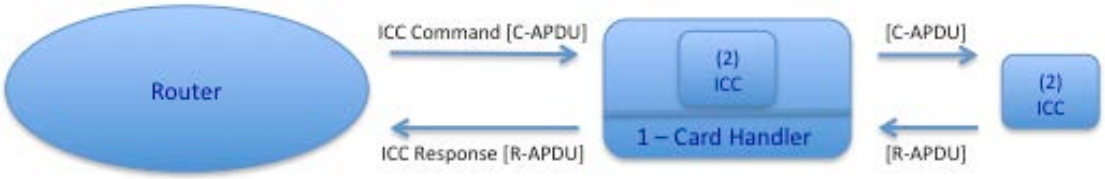

Figure 5: Handler to Processor Card Interface

6.2.1.1 If the interface to the Processor Card is T=0, the Get Response must be implemented as part of the Handler to deal with the requirements for case 2 and case 4 commands. (Please see reference 3, ISO/IEC 7816-4 and reference 6, EMV for further information on T=0 requirements).

6.2.2 Enciphered Messages

The interface to the Processor Card has been extended to handle enciphered data transfer. Commands can be partially (MT = ‘47’) and fully encrypted (MT = ‘48’) commands.

6.2.2.1 The Message Type must be used to show whether or not the command is partially (MT = ‘47’) and fully encrypted (MT = ‘48’) enciphered.

6.2.2.2 The sub-handler address of the destination controls whether or not the response shall be encrypted. The setting of the most significant bit of the destination sub-handler address requests an enciphered response to be generated.

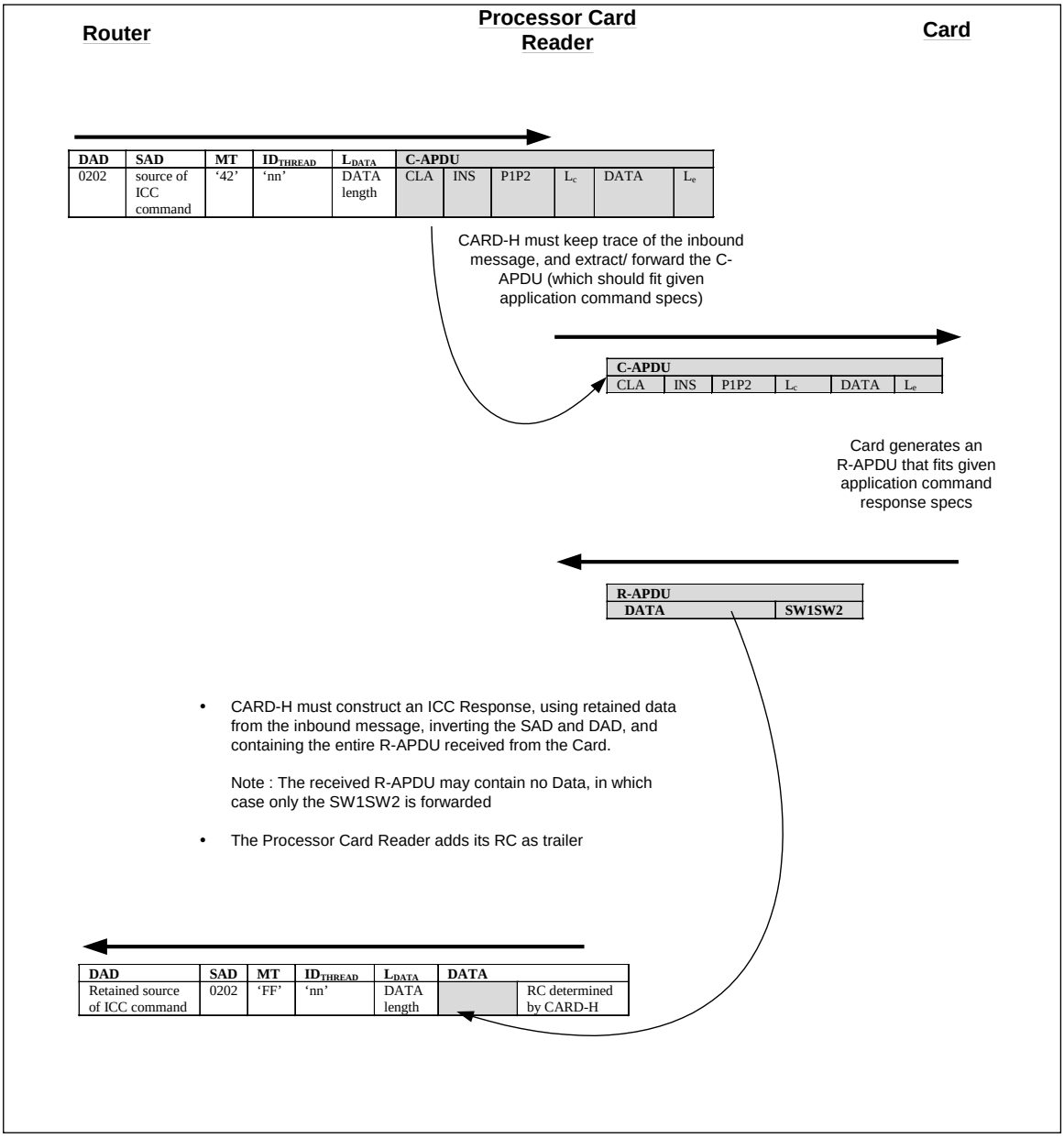

Figure 6: Processor Card Message Translation

6.2.3 ICC Command/Response

The ICC command is used to send a command APDU to an IC card.

6.2.3.1 An ICC command must conform to the format defined in Table 31.

Table 31: ICC command

Field	Value	Length
Destination Address	'0202'	2
Source Address	Any	2
Message Type	'42'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	Length of the Card Command	2
Card Command	C-APDU to be sent to the IC card	L_{DATA}

6.2.3.2 An ICC response must conform to the format defined in Table 32.

6.2.3.3 **SCD Requirement:** The response data must be enciphered using the $KSES_{CDP}$ of the PSAM that initiated the ICC command.

6.2.3.4 When constructing a response message to another Handler, the Processor Card Reader must use the source address and sub-address of the original request message as the destination address and sub-address of the response, set the Message Type to 'FF', and include the Thread Identifier from the original request message.

6.2.3.5 The Processor Card Reader must return the Response Code of “successful operation” if the Handler was able to deliver the C- APDU to the card successfully and receive a response.

6.2.3.6 The Processor Card Reader must return the appropriate Response Code if it is unable to deliver the C-APDU to the IC card or does not get a response.

Table 32: Response to ICC command

Field	Value	Length
Destination Address	Any	2
Source Address	'0202'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	Length of the Card Response + '0002'	2
Card Response	Complete R-APDU from card, including the Status Words	var.
Response Code	Response Code	2

6.2.3.7 The Response Codes applicable to the ICC command are defined in Table 33.

Table 33: Response Codes to ICC command

Response Code	Description
'FF23'	No response from card
'FF24'	No card in reader
'FF26'	Card buffer overflow
'FF28'	Response has no status words
'FF29'	Invalid buffer
'FF2A'	Other card error
'FF2B'	Card partially in reader
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later

Response Code	Description
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

6.2.4 ICC Power-On

The ICC Power-On command is used to apply power to the processor card and execute the 'card reset' function. The Answer to Reset (ATR) message must be returned in the Message Data field.

6.2.4.1 The ICC Power-On command must conform to the format defined in Table 34.

Table 34: ICC Power-On command

Field	Value	Length
Destination Address	'0202'	2
Source Address	Any	2
Message Type	'43'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0000'	2

6.2.4.2 The ICC Power-On response must conform to the format defined in Table 35.

Table 35: Response to ICC Power-On command

Field	Value	Length
Destination Address	Any	2
Source Address	'0202'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	Length of the ATR + '0002'	2
ATR	ATR from IC Card	var.
Response Code	Response Code	2

6.2.4.3 The Response Codes applicable to the ICC Power-On command are defined in Table 36.

Table 36: Response Codes to ICC Power-On command

Response Code	Description
'FF23'	No response from card
'FF24'	No card in reader
'FF2A'	Other card error
'FF2B'	Card partially in reader
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.

Response Code	Description
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

6.2.5 ICC Power-Off

The ICC Power-Off command is used when a transaction involving an IC card has been completed. Use of this command may additionally result in the ejection of the IC card in terminals where this feature is warranted.

6.2.5.1 The ICC Power-Off command must conform to the format defined in Table 37.

Table 37: ICC Power-Off command

Field	Value	Length
Destination Address	'0202'	2
Source Address	Any	2
Message Type	'44'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0000'	2

6.2.5.2 The ICC Power-Off response must conform to the format defined in Table 38.

Table 38: Response to ICC Power-Off command

Field	Value	Length
Destination Address	Any	2
Source Address	'0202'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1

L_{DATA}	'0002'	2
Response Code	Response Code	2

6.2.5.3 The Response Codes applicable to the ICC Power-Off command are defined in Table 39.

Table 39: Response Codes to ICC Power-Off command

Response Code	Description
'FF24'	No card in reader
'FF2B'	Card partially in reader
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

6.2.6 ICC Query

The ICC Query command is issued to the Processor Card Reader in order to determine if a card is physically present in the IC reader.

6.2.6.1 The ICC Query command must conform to the format defined in Table 40.

Table 40: ICC Query command

Field	Value	Length
Destination Address	'0202'	2
Source Address	Any	2
Message Type	'45'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0000'	2

6.2.6.2 The ICC Query response must conform to the format defined in Table 41.

6.2.6.3 The Handler must return the appropriate Response Code if the ICC Query if no card is present.

Table 41: Response to ICC Query command

Field	Value	Length
Destination Address	Any	2
Source Address	'0202'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	2
Response Code	Response Code	2

6.2.6.4 The Response Codes applicable to the ICC Query command are defined in Table 42.

Table 42: Response Codes to ICC Query command

Response Code	Description
'FF24'	*No card in reader*
'FF2B'	*Card partially in reader*
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

6.2.7 Verify Offline PIN

The Verify Offline PIN command is specific to the PIN Pad processing described in section 14.4. Terminals that do not support the PIN Pad Processing do not need to support this command.

The Verify Offline PIN command is an authenticated message, which is used to send a command APDU to the card while the Secure Cryptographic Device (SCD) is in PIN Entry State. The C-APDU is encrypted and embedded in a Verify Offline PIN message with MAC, which is sent to the Processor Card Reader. The command has been updated to ensure that no sensitive are transmitted unencrypted and that data are not encrypted more than once. The format of the command does thus differ, dependent on whether or not the PIN is plaintext or enciphered. Enciphered PIN will use MT ='46' and plaintext PIN will use MT = '42'. Selecting the new mode in the PSAM is performed during the configuration setup.

The Processor Card Reader authenticates the message and decrypts the C-APDU. The subsequent processing is the same as that for the standard ICC Command message. The response

to the Verify Offline PIN command also contains a MAC. The response may be enciphered or plaintext depending on the sub handler address.

6.2.7.1 The Verify Offline PIN enciphered command must conform to the format defined in Table 43.

Table 43: Verify Offline PIN enciphered, command

Field	Value	Length
Destination Address	'0202'	2
Source Address	Any	2
Message Type	'46'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	Length of [C-APDU]	2
[C-APDU]	PIN block enciphered using public RSA key	var.
MAC_{VOPE}	MAC on Dest. Address – [C-APDU], computed using $KSES_{MAC}$	8

6.2.7.2 The Verify Offline PIN plaintext command must conform to the format defined in Table 44.

Table 44: Verify Offline PIN plaintext, command

Field	Value	Length
Destination Address	'0202' / '0282'	2
Source Address	Any	2
Message Type	'47'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
FID	'01', Format Identifier	1
L_{HDR}	Length of (plaintext) Header	2
HDR	Plaintext header	var.
L_{ENCHR}	Length of encrypted data	2

Field	Value	Length
ENC($KSES_{DATA}$)[Data]	Enc($KSES_{DATA}$)[RND(8]||Enc($KSES_{PIN}$)[PIN block]]	var.
L_{TRAI}	Length of trailer	2
Trailer	Optional trailer	Var.
MAC_{VOPP}	MAC on Dest. Address –Trailer, computed using $KSES_{MAC}$	8

6.2.7.3 The Verify Offline PIN plaintext response must conform to the format defined in Table 45.

6.2.7.4 The Secure Cryptographic Device must verify the MAC_{VOP} in the command using the $KSES_{MAC}$, and decrypt the C-APDU using the $KSES_{DATA}$.

6.2.7.5 When constructing a response message to another Handler, the Processor Card Reader must use the source address and sub-address of the original request message as the destination address and sub-address of the response, set the Message Type to 'FF', and include the Thread Identifier from the original request message.

6.2.7.6 The Processor Card Reader must return the Response Code of “successful operation” if the Handler was able to deliver the C- APDU to the card successfully and receive a response.

6.2.7.7 The Processor Card Reader must return the appropriate Response Code if it is unable to deliver the C-APDU to the IC card or does not get a response.

Table 45: Plaintext response to Verify Offline PIN command

Field	Value	Length
Destination Address	Any	2
Source Address	'0202'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'000A' + length of Card response	2
Card Response	Complete R-APDU from card (including SW1-SW2)	var.
MAC_{RVOP}	MAC over Card Response || MAC_{VOPx} (from the command), computed using $KSES_{MAC}$	8
Response Code	Response Code	2

Table 46: Enciphered response to Verify Offline PIN command

Field	Value	Length
Destination Address	Any	2
Source Address	'0282'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'001A' + Length of Enciphered Card Response	2
Seed		4
Enciphered Response	Enc($KSES_{CDP}$)[RND(4) || R-APDU || '80…']	var.
MAC_{RVOP}	MAC over Seed || Enciphered Response || MAC_{VOPx} (from the command), computed using $KSES_{MAC}$	8
Response Code	Response Code	2

6.2.7.8 In addition to the Response Codes defined for the ICC command (in Table 33), the Response Codes defined in Table 47 are applicable to the Verify Offline PIN Command.

Table 47: Response Codes to Verify Offline PIN command

Response Code	Description
'FF82'	*Authentication Error (MAC validation failed)*
'FF87'	*Secure Cryptographic Device not in PIN Entry State*

6.3 Commands sent to Memory Card Reader

The interface to memory cards is proprietary and outside the scope of this specification. In addition to the common handler commands defined in Section 4.4, it is expected that the commands listed in Table 48 with a possible destination address of '0203' will also be used for the Memory Card Reader.

6.4 Commands sent to the Contactless Card Reader

The interface between the Contactless Card Reader and the ICC will use the protocol defined in Reference 4, EMV Contactless. The protocol is outside the scope of this specification.

In addition to the common handler commands defined in Section 4.4, the commands listed in Table 48 with destination address of '0204' will also be used for the contactless card reader.

6.5 Summary

Table 48: Card Handler commands

Destination Address	Source Address	Message Type	Description
'0201'	Any	'40'	Read Magnetic Stripe
'0201'	Any	'41'	Write Magnetic Stripe
'0202','0203',’0204’,’00xx’	Any	'42'	ICC Command
'0202','0203',’0204’,’00xx’	Any	'43'	ICC Power-On
'0202','0203',’0204’,’00xx’	Any	'44’	ICC Power-Off
'0202','0203',’00xx’	Any	'45’	ICC Query
‘0202’,’0204’	Any	‘46’	Verify Offline PIN

7. The User Interface Handler

The User Interface Handler is responsible for managing the interface to all user (customer) related equipment and peripherals, which may include the customer display, customer printer, PIN pad, and customer keypad.

7.1 Messages sent to the User Interface Handler

In addition to the common Handler commands provided in Section 4.4, the User Interface Handler must support the command set outlined in this section.

7.1.1 Display Message

The Display Message command is used to display a pre-defined text message on a display unit (either that of the User Interface Handler or the Merchant Application Handler).

7.1.1.1 The Display Message command must conform to the format defined in Table 49.

7.1.1.2 **PIN Pad requirement:** If the Display Message command is sent to the User Interface Display Handler while the Secure Cryptographic Device (SCD) is in PIN Entry State, the command must include the SP_{MAC}. The SCD must authenticate the message using the $KSES_{MAC}$ of the PSAM that initiated the PIN Entry.

Table 49: Display Message command

Field	Value	Length
Destination Address	'0304' or '0404'	2
Source Address	Any	2
Message Type	'61'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0001' or '0009'	2

Field	Value	Length
Message Code	Message Code to be translated into text by receiving handler	1
SP_{MAC}	MAC on Destination Address – Message Code, computed using the $KSES_{MAC}$.	0 or 8

7.1.1.3 The receiving Handler must convert the 1-byte message code contained in the Display Message command into a predefined text as listed in Table 177. The terminal should use the defined message or the equivalent in the preferred language.

Message Codes '01' – '3F' are defined in reference 6, EMV and are included in Table 177 only for completeness. In order to ensure compliance with EMV for use of that range, the terminal developer should reference the EMV specifications.

7.1.1.4 The Display Message response must conform to the format defined in Table 50.

Table 50: Response to Display Message command

Field	Value	Length
Destination Address	Any	2
Source Address	'0304' or '0404'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	2
Response Code	Response Code	2

7.1.1.5 The Response Codes applicable to the Display Message command are defined in Table 51

Table 51: Response Codes to Display Message command

Response Code	Description
'FF34'	*Unknown Message Code*
'FF82'	*Authentication Error (MAC validation failed)*
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

7.1.2 Print Message

The Print Message command is used to send pre-defined text messages to printer devices.

7.1.2.1 The Print Message command must conform to the format defined in Table 52.

7.1.2.2 The Print Message Code field must contain a 1-byte code as defined in Table 177, which the receiving Handler must interpret and convert to a predefined text message before being transferred to an attached printer.

Table 52: Print Message command

Field	Value	Length
Destination Address	'0302' or '0402'	2
Source Address	Any	2
Message Type	'63'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0001'	2
Message Code	Print Message Code	1

7.1.2.3 The Print Message response must conform to the format defined in Table 53.

Table 53: Response to Print Message command

Field	Value	Length
Destination Address	Any	2
Source Address	'0302' or '0402'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	2
Response Code	Response Code	2

7.1.2.4 The Response Codes applicable to the Print Message command are defined in Table 54.

Table 54: Response Codes to Print Message command

Response Code	Description
'FF31'	Printer out of paper
'FF32'	Printer has signalled an error
'FF33'	Printer does not appear to be connected and online

Response Code	Description
'FF34'	*Unknown Message Code*
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

7.1.3 Confirm Amount

When the User Interface Handler receives this command, it must perform any necessary processing to display and confirm the transaction amount. The particular steps performed will be proprietary and environment dependent.

7.1.3.1 The Confirm Amount command must conform to the format defined in Table 55.

7.1.3.2 **PIN Pad requirement:** If the Confirm Amount command is sent to the User Interface Handler while the Secure Cryptographic Device (SCD) is in PIN Entry State, the command must include the SP_{MAC}. The SCD must authenticate the message using the $KSES_{MAC}$ of the PSAM that initiated the PIN Entry.

Table 55: Confirm Amount command

Field	Value	Length
Destination Address	'0300' (User Interface Handler)	2
Source Address	Any	2
Message Type	'60'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'000C' or '0014'	2
Timer Flag	'00' = Not Timed '80' = Timed	1
Time	Time-out value in milliseconds	4
Amount	Transaction Amount	4
CURR	Currency Code and exponent	3
SP_{MAC}	MAC on Dest. Address – CURR, computed using the $KSES_{MAC}$.	0 or 8

7.1.3.3 The Confirm Amount response must conform to the format defined in Table 56.

7.1.3.4 **PIN Pad requirement:** If the Confirm Amount command is sent to the User Interface Handler while the Secure Cryptographic Device (SCD) is in PIN Entry State, the response must include the $SP_{MAC, R}$, generated by the SCD using the $KSES_{MAC}$ of the PSAM that initiated the PIN entry.

Table 56: Response to Confirm Amount command

Field	Value	Length
Destination Address	Any	2
Source Address	'0300' (User Interface Handler)	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1

Field	Value	Length
L_{DATA}	'0003' or '000B'	2
Amount Confirmed Indicator	'00' = Confirmed '01' = Not Confirmed '02' = Cancelled by user	1
$SP_{MAC, R}$	MAC over Amount Confirmed Indicator || SP_{MAC} (from the command).	0 or 8
Response Code	Response Code	2

7.1.3.5 The Response Codes applicable to the Confirm Amount command are defined in Table 57.

Table 57: Response Codes to Confirm Amount command

Response Code	Description
'FF82'	*Authentication Error (MAC validation failed)*
'FFF2'	*Time-out*: the requested operation is valid, but some external event necessary for the proper execution failed to arrive in time.
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

7.1.4 Purge Print Buffer

The Purge Print Buffer command is used to print and clear data that may be present in a print buffer.

7.1.4.1 The Purge Print Buffer command must conform to the format defined in Table 58.

Table 58: Purge Print Buffer command

Field	Value	Length
Destination Address	'0302'	2
Source Address	Any	2
Message Type	'64'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0000'	2

7.1.4.2 The Purge Print Buffer response must conform to the format defined in Table 59.

Table 59: Response to Purge Print Buffer command

Field	Value	Length
Destination Address	Any	2
Source Address	'0302'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	2
Response Code	Response Code	2

7.1.4.3 The Response Codes applicable to the Purge Print Buffer command are defined in Table 60.

Table 60: Response Codes to Purge Print Buffer command

Response Code	Description
'FF30'	Out of border
'FF31'	Printer out of paper
'FF32'	Printer has signalled an error
'FF33'	Printer does not appear to be connected

Response Code	Description
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

7.1.5 Get Amount

The User Interface Handler may also be able to receive and process the Get Amount and Get Amount Enhanced messages, defined in sections 8.1.1 and 8.1.2.

7.1.5.1 **PIN Pad requirement:** If a Get Amount command is sent to the User Interface Handler while the Secure Cryptographic Device (SCD) is in PIN Entry State, the command must include the SP_{MAC}. The SCD must authenticate the message using the $KSES_{MAC}$ of the PSAM that initiated the PIN Entry.

7.1.6 Funds Available

The User Interface Handler may also be able to receive and process the Funds Available message, defined in section 8.1.4.

7.2 PIN Pad Handler

This section defines requirements for commands sent to the User Interface.

All Secure Cryptographic Device's supporting PKC shall support the commands Get Key Check Value, Get Public Key Record and Verify PSAM Public Key Certificate (Submit Initial key).

The terminal's Secure Cryptographic Device - PIN Pad or separate Secure Cryptographic Device - needs to support these commands.

7.2.1 Get Key Check Value

7.2.1.1 The Get Key Check Value command must conform to the format defined in Table 61.

Table 61: Get Key Check Value command

Field	Value	Length
Destination Address	'0301'	2
Source Address	Any	2
Message Type	'65'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0012' + N_{VKP}	2
RID_{PSAM}	RID used by the PSAM Creator	5
$ID_{PSAMCREATOR}$	Identifier assigned to the PSAM Creator by the owner of the RID	4
ID_{PSAM}	Identifier assigned by the PSAM Creator to the PSAM.	4
N_{VKP}	Number of CA PP Public Keys contained in the PSAM	1
$VKP_{CA,PP}$	Key versions of the CA PP Public Keys contained in the PSAM	N_{VKP}
$CHALLENGE_{PSAM}$	Any non-repeating or random 4-byte value generated by the PSAM	4

7.2.1.2 The Secure Cryptographic Device/PIN Pad must verify that one of the public key version numbers ($VKP_{CA, PP}$) listed in the Get Key Check Value Command (to be used by the PSAM to verify the certificates) corresponds to the version number of the public key that created the highest level certificate in a public key certificate chain available to the Secure Cryptographic Device/PIN Pad.

From the intersection of $VKP_{CA, PP}$s supported by both the PSAM and the SCD/PIN pad, the SCD/PIN pad shall select the version of the

$VKP_{CA, PP}$ having the lowest value of the $VKP_{CA, PP}$ key version.

If there is not match then an error response must be returned with the appropriate response code.

7.2.1.3 The Get Key Check Value response must conform to the format defined in Table 62.

Table 62: Response to Get Key Check Value command

Field	Value	Length
Destination Address	Any	2
Source Address	'0301'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0013' + N_{VKP}	2
$ID_{PPCREATOR}$	Identifier of the PIN Pad Creator	4
ID_{PP}	Identifier assigned to the SCD/PIN Pad by the PIN Pad Creator	4
N_{VKP}	Number of CA PSAM Public Keys contained in the SCD/PIN Pad	1
$VKP_{CA, PSAM}$	Key versions of the CA PSAM Public Keys contained in the PIN Pad	N_{VKP}
$VKP_{CA, PP}$	Key version of the CA PP Public Key that must be used to verify the PIN Pad Creator Certificate.	1
$CHALLENGE_{PP}$	Any non-repeating or random 4-byte value generated by the SCD/PIN Pad	4
KCV_{PP}	Key Check Value	3
Response Code	Response Code	2

7.2.1.4 The Response Codes applicable to the Get Key Check Value command are defined in Table 63.

7.2.1.5 To enable the synchronization process to continue if the Response code is 'FF80', the response to the Get Key Check Value command shall contain all data elements defined in Table 62.

Table 63: Response Codes to Get Key Check Value command

Response Code	Description
'FF80'	*No KCV available, KSES not present*
'FF90'	*RSA key mismatch. VKP not recognized*
'FFF3'	*Handler Error*
'FFF5'	*Handler busy*
'FFF6'	*Insufficient resources*
'FFF7'	*Handler must be opened*
'FFFB'	*Unsupported operation*

7.2.2 Get PIN Pad Public Key Record

7.2.2.1 The Get PIN Pad Public Key Record command must conform to the format defined in Table 64.

Table 64: Get PIN Pad Public Key Record command

Field	Value	Length
Destination Address	'0301'	2
Source Address	Any	2
Message Type	'67'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0001'	2
Format Code	'C2' = PKC_{PPC}, 'C4' = PKC_{PP}	1

7.2.2.2 The Get PIN Pad public Key Record response must conform to the format defined in Table 65.

Table 65: Response to Get PIN Pad Public Key Record command

Field	Value	Length
Destination Address	Any	2
Source Address	'0301'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002' + Length of $Record_{PKEY}$	2
$Record_{PKEY}$	Record containing tags, certificate (PKC_{PPC}/PKC_{PP}) and remainder (if present) See Table 66 and Table 67 for the formats of the PIN Pad Creator and PIN PAD public key records.	Var.
Response Code	Response Code	2

Table 66: Contents of PIN Pad Creator Certificate Record

Description	Data element	Mandatory or conditional	Length (bytes)
Record Tag	'85'	M	1
Data length	Sum of lengths of succeeding fields	M	1
Certification Authority Key version	VKP_{CA}	M	1
Length of CA Public Key Modulus	$LPKM_{CA, PP}$	M	1
PIN Pad Creator Public Key Certificate (enciphered)	PKC_{PPC}	M	$LPKM_{CA, PP}$
Rightmost bytes of the PIN Pad Creator Key Modulus	PKR_{PPC}	C	Maximum (0, $LPKM_{PPC}$ + 36 – $LPKM_{CA, PP}$)

Table 67: Contents of PIN Pad Certificate Record

Description	Data element	Mandatory or conditional	Length (bytes)
Record Tag	'85'	M	1
Data length	Sum of lengths of succeeding fields	M	1
Length of PIN Pad Creator Public Key Modulus	$LPKM_{PPC}$	M	1
PIN Pad Public Key Certificate (enciphered)	PKC_{PP}	M	$LPKM_{PPC}$
Rightmost bytes of the PIN Pad Key Modulus	PKR_{PP}	C	Maximum (0, $LPKM_{PP}$ + 40 – $LPKM_{PPC}$)

7.2.2.3 The Response Codes applicable to the Get PIN Pad Public Key Record command are defined in Table 68.

Table 68: Response Codes to Get PIN Pad Public Key Record command

Response Code	Description
'FF89'	*Record not found*
'FFF3'	*Handler Error*
'FFF4'	*Handler must be initialized*
'FFF5'	*Handler busy*
'FFF6'	*Insufficient resources*
'FFF7'	*Handler must be opened*
'FFFB'	*Unsupported operation*

7.2.3 Verify PSAM Public Key Certificate

7.2.3.1 The Verify PSAM Public Key Certificate

command must conform to the format defined in Table 69.

Table 69: Verify PSAM Public Key Certificate Command (PKC_{ACQ})

Field	Value	Length
Destination Address	'0301'	2
Source Address	Any	2
Message Type	'66'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	Variable	2
Format Code	'A2' = PKC_{ACQ} 'A4' = PKC_{PSAM}	1
$VKP_{CA, PSAM}$	Key version of the CA PSAM Public Key	1
LPKM	Length of the modulus of the key that signed the certificate: $LPKM_{CA, PSAM}$ or $LPKM_{ACQ}$ as appropriate.	1
$PKC_{ACQ/PSAM}$	Public Key Certificate being sent for verification	$LPKM_{CA, PSAM}$ or $LPKM_{ACQ}$
$PKR_{ACQ/PSAM}$	Rightmost bytes of Public Key Modulus being sent for verification For the acquirer certificate, the length is the maximum of 0 or ($LPKM_{ACQ}$ + 41 – $LPKM_{CA,PSAM}$.) For the PSAM certificate, the length is the maximum of 0 or ($LPKM_{PSAM}$ + 45 – $LPKM_{ACQ}$.)	May be 0

7.2.3.2 The Verify PSAM Public Key Certificate response must conform to the format defined in Table 70.

7.2.3.3 The PIN Pad must return the appropriate Response Code if the Verify PSAM Public Key Certificate command has not been processed correctly.

Table 70: Response to Verify PSAM Public Key Certificate command

Field	Value	Length
Destination Address	Any	2
Source Address	'0301'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	2
Response Code	Response Code	2

7.2.3.4 The Response Codes applicable to the Verify PSAM Public Key Certificate command are defined in Table 71.

Table 71: Response Codes to Verify PSAM Public Key Certificate command

Response Code	Description
'FF8C'	*Certificate Error*
'FF8D'	*Hash algorithm not supported*
'FF8E'	*PK Algorithm not supported*
'FF8F'	*Hash result invalid*
'FF90'	*RSA key mismatch. VKP not recognized*
'FF91'	*Certificate format error*
'FF92'	*Certificate expired*
'FF93'	*Certificate ID mismatch*
'FFF3'	*Handler Error*
'FFF4'	*Handler must be initialized*
'FFF5'	*Handler busy*

Response Code	Description
'FFF6'	*Insufficient resources*
'FFF7'	*Handler must be opened*
'FFFB'	*Unsupported operation*

7.2.4 Submit Initial Key

7.2.4.1 The Submit Initial Key command must conform to the format defined in Table 72.

7.2.4.2 In order to generate the PS signature, the PSAM must perform the following steps.

1. Compute the "Hash Result" by using the SHA-1 algorithm on the data defined in Table 76.

   Note that the PIN Pad data (PIN Pad Identification and $CHALLENGE_{PP}$) are retained from the response to the Get Key Check Value at the beginning of the synchronization sequence.

2. Generate a digital signature "DS" on the data shown in Table 75, using the PSAM private key.

3. Split the digital signature into two components: a 96-byte DS_1 and a remainder DS_{REM}.

   $$DS = DS_1 \,||\, DS_{REM}$$

4. Generate the DS_2 by padding the DS_{REM} with sufficient bytes of binary zeros to create a 96-byte string.

   $$DS_2 = DS_{REM} \,||\, '00\ldots00'$$

5. Apply the Padding function defined in section 14.7.8 to each of the two DS components, using $L=LPKM_{PP}$.

$PDS_i = PAD(DS_i)$

6. Encrypt the results using the PIN Pad's public key, to generate the two signatures, PS_1 and PS_2.

$PS_i := RSAencipher(PK_{PP})[PDS_i]$

7. The result ($PS = PS_1 || PS_2$) is sent to the PIN Pad in the Submit Initial Key command.

Table 72: Submit Initial Key command

Field	Value	Length
Destination Address	'0301'	2
Source Address	'00pp' where pp is the sub-address assigned to the PSAM	2
Message Type	'68'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'000D' + 2*$LPKM_{PP}$ (length of the modulus of the PIN Pad Public Key)	2
RID_{PSAM}	RID used by the PSAM Creator	5
$ID_{PSAMCREATOR}$	Unique identifier of the PSAM Creator	4
ID_{PSAM}	Unique identifier of the PSAM	4
PS	$PS_1 \| \| PS_2$, Enciphered digital signature of the PSAM	2*$LPKM_{PP}$

7.2.4.3 The Submit Initial Key response must conform to the format defined in Table 73.

7.2.4.4 In order to decrypt and verify the encrypted digital signature (PS) and recover the Initial Session Key ($KSES_{INIT}$), the PIN Pad must perform the following steps.

1. Recover the padded digital signatures PDS_1 and PDS_2 by decrypting each of the PS components, using its own private key.

PDS_i := RSAdecipher(SK_{PP})[PS_i]

2. Recover each value DS_i from PDS_i as defined in Section 14.7.8.

3. Reconstruct the signature DS by concatenating DS_1 and DS_2, and taking the first $LPKM_{PSAM}$ bytes.

   DS := DS_1 || DS_{REM} = $LPKM_{PSAM}$ bytes (DS_1 || DS_2)

4. Recover the signed data in Table 75 using the PSAM's public key to verify the DS.

5. Validate the signed data to ensure that the header, format code and trailer all contain valid data as specified in Table 75.

6. Construct the DSHash as specified in Table 76, and perform the SHA-1 algorithm. The signature is verified by comparing the result to the Hash Result in the signed data previously recovered.

   HashResult:= SHA-1(DSHash).

   Note that the data from the PSAM (PSAM Identification and $CHALLENGE_{PSAM}$) are retained from the Get Key Check Value command received at the beginning of the synchronization sequence.

7. If all the above checks are successful then $KSES_{INIT}$ is accepted and synchronization is complete.

Table 73: Response to Submit Initial Key command

Field	Value	Length
Destination Address	'00pp' where pp is the sub-address assigned to the PSAM	2
Source Address	'0301'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0005'	2
KCV_{PP}	Key Check Value derived as specified in Table 79 using the $KSES_{INIT}$.	3
Response Code	Response Code	2

7.2.4.5 The Response Codes applicable to the Submit Initial Key command are defined in Table 74.

Table 74: Response Codes to Submit Initial Key command

Response Code	Description
'FF83'	*PSAM Identifier not recognized*
'FF8A'	*Signature Error*
'FF8B'	*Hash Error*
'FFF3'	*Handler Error*
'FFF4'	*Handler must be initialized*
'FFF5'	*Handler busy*
'FFF6'	*Insufficient resources*
'FFF7'	*Handler must be opened*
'FFFB'	*Unsupported operation*

Table 75: Format of Data Recovered from DS

Field	Content/Source	Length (bytes)
Header	'6A'	1
Format code	'89'	1
ALGH	Code for the algorithm used to produce the hash ('01' for SHA-1)	1
$KSES_{INIT}$	Initial Session Key produced by PSAM	16
Pad Pattern	Successive bytes containing 'BB'	$LPKM_{PSAM}$ – 40
Hash Result	Hash of signed data, see Table 76	20
Trailer	'BC'	1

Table 76: Contents of the DS Hash

Field	Content/Source	Length (bytes)
Format code	'89'	1
ALGH	Code for the algorithm used to produce the hash ('01' for SHA-1)	1
$KSES_{INIT}$	Initial Session Key produced by PSAM	16
Pad Pattern	Successive bytes containing 'BB'	$LPKM_{PSAM}$ – 40
PIN Pad Identification		
$ID_{PPCREATOR}$	Identifies the PP Creator	4
ID_{PP}	Identifies the SCD/PIN Pad	4
$CHALLENGE_{PP}$	Challenge from SCD/PIN Pad	4

Field	Content/Source	Length (bytes)
PSAM Identification		
RID_{PSAM}	RID used by the PSAM Creator	5
$ID_{PSAMCREATOR}$	Identifier of the PSAM Creator	4
ID_{PSAM}	Identifier of the PSAM	4
$CHALLENGE_{PSAM}$	Challenge from the PSAM	4

7.2.5 Initiate PIN Entry

7.2.5.1 The Initiate PIN Entry command must conform to the format defined in Table 77.

Table 77: Initiate PIN Entry command

Field	Value	Length
Destination Address	'0301'	2
Source Address	Any	2
Message Type	'69'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0023'	2
PIN Pad Identification		
$ID_{PPCREATOR}$	Unique identifier of the PIN Pad Creator	4
ID_{PP}	Unique identifier of the PIN Pad	4
PSAM Identification		
RID_{PSAM}	RID used by the PSAM Creator	5
$ID_{PSAMCREATOR}$	Unique identifier of the PSAM Creator	4
ID_{PSAM}	Unique identifier of the PSAM	4
KCV_{PSAM}	Current Key Check Value calculated by the PSAM	3
Min PIN Digits	Minimum number of PIN digits ('04' – '0C')	1

Field	Value	Length
Max PIN Digits	Maximum number of PIN digits ('04' – '0C')	1
Number of PIN entries left	'x0' – 'xE' and 'xF'. The high-order nibble (as indicated by the 'x') is reserved for proprietary coding. The low-order nibble indicates the number of PIN entry attempts that remain. An 'F' in the low-order nibble indicates that this information shall not be displayed.	1
MAC_{IPE}	MAC on the preceding data elements (Destination Address – Number of PIN Entries left) computed using $KSES_{MAC}$	8

7.2.5.2 The Initiate PIN Entry response must conform to the format defined in Table 78.

7.2.5.3 Prior to generating or verifying the MAC_{IPE} in the Initiate PIN Entry command, the PSAM and the Secure Cryptographic Device must each derive a new set of PIN session keys from the previous set. A new Key Check Value (KCV) for the Transaction Session Key (KSES) must also be calculated. The algorithms for deriving the new session keys, and for calculating the new check value, are specified in Table 79. As part of this derivation, each key (i.e. $KSES_{NEW}$, $KSES_{PIN,}$ $KSES_{DATA,}$ $KSES_{CDP}$ and $KSES_{MAC}$) must have its bytes adjusted for odd parity.

Note that the Initial Session Key ($KSES_{INIT}$) established during synchronization is only used to derive the first set of PIN Session Keys, and the KCV_{PP} returned in the response to the Submit Initial Key command.

*Note: Each byte in a key with odd parity must have an odd number of one-bits. Parity is adjusted by changing the low order (rightmost) bit. An example of a key **without** odd parity is '11 22 33 44 55 66 77 88'. The same key adjusted for odd parity is '10 23 32 45 54 67 76 89'.*

Table 78: Response to Initiate PIN Entry command

Field	Value	Length
Destination Address	Any	2
Source Address	'0301'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'000A'	2
MAC_{IPER}	MAC on the MAC_{IPE} from the command, computed using $KSES_{MAC}$	8
Response Code	Response Code	2

Table 79: SCD Session Key Derivation

Key	Value	Length
$KSES_{NEW}$	$DES3(KSES_{OLD})[D_L] \mid\mid DES3(KSES_{OLD})[D_R]$	16
D_L	ID_{PP} || 'F0 00 00 00'	8
D_R	ID_{PP} || '0F 00 00 00'	8
KCV_{NEW}	3MSB{ $DES3(KSES_{NEW})$['00 00 00 00 00 00 00 00']}	3
$KSES_{PIN}$	$DES3(KSES_{NEW})[DP_L] \mid\mid DES3(KSES_{NEW})[DP_R]$	16
DP_L	'FF FF 00 00 00 00 00 00'	16
DP_R	'F0 F0 00 00 00 00 00 00'	16
$KSES_{MAC}$	$DES3(KSES_{NEW})[DM_L] \mid\mid DES3(KSES_{NEW})[DM_R]$	16
DM_L	'F0 0F 00 00 00 00 00 00'	16
DM_R	'0F F0 00 00 00 00 00 00'	16
$KSES_{DATA}$	$DES3(KSES_{NEW})[DD_L] \mid\mid DES3(KSES_{NEW})[DD_R]$	16
DD_L	'F0 F0 00 00 00 00 00 00'	16
DD_R	'0F 0F 00 00 00 00 00 00'	16

Table 80: CDP Key Derivation

Key	Value	Length
$KSES_{CDP}$	$DES3(KSES_{INI})[D_{LCDP}]$ || $DES3(KSES_{INI})[D_{RCDP}]$	16
D_{LCDP}	ID_{PP} ||'00 F0 00 00'	8
D_{RCDP}	ID_{PP} || '00 0F 00 00'	8

7.2.5.4 The Response Codes applicable to the Initiate PIN Entry command are defined in Table 81.

Table 81: Response Codes to Initiate PIN Entry command

Response Code	Description
'FF81'	*Wrong PIN Pad ID*
'FF82'	*Authentication Error (MAC validation failed)*
'FF83'	*PSAM Identifier not recognized*
'FF84'	*Parameters out of range*
'FF85'	*Key Check values not identical, synchronization necessary*
'FFF3'	*Handler Error*
'FFF4'	*Handler must be initialized*
'FFF5'	*Handler busy*
'FFF6'	*Insufficient resources*
'FFF7'	*Handler must be opened*
'FFFB'	*Unsupported operation*

7.2.6 Get PIN

7.2.6.1 The Get PIN command must conform to the format defined in Table 82.

Table 82: Get PIN command

Field	Value	Length
Destination Address	'0301'	2
Source Address	Any	2
Message Type	'6A'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'000D'	2
Timer Flag	'00' = Not Timed '80' = Timed	1
Time	Time-out value in milliseconds	4
MAC_{GP}	MAC on the preceding data elements (Dest. address – Time) computed using $KSES_{MAC}$	8

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

Figure 7: PIN Block Format

Table 83: Definition of PIN block format

	Name	Value
C	Control field	4-bit binary control field. Shall be ('2')
N	PIN length	4-bit binary number with permissible values of '4' - 'C'
P	PIN digit	4-bit binary number with permissible values of '0' - '9'
P/F	PIN/Filler	Determined by PIN length
F	Filler	4-bit binary number with value 'F'

7.2.6.3 The Get PIN response must conform with the format defined in Table 84.

Table 84: Response to Get PIN command

Field	Value	Length
Destination Address	Any	2
Source Address	'0301'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'001A'	2
Enc($KSES_{DATA}$)[Data]	Data consisting of the following	
RND(8)	Random Data	8
Enc($KSES_{PIN}$)[PIN]]	PIN block encrypted under $KSES_{PIN}$ transferred concatenated with random number in envelope encrypted under $KSES_{DATA}$. See Figure 8 and Table 83 for the format of the PIN Block. No padding is included.	8
MAC_{RGP}	MAC on ENC($KSES_{PIN}$) || MAC_{GP} (from the command), computed using $KSES_{MAC}$	8
Response Code	Response Code	2

7.2.6.4 The PIN Pad must be capable of generating the Response Codes to the Get PIN command as defined in Table 85.

7.2.6.5 The plaintext PIN block format to be enciphered must be formatted as shown in Figure 8 and as specified in reference 6, EMV (section 2.4.12).

7.2.6.6 The Response Codes applicable to the Get PIN command are defined in Table 85.

Table 85: Response Codes to Get PIN command

Response Code	Description
'FF82'	*Authentication Error (MAC validation failed)*
'FF86'	*PIN not available*
'FF87'	*Secure Cryptographic Device not in PIN Entry State*
'FFF2'	*Time-out*
'FFF3'	*Handler Error*
'FFF4'	*Handler must be initialized*
'FFF5'	*Handler busy*
'FFF6'	*Insufficient resources*
'FFF7'	*Handler must be opened*
'FFFB'	*Unsupported operation*

7.2.7 Terminate PIN Entry

7.2.7.1 The Terminate PIN Entry command must conform to the format defined in Table 86.

Table 86: Terminate PIN Entry command

Field	Value	Length
Destination Address	'0301'	2
Source Address	Any	2
Message Type	'6C'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0015'	2
PIN Pad Identification		
$ID_{PPCREATOR}$	Unique identifier of the PIN Pad Creator	4

Field	Value	Length
ID_{PP}	Unique identifier of the PIN Pad	4
PSAM Identification		
RID_{PSAM}	RID used by the PSAM Creator	5
$ID_{PSAMCREATOR}$	Unique identifier of the PSAM Creator	4
ID_{PSAM}	Unique identifier of the PSAM	4

The Terminate PIN Entry response must conform to the format defined in

7.2.7.2 Table 87.

Table 87: Response to Terminate PIN Entry command

Field	Value	Length
Destination Address	Any	2
Source Address	'0301'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	2
Response Code	Response Code	2

7.2.7.3 The Response Codes applicable to the Terminate PIN Entry command are defined in Table 88.

Table 88: Response Codes to Terminate PIN Entry command

Response Code	Description
'FF81'	*Wrong PIN Pad ID*
'FF83'	*PSAM Identifier not recognized*
'FF87'	*Secure Cryptographic Device not in PIN Entry State*
'FF88'	*Termination Failed*
'FFF3'	*Handler Error*
'FFF4'	*Handler must be initialized*
'FFF5'	*Handler busy*
'FFF6'	*Insufficient resources*
'FFF7'	*Handler must be opened*
'FFFB'	*Unsupported operation*

7.3 Summary

Table 89: User Interface-Specific commands

Destination Address	Source Address	Message Type	Description
'0300'	Any	'60'	Confirm Amount
'0304'	Any	'61'	Display a predefined message
'0302'	Any	'63'	Print a predefined message
'0302'	Any	'64'	Purge print buffer
'0300'	Any	'80'	Get Amount
'0300'	Any	'82'	Funds Available
'0301'	Any	'65'	Get Key Check Value
'0301'	Any	'67'	Get PIN Pad Public Key Record

Destination Address	**Source Address**	**Message Type**	**Description**
'0301'	Any	'66'	Verify PSAM Public Key Certificate
'0301'	Any	'68'	Submit Initial Key
'0301'	Any	'69'	Initiate PIN Entry
'0301'	Any	'6A'	Get PIN
'0301'	Any	'6C'	Terminate PIN Entry

8. The Merchant Application Handler

The Merchant Application Handler is responsible for managing the interface to all merchant-related equipment and peripherals, which may include the merchant display, printer, or merchant keypad.

In addition to the common Handler commands provided in Section 4.4, the Merchant Application Handler must support the command set documented in this section. Additional implementation specific functions may be performed by the Merchant Application Handler, but are outside the scope of this specification.

8.1 Messages sent to the Merchant Application Handler

This section provides a list of additional commands that should be accepted and processed by the Merchant Application Handler.

The Get Amount commands consist of the basic Get Amount command and the Get Amount Enhanced command in which additional transaction specific data may be exchanged using the Discretionary Data field. The definition of the Discretionary Data may be different for command and response. Which version of the Get Amount command to use is application specific.

8.1.1 Get Amount

8.1.1.1 The Get Amount command must conform to the format defined in Table 90.

Table 90: Get Amount command

Field	Value	Length
Destination Address	‘0300’ or '0400'	2
Source Address	Any	2
Message Type	'80'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0009' or ‘0011'	2

Field	Value	Length
Timer Flag	'00' = Not Timed '80' = Timed	1
Time	Time-out value in milliseconds	4
Display Message Code	Code indicating the message to be displayed (see Table 177) '00' indicates that no message is to be displayed.	1
CURR	Currency Code and exponent	3
SP_{MAC}	MAC on Destination Address – CURR, computed using $KSES_{MAC}$.	0 or 8

8.1.1.2 The Get Amount response must conform to the format defined in Table 91.

8.1.1.3 If the currency code and exponent in the command were zeros, then the Merchant Application Handler must return the currency of the amount in the response.

8.1.1.4 If the merchant application must display a message to the merchant or the user for amount entry, the Display Message Code indicates the message to be displayed.

8.1.1.5 If the Merchant Application does not use a display to request an amount entry, and the command issued contained a Display Message Code, but the amount was still successfully entered, the Response Code 'successfully processed' must only be returned in the case where the merchant application automatically replies to the command (for example, in a vending machine).

8.1.1.6 If a display is used in the Get Amount process and the Merchant Application Handler does not recognize the Display Message Code, a Response Code 'FF34' must be returned. In this case the amount returned, if any, is not reliable.

Table 91: Response to Get Amount command

Field	Value	Length
Destination Address	Any	2
Source Address	'0400'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	’0009’	2
Transaction Amount	Transaction Amount	4
CURR	Currency Code and exponent	3
Response Code	Response Code	2

8.1.1.7 The Response Codes applicable to the Get Amount command are defined in Table 92.

Table 92: Response Codes to Get Amount command

Response Code	Description
‘FF34’	Unknown Message Code
‘FF40’	Invalid Currency
‘FF41’	Invalid Currency Exponent
'FFF2'	*Time-out*: the requested operation is valid, but some external event necessary for the proper execution failed to arrive in time; or merchant or cardholder requests a cancellation.
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.

Response Code	Description
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

8.1.2 Get Amount Enhanced

8.1.2.1 The Get Amount Enhanced command must conform to the format defined in Table 93.

Table 93: Get Amount Enhanced command

Field	Value	Length
Destination Address	'0300' or '0400'	2
Source Address	Any	2
Message Type	'80'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'000A' or '0012' + Length of Discretionary Data	2
Timer Flag	'00' = Not Timed '80' = Timed	1
Time	Time-out value in milliseconds	4
Display Message Code	Code indicating the message to be displayed (see Table 177) '00' indicates that no message is to be displayed.	1
CURR	Currency Code and exponent	3
LEN_{DD}	Length of Discretionary Data	2
Discretionary Data	Discretionary Data	variable
SP_{MAC}	MAC on preceding data elements [Destination Address – Discretionary Data], computed using $KSES_{MAC}$.	0 or 8

8.1.2.2 If the Destination Address is '0300', the MAC

must be included.

8.1.2.3 The requirements for the Get Amount command cover the Get Amount Enhanced command, too.

8.1.2.4 The Get Amount Enhanced response must conform to the format defined in Table 94.

Table 94: Response to Get Amount Enhanced command

Field	**Value**	**Length**
Destination Address	Any	2
Source Address	'0300' or '0400'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'000B' or '0013' + Length of Discretionary Data	2
Transaction Amount	Transaction Amount	4
CURR	Currency Code and exponent	3
LEN_{DD}	Length of Discretionary Data E.g. Amount Other.	2
Discretionary Data	Discretionary Data	variable
SP_{MAC}	If Source Address is '0300', MAC included. MAC on preceding data elements [Transaction Amount – Discretionary Data] || SP_{MAC} (from the command), computed using $KSES_{MAC}$.	0 or 8
Response Code	Response Code	2

8.1.2.5 The Response Codes applicable to the Get Amount Enhanced command are defined in Table 95.

Table 95: Response Codes to Get Amount Enhanced command

Response Code	Description
'FF34'	Unknown Message Code
'FF40'	Invalid Currency
'FF41'	Invalid Currency Exponent
'FFF2'	*Time-out*: the requested operation is valid, but some external event necessary for the proper execution failed to arrive in time.
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

8.1.3 Transaction Completed

The Transaction Completed command is issued to the Merchant Application Handler to inform it of the completion status of a specified transaction.

8.1.3.1 The Transaction Completed command must conform to the format defined in Table 96.

Table 96: Transaction Completed command

Field	Value	Length
Destination Address	'0400'	2
Source Address	Any	2
Message Type	'81'	1

Field	Value	Length
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0001'	2
Transaction Results	'00' = Transaction Successful '01' = Transaction Failed All other values are reserved for future use.	1

8.1.3.2 The Transaction Completed response must conform to the format defined in Table 97.

Table 97: Response to Transaction Completed command

Field	Value	Length
Destination Address	Any	2
Source Address	'0400'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	2
Response Code	Response Code	2

8.1.3.3 The Response Codes applicable to the Transaction Completed command are defined in Table 98.

Table 98: Response Codes to Transaction Completed command

Response Code	Description
'FF42'	Invalid Transaction Results value
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later.

Response Code	Description
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

8.1.4 Funds Available

The Funds Available command may be used to inform the Merchant Application of the funds available to make a purchase.

8.1.4.1 The Funds Available command must conform to the format defined in Table 99.

Table 99: Funds Available command

Field	Value	Length
Destination Address	'0400'	2
Source Address	Any	2
Message Type	'82'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0008'	2
Amount	Amount of funds available	4
CURR	Currency Code and exponent	3

8.1.4.2 The Funds Available response must conform to the format defined in Table 100.

Table 100: Response to Funds Available command

Field	Value	Length
Destination Address	Any	2
Source Address	'0400'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	’0002’	2
Response Code	Response Code	2

8.1.4.3 The Response Codes applicable to the Funds Available command are defined in Table 101.

Table 101: Response Codes to Funds Available command

Response Code	Description
‘FF40’	Invalid Currency
‘FF41’	Invalid Currency Exponent
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.

8.1.5 Display Message

The Display handler must be able to receive and process the Display Message commands, which are defined in section 7.1.1.

8.1.6 Print commands

The Printer handler must be able to receive and process the

Print Message and Purge Print Buffer commands, which are defined in sections 7.1.2 and 7.1.4.

8.2 Summary

Table 102: Merchant Application Handler-Specific commands

Destination Address	Source Address	Message Type	Description
'0404'	Any	'61'	Display a predefined message
'0402'	Any	'63'	Print a predefined message
'0402'	Any	'64'	Purge print buffer
'0400'	Any	'80'	Get Amount
‘0400’	Any	‘81’	Transaction Completed
‘0400’	Any	‘82’	Funds Available

9. The PSAM Handler

The PSAM Handler is responsible for managing the interface to any number of PSAMs that may be resident in the terminal.

The interface between the PSAM Handler and the PSAM is a standard command/response protocol as defined in reference 3, ISO/IEC 7816-4 and reference 6, EMV Part II, section 2.1. The PSAM is usually implemented as a processor card, where the technical interface is either T=0 or T=1, as defined in reference 2, ISO/IEC 7816-3 and reference 6, EMV Part I. However, it is possible to implement a PSAM in an alternative manner, such as in a Hardware Security Module (HSM) to support a server or other high-volume system.

The PSAM may perform any combination of the following services:

- Be the storage medium where application specific code is stored and executed to perform one or more business functions such as CEP, EMV or any other application, and hence effectively run the application.
- Provide specialized cryptography services for one or more payment applications.
- Provide generic (e.g. ISO/IEC DIS 7816-8) cryptography services.

9.1 Message Handling

The PSAM Handler interface to the PSAM is more complex than that between the Card Handler and the processor card. While the PSAM does use the standard command/response protocol, it can have broader functionality than a normal “smart card” as used for a consumer card application.

The two principal features of a PSAM, which the PSAM Handler must cater to, are:

1. The MAD Handler may delegate control over the terminal processing to the PSAM. In this case, the PSAM must be able to send commands to other terminal devices and receive their responses. These commands from the PSAM are known as “derived commands”.
2. The PSAM may be “multi-threaded”, handling several concurrent transactions (each with a different ID_{THREAD}), each in a different state of completion.

9.1.1 Messages sent to the PSAM Handler

The PSAM Handler must be able to process all of the ICC-related commands supported by the Processor Card Reader that are listed in Table 103.

The handler for each individual PSAM must be able to process all the commands supported by the Processor Card Reader. However, the treatment of the command and its response might be different.

The next section describes how a Terminal Message which conveys an ICC command (Message Type = '42'), or a response from another device (Message Type = 'FF'), is transformed to a Command APDU for the PSAM as defined in reference 3, ISO/IEC 7816-4.

Table 103: ICC Commands supported by PSAM Handler

Message Type	Description
'42'	ICC Command
'43'	ICC Power-On
'44'	ICC Power-Off
'45'	ICC Query

9.1.2 Messages sent to the PSAM

The PSAM Handler will receive messages that are intended for delivery to a PSAM (as opposed to the PSAM Handler itself), in one of the following message structures:

- PSAM Command: Message Type '42' indicates A PSAM command. In this case, the Message Data field contains a complete C-APDU that must be forwarded to the PSAM.
- Response Message: Message Type 'FF' indicates a response message from another terminal device. These are received if the PSAM has previously originated a derived command to the responding device.

Figure 9 illustrates the message translation that is performed by the PSAM Handler for commands sent to the PSAM.

9.1.2.1 If present, the L_c must be coded on one byte. The L_e must always be present and be coded on one byte with the value '00'.

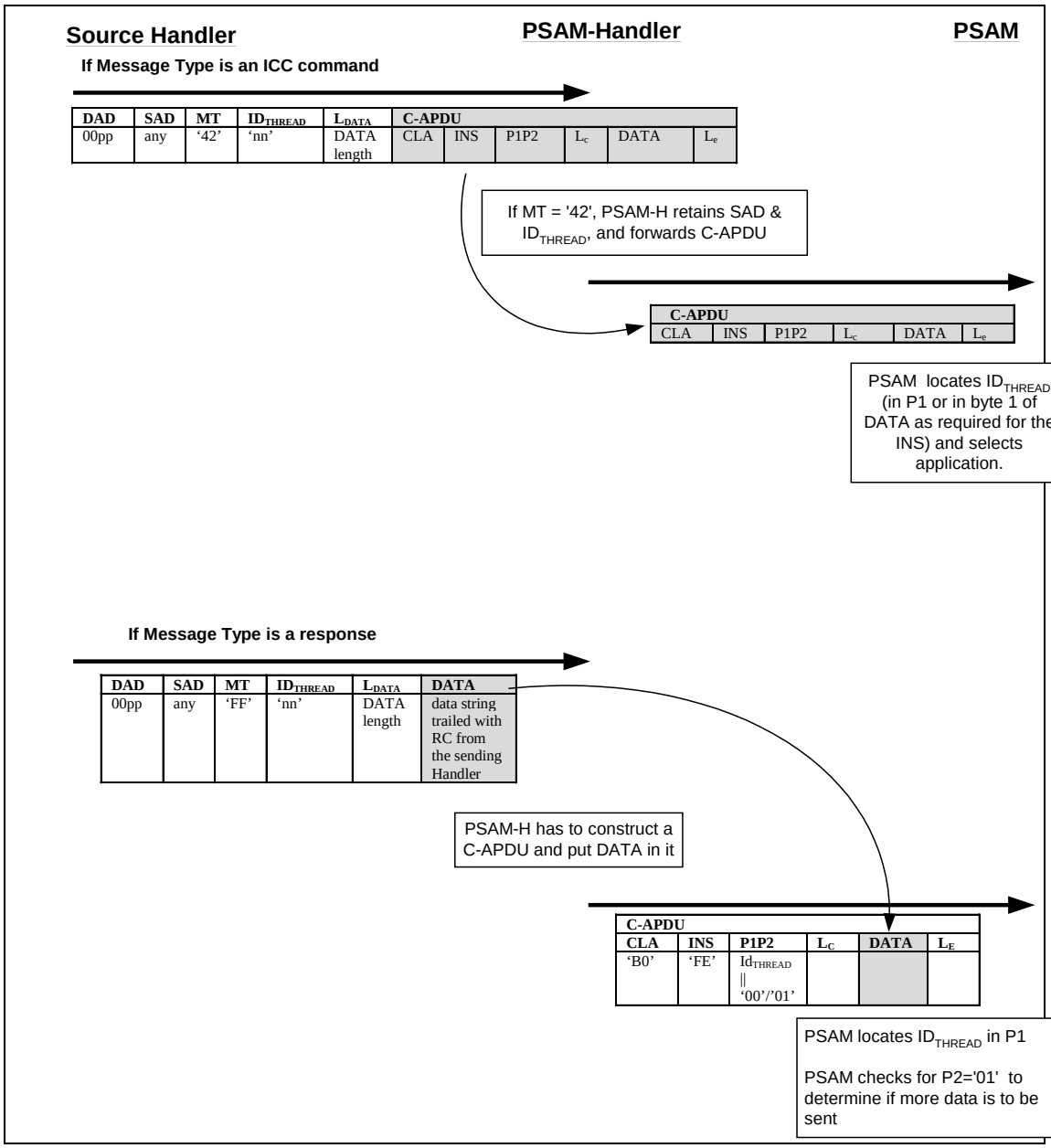

Figure 9: Message Translation for commands to PSAM

9.1.2.2 When the PSAM Handler receives an ICC command (Message Type '42') it must forward the C-APDU contained within the message to the PSAM. The Source Address and ID_{THREAD} in the message must be retained in order to route correctly the subsequent response from the PSAM.

9.1.2.3 When the PSAM Handler receives a response

message (Message Type 'FF'), it must construct a Response Command APDU as shown in Table 118 and send this command to the PSAM.

9.1.2.4 If the L_{DATA} field in a Response Message exceeds 248 bytes, the PSAM Handler must deliver the response in multiple response commands. In such a response command, the PSAM Handler must set the value of P2 equal to '01'. If P2 equals '01', then the L_c of the response command must be 248[1].

9.1.2.5 The PSAM Handler must continue sending response commands with P2 = 01 until the remainder of the data to be sent does not exceed 248 bytes. The final response command of the series must use P2 = 00.

9.1.2.6 If the PSAM Handler receives a command for a PSAM (Message Type '42') and the C-APDU cannot be successfully forwarded to the PSAM, the PSAM Handler must reply to the originator of the command with the appropriate Response Code.

9.1.3 Messages from the PSAM

The PSAM will output all messages in the form of Response APDUs. For derived commands being sent to other terminal devices, the data portion of the Response APDU will be in the Terminal Message format, ready to be delivered to the addressed handler. For response messages, the PSAM Handler must insert the correct destination address, and a Response Code, prior to forwarding the message.

Figure 10 illustrates the message translation performed by the PSAM handler when Response APDUs are received from the PSAM.

[1] Note that this applies only to response messages (MT = 'FF'). Some application designs will require that data be sent to the PSAM application in commands, which exceed the amount that can be accommodated in a single C-APDU. In this case, the application design must provide the ability to send the information in multiple command APDU's.

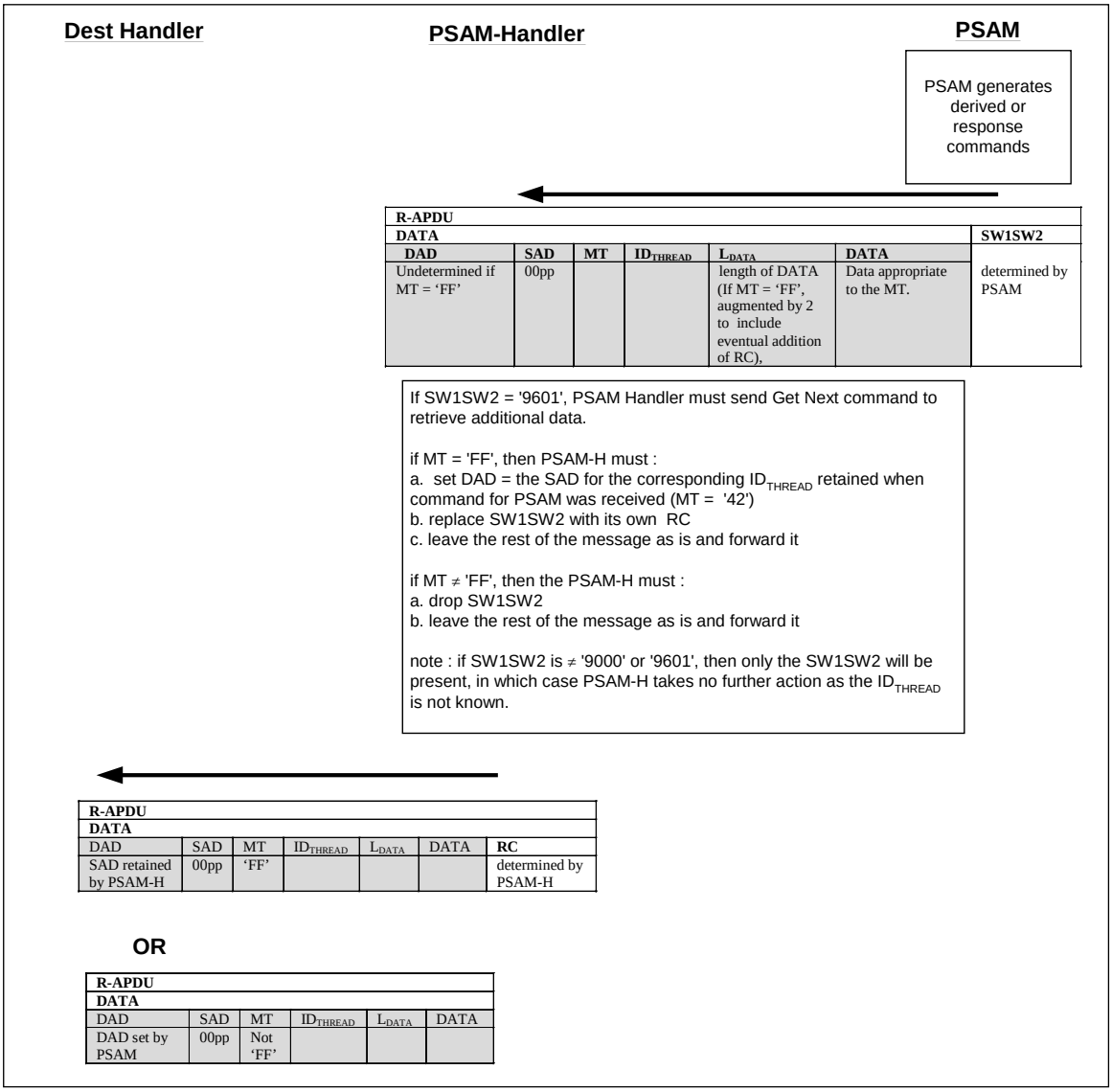

Figure 10: Message Translation for response from PSAM

9.1.3.1 The PSAM must send all derived commands in the form of a Response APDU. The data portion must be in Terminal Message format, ready to be forwarded to the recipient. The source address must specify the sub-address assigned to the PSAM, the destination address must be the intended recipient, and the ID_{THREAD} must be that assigned by the MAD Handler.

9.1.3.2 The PSAM must send all response messages in the form of a Response APDU. The data portion must be in the Terminal Message format, but without the Response Code. The source address must specify the sub-address assigned to the PSAM, and the ID_{THREAD} must be that assigned

by the MAD Handler.

9.1.3.3 If the amount of data to send is greater than 252 bytes, the PSAM must deliver the data in multiple response APDUs. All but the last one have SW1SW1 = '9601', indicating more data is to come. The last response APDU must have status bytes SW1SW2 = '9000', indicating all data is sent successfully.

9.1.3.4 On receipt of an SW1SW2 = '9601', the PSAM Handler must send a "Get Next" command, requesting further data. The Get Next command is detailed in Section 10.3.6.

9.1.3.5 The PSAM Handler must concatenate the series of responses until all data is received or the Get Next command is rejected.

9.1.3.6 When the PSAM Handler has received the complete response from the PSAM, the PSAM handler must forward the message to the assigned destination address.

If the Message Type is different from 'FF', the PSAM Handler passes the message unaltered to the router.

If the Message Type = 'FF', then prior to forwarding the message, the following modifications must be made:

- the PSAM Handler must set the destination address to the source address saved from the last PSAM command (Message Type = '42') received for the specified ID_{THREAD};
- the PSAM Handler must insert the two byte Response Code = '0000'.

9.1.3.7 If the response from the PSAM does not contain a valid Terminal Message (that is, the associated Thread cannot be determined, and the destination is either not specified or cannot be derived) the PSAM Handler must not forward the message.

Table 104: Response Codes applicable to PSAM Handler

Response Code	Description
'FF23'	Card did not respond
'FF24'	No card in reader
'FF25'	Unrecoverable Transmission error
'FF26'	Card buffer overflow
'FF27'	Unrecoverable Protocol error
'FF28'	Response has no status words
'FF29'	Invalid buffer
'FF2A'	Other card error
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

10. PSAM Applications

A PSAM receives incoming commands, and invokes the correct application for processing based on the contents of the command.

The specific requirements for individual PSAM applications are defined in application-specific specifications. This section defines only the requirements for generic functionality, common to all applications.

10.1 PSAM Initialization

After a terminal resets a PSAM, the terminal applications must each initialize the corresponding applications in the PSAM(s) by sending the Start-up PSAM command. This command allows the terminal to provide the PSAM with its current assigned “sub-address”, which must be included in subsequent messages originated by the PSAM application.

The response to the command contains the PSAM identification ($RID_{PSAM}||ID_{PSAMCREATOR}||ID_{PSAM}$) as a first field and other application specific data following that.

Following the start-up command, each terminal application may obtain a list of all AIDs, which can be processed by that specific PSAM application.

Using the list of AIDs supported by the different PSAM applications, the MAD-Handler can determine the set of AIDs mutually supported by the terminal and PSAM, and which PSAM application is to be used for each card AID.

The MAD-Handler applications must each perform any start-up procedures required by the associated specifications. Those start-up procedures will be defined in the application specifications. These application-specific requirements are outside the scope of this document.

This section provides an overview of the PSAM initialization sequence. The overall initialization process is described in section 5.2.

10.1.1.1 On reset, the PSAM will respond with the ATR, including the Historical Bytes, if any.

10.1.1.2 In the response to the PSAM Startup command, the PSAM must include the PSAM Identification ($RID + ID_{PSAMCREATOR} + ID_{PSAM}$) and may include additional application specific data.

10.1.1.3 The PSAM will respond to the Get Supported AIDs command with the list of AIDs supported by that application.

10.2 PSAM Shut-down

The Shutdown Command allows the PSAM application to save all outstanding data, prior to withdrawal power from the PSAM.

10.2.1.1 The PSAM must send a successful response, even if the particular PSAM implementation does not require any processing as a result of receiving this command.

10.3 PSAM Commands and Responses

This specification defines the use of commands with CLA byte 'B0'. The INS ranges and their usage are defined in Table 105. Table 106 lists the application-independent commands that must be supported by the PSAM Manager.

Table 105: CLA/INS Byte Definitions

CLA	INS	P1-P2	Command
B0	'00'-'2E'	$ID_{PSAMAPP}$	Generic commands supported by all applications. Commands cannot be concurrent within the same PSAM application. ID_{THREAD} must be in most significant byte of the command data These commands are defined in this specification.
B0	'30'-'5E' '70'-'7E'	$ID_{PSAMAPP}$	Application-specific, non-concurrent commands ID_{THREAD} must be in most significant byte of the command data
B0	'80'-'8E' 'A0'-BE'	$ID_{PSAMAPP}$	Application-specific commands for a particular thread. ID_{THREAD} must be in most significant byte of the command data.
B0	'C0'-'DE'	$ID_{PSAMAPP}$	Generic commands supported by all applications, for a particular thread. These commands are defined in this specification. ID_{THREAD} must be in most significant byte of the command data.
B0	'E0'-'FE'	ID_{THREAD} || '00'/'01'	Generic commands supported by all applications, for a particular thread. These commands are defined in this specification.

Table 106: Application-Independent PSAM Commands

CLA INS	Command	Description
'B0 02'	Start-up PSAM	Used to exchange identification information with the PSAM application
'B0 04'	PSAM Shutdown	Informs the PSAM application that power will be withdrawn and allows it to prepare for subsequent re-start.
'B0 08'	Get Supported AIDs	Obtains the set of AIDs supported by the PSAM application.
'B0 C2'	Synchronize PSAM/PIN Pad	Instructs the PSAM application to synchronize with the PIN Pad/SCD.
'B0 FC'	Get Next	Used to obtain the next incremental response from the PSAM.
'B0 FE'	Response Command	Used to convey to the PSAM responses received from other terminal devices.

10.3.1 Message Formats

All commands must be delivered to the PSAM in the form of Command APDUs.

Commands from the terminal application (in the MAD-Handler) must be sent to the PSAM Handler in an ICC command Terminal Message (Message Type '42'). The PSAM sends all responses to these commands in the form of Response Messages (Message Type 'FF') embedded within Response APDUs.

The PSAM Handler itself generates two types of commands to the PSAM.

- The Response Command is used to forward to the PSAM response data received from another terminal device
- The Get Next Command is used to retrieve continuation data when the PSAM is sending more data than can be fitted into a single R-APDU.

Sections 9.1.2 and 9.1.3 specify the PSAM-Handler requirements for handling these messages.

A successful response to the MAD-Handler will be in the "nominal" formats shown for each defined command. The general format for a successful response is shown in Table 107. An error response will be as defined in Table 108.

Note that there are two types of error response that the PSAM

may send:

- If the error was detected by the PSAM application (for example, a data format error in the command from the MAD Handler application), the PSAM will respond with a full Terminal Message containing an Application Status Word different from '0000'. Additionally, the response may also contain application specific error information (Error Response Data).
- If an error was detected on transport layer, the PSAM may only respond with SW1SW2.

Note: For PSAM application commands (which originate from the terminal application), this section documents the command and response formats as sent and received by the terminal application. To aid PSAM developers and designers, the part of the message exchanged between the PSAM and PSAM Handler is shaded.

For the two PSAM-Handler-originated commands, the C-APDU format is shown.

Table 107: Successful response to PSAM application command

Field	Value	Length (bytes)
Destination Address	Destination Address from the command message	2
Source Address	'00pp' where pp is the sub-address assigned to the PSAM	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0004' + Length of the response data.	2
Response Data	Response Data as defined for each PSAM command	var.
ASW1 ASW2	'0000'	2
RC	'0000'	2

Table 108: Error response to PSAM application command

Field	Value	Length (bytes)
Destination Address	Destination Address from the command message	2
Source Address	'00pp' where pp is the sub-address assigned to the PSAM	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0004' + Length of the Error Response Data.	2
Error Response Data	Application specific data returned in case of error.	var.
ASW1 ASW2	Must be different from '0000'	2
RC	'0000'	2

10.3.2 Application Status Words

Table 109 lists the Application Status Words that may be received from the PSAM application in a response to a command defined in this section.

Table 109: Application Status Words

ASW1	ASW2	Meaning
'00'	'00'	Successful
	all other	RFU
'01'	'00'	RFU
'02'	'00'	No information given
	'01'	Application not supported
	'02'	Function not supported
	'03'	PIN Pad is unresponsive
	'04'	PIN Pad unable to synchronize
	all other	RFU
'03'-0F'	all	RFU

ASW1	ASW2	Meaning
'1x'	all	Application-specific ASWs
'20'-'60'	all	RFU
'61'-'6F'	all	Reserved from conveying SW1SW2 as received from the Processor Card Reader.
'70' '90'	all	Reserved for future use
'91'-'9F'	all	Reserved from conveying SW1SW2 as received from the Processor Card Reader.
'A0'-'FF'	all	RFU

10.3.3 Start-up PSAM

The Start-up PSAM command is issued by a MAD Handler application to exchange identification information about the PSAM application, and to allow the PSAM application to perform any necessary initialization

10.3.3.1 The Start-up PSAM command must conform to the format defined in Table 110.

10.3.3.2 The Start-up PSAM command response must conform to the format defined in Table 111.

Table 110: Start-up PSAM command

Field	Value	Length
Destination Address	'00pp' where pp is the sub-address assigned to the PSAM	2
Source Address	'0100' for the MAD-Handler[2]	2
Message Type	'42'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0006' + L_c	2

[2] Normally PSAM application commands will originate from the terminal application in the MAD-Handler. However, TAPA does not preclude the ability to send commands from one PSAM to another. In that case, the source address would be '00xx' where xx is the sub-address of the sending PSAM.

Field	Value	Length
CLA	'B0'	1
INS	'02'	1
P1, P2	$ID_{PSAMAPP}$	2
L_c	'02'	1
ID_{THREAD}	Thread Identifier	1
PSAM sub-address	'pp'	1
L_e	'00'	1

Table 111: Start-up Command Response

Field	Value	Length (bytes)
Destination Address	The PSAM Handler will insert the address of the source of the command.	2
Source Address	'00pp' where pp is the sub-address assigned to the PSAM	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0011'+$L_{ApplicaitonData}$	2
PSAM Identification	RID_{PSAM} || $ID_{PSAMCREATOR}$ || ID_{PSAM}	13
Application Data	Data specific to an $ID_{PSAMAPP}$	$L_{Applicaiton Data}$
ASW1 ASW2	'0000'	2
RC	'0000'	2

10.3.4 Get Supported AIDs

The Get Supported AIDs command is issued by the MAD Handler to retrieve information about the supported AIDs for a specific PSAM application.

10.3.4.1 The Get Supported AIDs command must

conform to the format defined in Table 112.

10.3.4.2 The Get Supported AIDs response must conform to the format defined in Table 113.

Table 112: Get Supported AIDs Command

Field	Value	Length (bytes)
Destination Address	'00pp' where pp is the sub-address assigned to the PSAM	2
Source Address	'0100' for the MAD-Handler	2
Message Type	'42'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0006' + L_c	2
CLA	'B0'	1
INS	'08'	1
P1, P2	$ID_{PSAMAPP}$	2
L_c	'01'	1
ID_{THREAD}	Thread Identifier	1
L_e	'00'	1

Table 113: Response to Get Supported AIDs

Field	Value	Length (bytes)
Destination Address	The PSAM Handler will insert the address of the source of the command.	2
Source Address	'00pp' where pp is the sub-address assigned to the PSAM	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1

Field	Value	Length (bytes)
L_{DATA}	Variable	2
CNT_{AID}	Number of AIDs listed in this response The following fields (subscripted by "N") are repeated CNT_{AID} times (N=0 to CNT_{AID})	1
LEN_{AIDN}	Length of Nth AID	1
AID_N	Nth AID	5-16
$ID_{SCHEME,N}$	A reference number assigned to AID N by the acquirer.	1
ASW1 ASW2	'0000'	2
RC	'0000'	2

10.3.5 PSAM Shutdown

The PSAM Shutdown command is issued as an instruction to the PSAM application prior to withdrawing power from the PSAM.

10.3.5.1 The PSAM Shutdown command must conform to the format defined in Table 114.

10.3.5.2 The PSAM Shutdown response must conform to the format defined in Table 115.

Table 114: PSAM Shutdown command

Field	Value	Length (bytes)
Destination Address	'00pp' where pp is the sub-address assigned to the PSAM	2
Source Address	'0100' for the MAD-Handler	2
Message Type	'42'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0006' + L_c	2

Field	Value	Length (bytes)
CLA	‘B0’	1
INS	‘04’	1
P1P2	$ID_{PSAMAPP}$	2
L_c	‘01’	1
ID_{THREAD}	Thread Identifier	1
L_e	‘00’	1

Table 115: Response to PSAM Shutdown command

Field	Value	Length (bytes)
Destination Address	The PSAM Handler will insert the address of the source of the command.	2
Source Address	‘00pp’ where pp is the sub-address assigned to the PSAM	2
Message Type	‘FF’	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	‘0004’	2
ASW1 ASW2	‘0000’	2
RC	‘0000’	2

10.3.6 Get Next

The Get Next command is issued by the PSAM Handler, after receiving a response from the PSAM with SW1SW2 = ‘9601’, in order to get the next incremental response from the PSAM.

10.3.6.1 The Get Next command must conform to the format defined in Table 116.

10.3.6.2 The Get Next response must conform to the

format defined in Table 117.

Note: The Get Next command APDU is created directly by the PSAM Handler, and is never transmitted between handlers in the Terminal Message format.

Table 116: Get Next command

Field	Value	Length (bytes)
CLA	'B0'	1
INS	'FC'	1
P1	ID_{THREAD}	1
P2	'00'	1
L_e	'00'	1

Table 117: Response to Get Next command

Field	Value	Length (bytes)
Response Data	Next increment of Response data	var.
SW1SW2	'9000' or '9601' ('9000' indicates that this is the last increment of data to be given to the PSAM Handler)	2
SW1SW2	'6F01' Syntax error in command. Resend Command.	2
SW1SW2	'6F02' Abort chaining.	2

10.3.7 Response Command

The PSAM Handler issues the Response command in order to send response data from another terminal device to the PSAM.

10.3.7.1 The Response command must conform to the format defined in Table 118.

10.3.7.2 If the P2 in the command is '01', the PSAM must respond with an R-APDU consisting of only an SW1SW2 = '90 00'. The PSAM Handler will then send another response command containing additional data to be concatenated to the data already received.

When two or more response commands are "chained" as indicated by P2 = '01', the PSAM must concatenate the data portion from each command, left to right, until the final command with P2 = '00' is received. When all data have been received, the PSAM may then proceed with processing.

Note: The Response Command APDU is created directly by the PSAM Handler, and is never transmitted between handlers in the Terminal Message format.

Table 118: Response command

Field	Value	Length (bytes)
CLA	'B0'	1
INS	'FE'	1
P1	ID_{THREAD}	1
P2	'00' or '01'	1
L_c	Length of the response data	1
Response Data	Response data from other device	variable
L_e	'00'	1

10.3.8 Synchronize PSAM - PIN Pad/Secure Cryptograhic Device

The Synchronize PSAM/PIN Pad command is specific to the PIN Pad/Secure Cryptographic Device processing described in section 13.3, and is only used if the PSAM provides the

application control. PSAM applications that do not support the PIN Pad/Secure Cryptographic Device Processing, or do not provide application control, do not need to support this command.

10.3.8.1 The Synchronize PSAM/PIN Pad command must conform to the format defined in Table 119.

Table 119: Synchronize PSAM/PIN Pad command

Field	Value	Length (bytes)
Destination Address	'00pp' where pp is the sub-address assigned to the PSAM	2
Source Address	'0100' for the MAD-Handler	2
Message Type	'42'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0006' + L_c	2
CLA	'B0'	1
INS	'C2'	1
P1P2	$ID_{PSAMAPP}$	2
L_c	'01'	1
ID_{THREAD}	Thread Identifier	1
L_e	'00'	1

10.3.8.2 The Synchronize PSAM/PIN Pad response must conform to the format defined in Table 120.

10.3.8.3 The PSAM must return the appropriate Response Code if the Synchronize PSAM/PIN Pad command has not been processed correctly.

Table 120: Response to Synchronize PSAM/PIN Pad command

Field	Value	Length (bytes)
Destination Address	The PSAM Handler will insert the address of the source of the command.	2
Source Address	'00pp' where pp is the sub-address assigned to the PSAM	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'000C'	2
PIN Pad Identifier		
$ID_{PPCREATOR}$	Unique Id of PIN Pad Creator	4
ID_{PP}	Unique Id of PIN Pad/secure Device	4
ASW1 ASW2	'0000'	2
RC	'0000'	2

Table 121: ASW1-ASW2 Response Codes to Synchronize PSAM/PIN Pad command

ASW1-ASW2	Description
'0203'	*PIN Pad/Secure Cryptographic Device is unresponsive*
'0204'	*PIN Pad/Secure Cryptographic Device unable to synchronize*

11. The Data Store Handler

The Data Store Handler is responsible for managing the non-volatile memory of the terminal.

11.1 General requirements

The Data Store Handler only recognizes the common Handler commands: Open Handler and Close Handler.

11.2 Messages sent to the Data Store Handler

This section provides a list of additional commands that should be accepted and processed by the Data Store Handler.

11.2.1 File Management

11.2.1.1 The Data Store Handler must provide file management services as requested by other terminal components (typically MAD-Handler and PSAM applications). Terminal components must be able to request the storage of both keyed and non-keyed records grouped into files.

For the purposes of this section, a “record” is defined to consist of a string of “key data” (which may have length zero) and a string of “record data”.

11.2.1.2 If a keyed file is created, then each record stored in that file must have a unique key.

11.2.1.3 If a service is requested, it is fulfilled either entirely or not at all.

11.2.2 Create File

The Create File command is used to create one or more files within the terminal Data Store.

11.2.2.1 The Create File command must conform to the format defined in Table 122.

Table 122: Create File command

Field	Value	Length
Destination Address	'0500'	2
Source Address	Any	2
Message Type	'90'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0004'	2
NUM_{FILE}	Number of files of this type that should be created	1
LEN_{SKEY}	Maximum length of search key to associate with record. ('00' if no search key is used).	1
LEN_{REC}	Maximum length of a record	2

11.2.2.2 The Create File response must conform to the format defined in Table 123.

11.2.2.3 If there is insufficient memory to successfully process the Create File command, the Data Store Handler must return a Response Code indicating "Insufficient resources".

Table 123: Response to Create File command

Field	Value	Length
Destination Address	Any	2
Source Address	'0500'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	$2+2*NUM_{FILE}$	2
File Identifiers	N file identifiers of the created files	$2*NUM_{FILE}$
Response Code	Response Code	2

11.2.2.4 The Response Codes applicable to the Create

File command are defined in Table 124.

Table 124: Response Codes to Create File command

Response Code	Description
'FF51'	Invalid File ID
'FF52'	Record too large
'FF54'	File creation error.
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

11.2.3 Delete File

The Delete File command is used to delete one or more files within the terminal Data Store. File deletion may be necessary to recover the memory they occupy and release the File Ids associated with them.

11.2.3.1 The Delete File command must conform to the format defined in Table 125.

Table 125: Delete File command

Field	Value	Length
Destination Address	'0500'	2
Source Address	Any	2
Message Type	'91'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	$2*NUM_{FILE}$	2
File Identifiers	N file identifiers of the files to be deleted	$2*NUM_{FILE}$

11.2.3.2 The Delete File response must conform to the format defined in Table 126.

Table 126: Response to Delete File command

Field	Value	Length
Destination Address	Any	2
Source Address	'0500'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	2
Response Code	Response Code	2

11.2.3.3 The Response Codes applicable to the Delete File command are defined in Table 127.

Table 127: Response Codes to Delete File command

Response Code	Description
'FF51'	Invalid File ID
'FF55'	File could not be accessed.
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

11.2.4 Add File Record

The Add File Record command is used to add a record to an existing file within the terminal Data Store. Adding a record to a file means making an entry of the maximum file record + key size available.

11.2.4.1 The Add File Record command must conform to the format defined in Table 128.

11.2.4.2 The Data Store Handler must not reformat the file record data supplied in the DATA field.

11.2.4.3 If LEN_{REC} = '0000', the Data Store must reserve space for the maximum record size. However, the actual record length must be assigned as '0000' until a subsequent Update is received with a defined size record.

Table 128: Add File Record command

Field	Value	Length
Destination Address	'0500'	2
Source Address	Any	2
Message Type	'92'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0005' + LEN_{SKEY} + LEN_{REC}	2
ID_{FILE}	File to which the record must be added	2
LEN_{SKEY}	Length of search key to associate with record. ('00' if no search key is used, or if no data are present).	1
Key Data	Search Key data	LEN_{SKEY}
LEN_{REC}	Length of a record (may be '0000')	2
Record Data	Record data	LEN_{REC}

11.2.4.4 The Add File Record response must conform to the format defined in Table 129.

11.2.4.5 If the Data Store Handler returns the Response Code of "successful operation", the entire record must have been added to the file as requested.

11.2.4.6 If there is insufficient memory to successfully process the Add File Record command, the Data Store Handler must return a Response Code indicating "Insufficient resources".

Table 129: Response to Add File Record command

Field	Value	Length
Destination Address	Any	2
Source Address	'0500'	2
Message Type	'FF'	1

Field	Value	Length
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0004'	2
Record Pointer	Pointer to record within File	2
Response Code	Response Code	2

11.2.4.7 The Response Codes applicable to the Add File Record command are defined in Table 130.

Note: the Data Store Handler may reject the Add File Record command for a keyed file if the search key already exists. It is up to the application adding the record to ensure uniqueness of the search key.

Table 130: Response Codes to Add File Record command

Response Code	Description
'FF51'	Invalid File ID
'FF52'	Record too large
'FF53'	Search key too large
'FF55'	File could not be accessed.
'FF57'	File read error.
'FF58'	File write error.
'FF59'	Search key already existing
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.

Response Code	Description
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

11.2.5 Get File Record

The Get File Record command is used to retrieve data based on the record pointer within a given file. This function is non-destructive. Note that '0000' is an invalid record pointer, which may be returned when there is no next or previous record. In the context of this command, the previous record is the record that was last added to the file before the current record and the next record is the record that was added after the current record was added.

11.2.5.1 The Get File Record command must conform to the format defined in Table 131.

Table 131: Get File Record command

Field	Value	Length
Destination Address	'0500'	2
Source Address	Any	2
Message Type	'93'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0005'	2
ID_{FILE}	File from which the record must be retrieved	2
Record Pointer	Pointer to record to get. Not used (and should be '0000') when the Pointer Orientation is either '02' or '03'.	2

Field	Value	Length
Pointer Orientation	'00': Get the record, which was pointed to by the Record Pointer field. When the record has been returned, the record pointer must be set to the next record (allows FIFO processing). '01': Get the record, which was pointed to by the Record Pointer field. When the record has been returned, the record pointer must be set to the previous record (allows LIFO processing). '02': Get the first record in the file. When the record has been returned, the record pointer must be set to the next record (allows FIFO processing). '03': Get the last record in the file. When the record has been returned, the record pointer must be set to the next to last record (allows LIFO processing).	1

11.2.5.2 The Get File Record response must conform to the format defined in Table 132.

Table 132: Response to Get File Record command

Field	Value	Length
Destination Address	Any	2
Source Address	'0500'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	LEN_{SKEY} + LEN_{REC} + '0007'	2
LEN_{SKEY}	Length of search key associated with the retrieved record. ('00' if no search key is used).	1
Key Data	Search key data	LEN_{SKEY}
LEN_{REC}	Length of the record	2
Record Data	Retrieved record data	LEN_{REC}
Record Pointer	Pointer to next/previous record position within file	2
Response Code	Response Code	2

11.2.5.3 The Response Codes applicable to the Get File Record command are defined in Table 133.

Table 133: Response Codes to Get File Record command

Response Code	Description
'FF50'	Invalid record pointer. Record pointer outside the range defined for the current structure (Has not been added yet).
'FF51'	Invalid File ID
'FF55'	File could not be accessed
'FF56'	File seek error. A selected record (key) could not be found.
'FF57'	File read error
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

11.2.6 Update File Record

The Update File Record command is used to update an existing record with an amount of data that must not exceed the maximum indicated at file creation. The Update File Record command is destructive in that the previous content of the record is erased.

11.2.6.1 The Update File Record command must conform to the format defined in Table 134.

Note: The record to be updated must have been previously retrieved using either the Get File

Record command or the Find and Get File Record command. The Update Record command must specify the record pointer of the record to be updated. The data included in the Key Data field is used solely to update the indicated record, and not to locate it.

Table 134: Update File Record command

Field	Value	Length
Destination Address	'0500'	2
Source Address	Any	2
Message Type	'94'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0007' + LEN_{SKEY} + LEN_{REC}	1
ID_{FILE}	File in which the record must be updated	2
Record Pointer	Pointer to record to update The first record in a file may be addressed using '0000'.	2
LEN_{SKEY}	Length of search key to associate with record. ('00' if no search key is used).	1
Key Data	Search Key data	LEN_{SKEY}
LEN_{REC}	Length of a record	2
Record Data	New record data	LEN_{REC}

11.2.6.2 The Update File Record response must conform to the format defined in Table 135.

11.2.6.3 If the Data Store Handler returns the Response Code of "successful operation", the entire record, as specified in the Update command, must have been updated.

11.2.6.4 If the Data Store Handler rejects the command, the addressed record must not have been modified.

Table 135: Response to Update File Record command

Field	Value	Length
Destination Address	Any	2
Source Address	'0500'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0004'	2
Record Pointer	Pointer to record position within File	2
Response Code	Response Code	2

11.2.6.5 The Response Codes applicable to the Update File Record command are defined in Table 136.

Table 136: Response Codes to Update File Record command

Response Code	Description
'FF50'	Invalid record number. Record number outside the range defined for the current structure (Has not been added yet).
'FF51'	Invalid File ID
'FF52'	Record too large
'FF53'	Search key too large
'FF55'	File could not be accessed.
'FF57'	File read error.
'FF58'	File write error.
'FF59'	Search key already existing
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later

Response Code	Description
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

11.2.7 Find and Get File Record

The Find and Get File Record command is used to locate and retrieve an existing record based on the associated key. This function is non-destructive to the file record.

11.2.7.1 The Find and Get File Record command must conform to the format defined in Table 137.

Table 137: Find and Get File Record command

Field	Value	Length
Destination Address	'0500'	2
Source Address	Any	2
Message Type	'95'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0003' + LEN_{SKEY}	2
ID_{FILE}	File from which the record must be retrieved	2
LEN_{SKEY}	Length of search key associated with the retrieved record.	1
Key Data	Search Key data	LEN_{SKEY}

11.2.7.2 The Find and Get File Record response must conform to the format defined in Table 138.

Table 138: Response to Find and Get File Record command

Field	Value	Length
Destination Address	Any	2
Source Address	'0500'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0007' + LEN_{SKEY} + LEN_{REC}	2
LEN_{SKEY}	Length of the search key associated with the retrieved record. This must be the same as the length specified when the record was Added.	1
Key Data	Search Key data	LEN_{SKEY}
LEN_{REC}	Length of the retrieved record	2
Record Data	Retrieved record data	LEN_{REC}
Record Pointer	Pointer to retrieved record	2
Response Code	Response Code	2

11.2.7.3 The Response Codes applicable to the Find and Get File Record command are defined in Table 139.

Table 139: Response Codes to Find and Get File Record command

Response Code	Description
'FF50'	Invalid record number. Record number outside the range defined for the current structure (Has not been added yet).
'FF51'	Invalid File ID
'FF53'	Search key too large
'FF55'	File could not be accessed.
'FF56'	File seek error. A selected record (key) could not be found.
'FF57'	File read error.

Response Code	Description
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

11.2.8 Delete File Record

The Delete File Record command is used to delete a record based on the record pointer for a given file. This function not only erases the data from the record but also frees the record space associate with it.

11.2.8.1 The Delete File Record command must conform to the format defined in Table 140.

Table 140: Delete File Record command

Field	Value	Length
Destination Address	'0500'	2
Source Address	Any	2
Message Type	'96'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0004'	2
ID_{FILE}	File from which the record must be deleted	2

Field	Value	Length
Record Pointer	Pointer to record to delete The first record in a file may be addressed using '0000'.	2

11.2.8.2 The Delete File Record response must conform to the format defined in Table 141.

Table 141: Response to Delete File Record command

Field	Value	Length
Destination Address	Any	2
Source Address	'0500'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	2
Response Code	Response Code	2

11.2.8.3 The Response Codes applicable to the Delete File Record command are defined in Table 142.

Table 142: Response Codes to Delete File Record command

Response Code	Description
'FF50'	Invalid record pointer. Record pointer outside the range defined for the current structure (Has not been added yet).
'FF51'	Invalid File ID
'FF55'	File could not be accessed.
'FF57'	File read error.
'FF58'	File write error.
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.

Response Code	Description
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

11.2.9 Find and Delete File Record

The Find and Delete File Record command is used to locate and erase a record based on the search key from a given file. This function not only erases the data from the record but also frees the actual record space associated with it.

11.2.9.1 The Find and Delete File Record command must conform to the format defined in Table 143.

Table 143: Find and Delete File Record command

Field	Value	Length
Destination Address	'0500'	2
Source Address	Any	2
Message Type	'97'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0003' + LEN_{SKEY}	2
ID_{FILE}	File from which the record must be deleted	2
LEN_{SKEY}	Length of search key associated with the record to delete.	1
Key Data	Search Key data	LEN_{SKEY}

11.2.9.2 The Find and Delete File Record response must

conform to the format defined in Table 144.

Table 144: Response to Find and Delete File Record command

Field	Value	Length
Destination Address	Any	2
Source Address	'0500'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	2
Response Code	Response Code	2

11.2.9.3 The Response Codes applicable to the Find and Delete File Record command are defined in Table 145.

Table 145: Response Codes to Find and Delete File Record command

Response Code	Description
'FF50'	Invalid record pointer. Record pointer outside the range defined for the current structure (Has not been added yet).
'FF51'	Invalid File ID
'FF53'	Search key too large
'FF55'	File could not be accessed.
'FF56'	File seek error. A selected record (key) could not be found.
'FF57'	File read error.
'FF58'	File write error.
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later

Response Code	Description
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

11.2.10 Clear File

The Clear File command is used to delete all records from a specified file. This function not only erases the data from the record but also frees the actual record space associated with it. However, the cleared file remains allocated to the previously defined File ID.

11.2.10.1 The Clear File command must conform to the format defined in Table 146.

Table 146: Clear File command

Field	Value	Length
Destination Address	'0500'	2
Source Address	Any	2
Message Type	'98'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0002'	2
ID_{FILE}	File from which the records must be deleted	2

11.2.10.2 The Clear File response must conform to the format defined in Table 147.

Table 147: Response to Clear File command

Field	Value	Length
Destination Address	Any	2
Source Address	'0500'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	2
Response Code	Response Code	2

11.2.10.3 The Response Codes applicable to the Clear File command are defined in Table 148.

Table 148: Response Codes to Clear File command

Response Code	Description
'FF51'	Invalid File ID
'FF55'	File could not be accessed.
'FF57'	File read error.
'FF58'	File write error.
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

11.3 Summary

Table 149: Data Store Handler specific commands

Destination Address	Source Address	Message Type	Description
'0500'	Any	'90'	Create File
'0500'	Any	'91'	Delete File
'0500'	Any	'92'	Add File Record
'0500'	Any	'93'	Get File Record
'0500'	Any	'94'	Update File record
'0500'	Any	'95'	Find and Get File Record
'0500'	Any	'96'	Delete File Record
'0500'	Any	'97'	Find and Delete File Record
'0500'	Any	'98'	Clear File

12. The Communication Handler

The Communication Handler is responsible for managing the communication interface and providing application level online communication services to one or more host systems. The physical communication interface or protocol used to interact with an online host system is not dictated by this specification.

12.1 Messages sent to the Communication Handler

This section provides a list of additional commands that should be accepted and processed by the Communication Handler.

12.1.1 Initiate Communication Session

The Initiate Communication Session command is used to establish initial communication with an online host system. The information needed to establish this session must be conveyed in the Session Data field. The coding of the Session Data field is proprietary to the terminal and outside the scope of this specification.

Following the session setup, data is exchanged using the Read Handler String and Write Handler String commands.

12.1.1.1 The Initiate Communication command must conform to the format defined in Table 150. The coding of the Session Data field is proprietary to the terminal and outside the scope of this specification.

Table 150: Initiate Communication Session command

Field	Value	Length
Destination Address	'0600'	2
Source Address	Any	2
Message Type	'B0'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1

Field	Value	Length
L_{DATA}	Length of Session Data	2
Session Data	Data needed to initiate the communication session	Var.

12.1.1.2 The Initiate Communication Session response must conform to the format defined in Table 151.

Table 151: Response to Initiate Communication Session command

Field	Value	Length
Destination Address	Any	2
Source Address	'0600'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	2
Response Code	Response Code	2

12.1.1.3 The Response Codes applicable to the Initiate Communication Session command are defined in Table 152.

Table 152: Response Codes to Initiate Communication Session command

Response Code	Description
'FF60'	Invalid session setup parameters.
'FF62'	Connection in progress
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later

Response Code	Description
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

12.1.2 Terminate Communication Session

The Terminate Communication Session command is used to discontinue a communication session with a host system.

12.1.2.1 The Terminate Communication Session command must conform to the format defined in Table 153.

Table 153: Terminate Communication Session command

Field	Value	Length
Destination Address	'0600'	2
Source Address	Any	2
Message Type	'B1'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0000'	2

12.1.2.2 The Terminate Communication Session response must conform to the format defined in Table 154.

Table 154: Response to Terminate Communication Session command

Field	Value	Length
Destination Address	Any	2
Source Address	'0600'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	2
Response Code	Response Code	2

12.1.2.3 The Response Codes applicable to the Terminate Communication Session command are defined in Table 155.

Table 155: Response Codes to Terminate Communication Session command

Response Code	Description
'FF61'	No connection
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

12.2 Summary

Table 156: Communication Handler-Specific commands

Destination Address	Source Address	Message Type	Description
'0600'	Any	'60'	Initiate communication
'0600'	Any	'61'	Terminate Session

13. Event Handler

The Event Handler provides a mechanism for external events to be posted to the controlling application processes. Devices may post events to the Event queue by sending an Add Event Message to the Event Handler. Application processing code (either in the MAD-Handler or in the PSAM) may send messages to the Event Handler in order to retrieve events from the queue.

13.1 Event Types

The event type codes are defined in Table 157, with the addresses of the handlers where the events may have occurred.

Table 157: Event Types

Event Type Code	Event Description	Event Location	
'01'	Chip Card Inserted	Processor Card Reader	'0202'
		Memory Card Reader	'0203'
		Contactless Card Reader	'0204'
		PSAM Handler	'00pp'
'02'	Magnetic Stripe Card Swiped	Magnetic Stripe Reader	'0201'
'03'	Key Pressed	Customer Key Pad	'0303'
		Merchant Key Pad	'0401'
'04'	Incoming Call	Communication Handler	'0600'
'05'-'7F'	Reserved for Future Use		
'80'-'FF'	Reserved for Proprietary Use		

13.2 Event Handler Messages

The Event Handler must be able to process the commands defined in this section.

13.2.1 Add Event

The Add Event message is used to post an event to the end of the event queue.

13.2.1.1 An Add Event command must conform to the format defined in Table 158.

13.2.1.2 The Add Event message may originate from a device handler that is not able to assign a valid Thread Identifier. To ensure that there is no collision with on-going threads being managed by the MAD-Handler, the Event Handler must not send a response to the Add Event message.

The Add Event message is not a “command” requesting an action, and therefore does not transfer control. It is the logical equivalent of writing a message directly to the Event Queue.

13.2.1.3 The Event Handler must retain the Event Type Code and Event Location in the Event Queue.

Table 158: Add Event Command

Field	Value	Length
Destination Address	‘0700’	2
Source Address	Any	2
Message Type	'C0'	1
ID_{THREAD}	Any	1
L_{DATA}	‘0003’	2
Event Type Code	Type of Event	1
Event Location	Address of Event location	2

13.2.2 Get Event

The Get Event message is used to remove the oldest event from the event queue.

13.2.2.1 A Get Event command must conform to the

format defined in Table 159.

Table 159: Get Event Command

Field	Value	Length
Destination Address	'0700'	2
Source Address	Any	2
Message Type	'C1'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0000'	2

13.2.2.2 A Get Event response must conform to the format defined in Table 160.

13.2.2.3 The Event Handler must return the Response Code of "successful operation" if the Handler was able to successfully retrieve the oldest event from the event queue. The event must be removed from the queue as a result of a successful retrieval.

13.2.2.4 The Event Handler must return the appropriate Response Code if it is unable to retrieve an event from the event queue. The event must not be removed from the queue if the retrieval was unsuccessful.

Table 160: Response to Get Event command

Field	Value	Length
Destination Address	Any	2
Source Address	'0700'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0005'	2
Event Type Code	Type of Event	1

Field	Value	Length
Event Location	Address of Event location	2
Response Code	Response Code	2

13.2.2.5 The Response Codes applicable to the Get Event command are defined in Table 161.

Table 161: Response Codes to Get Event command

Response Code	Description
'FF72'	*No Events in Queue*
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

13.2.3 Find Event

The Find Event message is used to find the first (or oldest) message of a particular type, or for a particular location, and remove it from the event queue.

13.2.3.1 A Find Event command must conform to the format defined in Table 162.

Table 162: Find Event Command

Field	Value	Length
Destination Address	'0700'	2
Source Address	Any	2
Message Type	'C2'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0004'	2
Search Type	See Table 179	1
Event Type Code	Event Type to find	1
Event Location	Event location to find	2

13.2.3.2 A Find Event response must conform to the format defined in Table 163.

13.2.3.3 The Event Handler must return the Response Code of "successful operation" if the Handler was able to successfully find and retrieve an event from the queue. The event must be removed from the queue as a result of a successful retrieval.

13.2.3.4 The Event Handler must return the appropriate Response Code if it is unable to find or retrieve an event from the event queue. No event must be removed from the queue if the retrieval was unsuccessful.

Table 163: Response to Find Event command

Field	Value	Length
Destination Address	Any	2
Source Address	'0700'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1

Field	Value	Length
L_{DATA}	'0005'	2
Event Type Code		1
Event Location		2
Response Code	Response Code	2

13.2.3.5 The Response Codes applicable to the Find Event command are defined in Table 164.

Table 164: Response Codes to Find Event command

Response Code	Description
'FF72'	*No Events in Queue*
'FF73'	*No Matching Events in Queue*
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

13.2.4 Flush Event Queue

The Flush Event Queue is used remove all outstanding events from the event queue.

13.2.4.1 A Flush Event Queue command must conform to the format defined in Table 165.

Table 165: Flush Event Queue Command

Field	Value	Length
Destination Address	'0700'	2
Source Address	Any	2
Message Type	'C3'	1
ID_{THREAD}	Thread Identifier assigned by the MAD-Handler	1
L_{DATA}	'0000'	2

13.2.4.2 A Flush Event Queue response must conform to the format defined Table 166.

13.2.4.3 The Event Handler must return the Response Code of "successful operation" if the Handler was able to successfully flush all events from the queue.

13.2.4.4 The Event Handler must return the appropriate Response Code if it is unable to empty the event queue. No event must be removed from the queue if the flush was unsuccessful.

Table 166: Response to Flush Event Queue command

Field	Value	Length
Destination Address	Any	2
Source Address	'0700'	2
Message Type	'FF'	1
ID_{THREAD}	Thread Identifier of the request	1
L_{DATA}	'0002'	2
Response Code	Response Code	2

13.2.4.5 The Response Codes applicable to the Flush Event Queue command are defined in Table 167.

Table 167: Response Codes to Flush Event Queue command

Response Code	Description
'FF72'	*No Events in Queue*
'FFF3'	*Handler Error*: generic message that an unspecified error has occurred.
'FFF5'	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFFB'	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.

13.3 Summary

Table 168: Event Handler-Specific commands

Destination Address	Source Address	Message Type	Description
'0700'	Any	'C0'	Add Event
'0700'	Any	'C1'	Get Event
'0700'	Any	'C2'	Find Event
'0700'	Any	'C3'	Flush Event Queue

14. Secure Cryptographic Device Processing

14.1 Overview

A terminal may support one or more Secure Cryptographic Devices. The Card Reader may be integrated with the PIN Pad or be an independent Secure Cryptographic Device. Terminals not supporting a PIN Pad may support another Secure Cryptographic Device, e.g. in a Secure Card Reader.

A Secure Cryptographic Device may utilize the same PKC encryption scheme as a PIN Pad or for Secure Cryptographic Devices not supporting PKC use a proprietary key management scheme.

14.2 PIN Pad processing

In order to support applications that require use of PINs, this specification defines a method of establishing and using a secure zone between a PSAM application and the PIN Pad/Secure Cryptographic Device. This facility provides the following features:

- The acquirer's online PIN Encryption keys are maintained in the PSAM, rather than the PIN Pads.
- Application-specific logic for PIN verification is maintained in the PSAM rather than the PIN Pad.
- Multiple acquirers, responsible for different applications, may securely use the same PIN pad.

14.2.1 Physical Environment

14.2.1.1 The PIN Pad, with its keypad, must be contained within a Secure Cryptographic Device (SCD). The SCD may as well contain the Card Reader. The SCD must also contain User Interface Display. Each of these devices is addressed as specified in Table 2.

14.2.1.2 The Card Reader shall if it is a stand-alone unit, be a Secure Cryptographic Device by itself. The Card Reader must transfer sensitive information to other devices in a secure way.

14.2.2 Establishing the Secure Zone

To transmit enciphered PINs between the PIN Pad and a PSAM application, a secure zone must first be established between the two entities. This is done using public key cryptography to establish an initial symmetric PIN Session master key. Subsequently, symmetric key cryptography (triple-DES) is used for PIN encryption and message authentication between the entities.

The PIN Pad and the PSAM each contain their own unique asymmetric key pairs (SK_{PP}, PK_{PP}) and (SK_{PSAM}, PK_{PSAM}). The public keys are certified by the PIN Pad Creator and by the Acquirer's agent, known as the PSAM Creator, respectively. (For the remainder of this section, this entity will be referred to as the Acquirer. If the PSAM's are created by multiple systems on behalf of the Acquirer, each system must have its own unique PSAM Creator Identifier and must have a different set of public keys.)

The required key hierarchy for the PIN Pad and the PSAM application is illustrated in Figure 11. At the top of this key hierarchy is a Certification Authority (CA), which is managed by the Acquirer.

For the PSAM application, the CA creates an Acquirer certificate on the Acquirer public key, and the Acquirer in turn creates a PSAM certificate on the PSAM public key.

Similarly, for PIN Pads, the CA creates a PIN PAD Creator certificate, and the PIN Pad Creator creates a PIN Pad certificate.

The required keys and certificates are inserted into the PIN Pad and PSAM during their personalization processes.

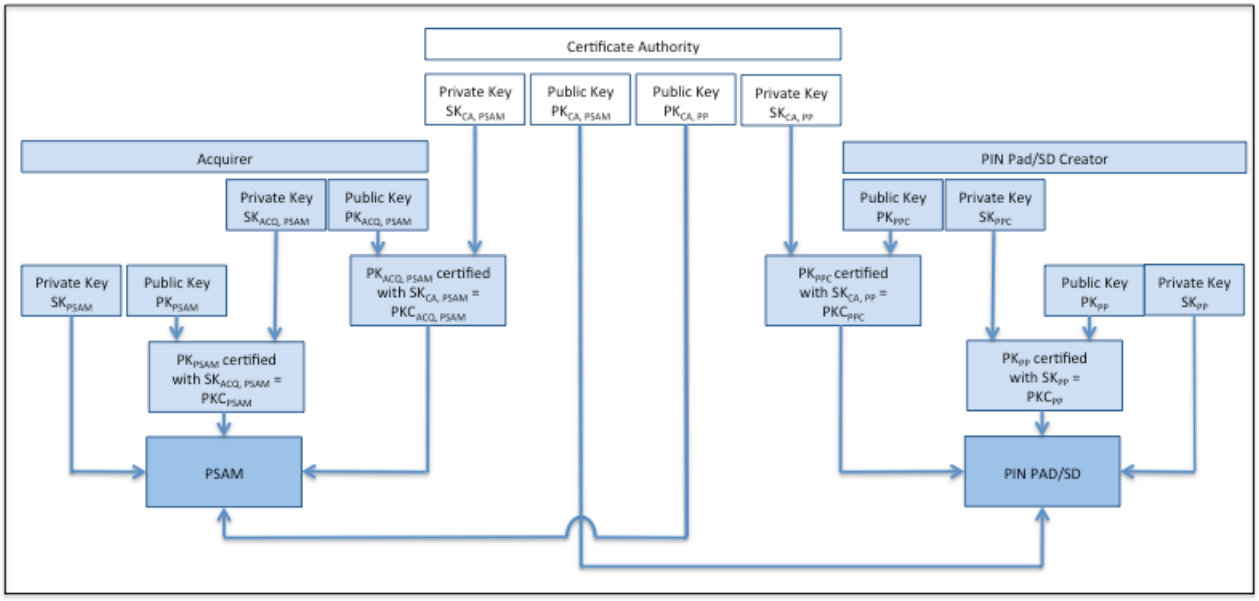

Figure 11: PIN Pad and PSAM Key Hierarchy

14.2.3 Supported Configurations

The PIN processing supports a variety of POS configurations as illustrated in Figure 12.

- Environment 1 illustrates a one-to-one relationship between the PSAM and the PIN Pad, as may be found in a stand-alone POS environment.
- Environment 2 illustrates multiple PIN Pads associated with a single PSAM, as might be the case in a distributed multi-lane stored environment.
- Environment 3 illustrates multiple PSAMs, owned by the same Acquirer, which are associated with one or more PIN Pads. This might be required in a multi-lane environment where an Acquirer uses more than one PSAM for backup and/or load balancing.
- Environment 4 illustrates an environment with multiple PSAMs, which are owned by different Acquirers, associated with one or more PIN Pads. Note that this environment introduces some special security issues, which are further discussed in Section 14.6.

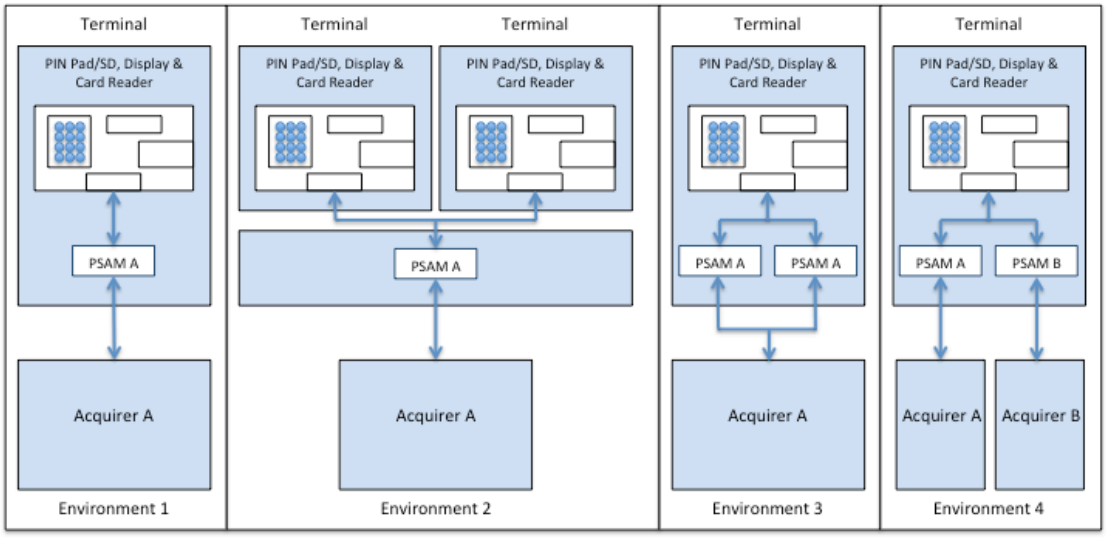

Figure 12: PIN Pad/PSAM Environments

14.2.4 Implementation

The PIN Processing defined in these specifications is implemented within a TAPA application. Certain specific functions must be securely performed within the PSAM itself. For all other functions, the choice of whether the function is performed within the PSAM or the MAD-Handler part of the application is up to the application designer.

The functions that must be performed within the PSAM are specified in section 14.6.3.

14.3 PIN Pad/PSAM Initialization

During PSAM initialization, each PSAM application that uses PIN Processing must be synchronized with the PIN Pad(s). In order to perform synchronization, the application engages in a dialogue with the PIN Pad in order to establish a secure zone using a shared symmetric key.

14.3.1.1 The application must begin the synchronization process by sending the Get Key Check Value command to the PIN Pad. The response identifies the PIN Pad, and provides information about its current keys, including a check value (KCV_{PIN}) of the current Transaction Session Key (KSES). This check value must be compared to the key check value of the PSAM's current PIN Session master key (KCV_{PSAM}). If the check values are the same, then the two entities are currently synchronized and no further dialogue is necessary.

14.3.1.2 The application must send the Get PIN Pad Public Key Record commands to the PIN Pad to retrieve the PIN Pad certificates. These certificates must be verified by the PSAM, and the PIN Pad's public key recovered from them.

14.3.1.3 The application must send PSAM certificates to the PIN Pad, using the Verify PSAM Public Key Certificate command. The PIN Pad must verify the PSAM's certificates and recover the PSAM's public key.

Note: The required processing for verifying the certificates and recovering the public keys is defined in section 14.7.

14.3.1.4 The PSAM must generate an Initial Session Key, which must be sent to the PIN Pad by the application using the Submit Initial Key command. The Submit Initial Key command contains a public-key signature (PS), which must be generated by the PSAM. The PIN Pad must verify the PS and recover from it the Initial Session Key.

Note: The required processing for generating and verifying the PS and recovering the Initial Session Key is defined in section 14.7.

14.3.1.5 If the application control is implemented in the PSAM, the MAD-Handler application must initiate the synchronization process by sending a Synchronize PIN Pad command to the PSAM application for each PIN Pad with which the PSAM must have a relationship. This command is defined in section 10.3.

Figure 13 illustrates the message flow for the case where the application control is implemented within the PSAM.

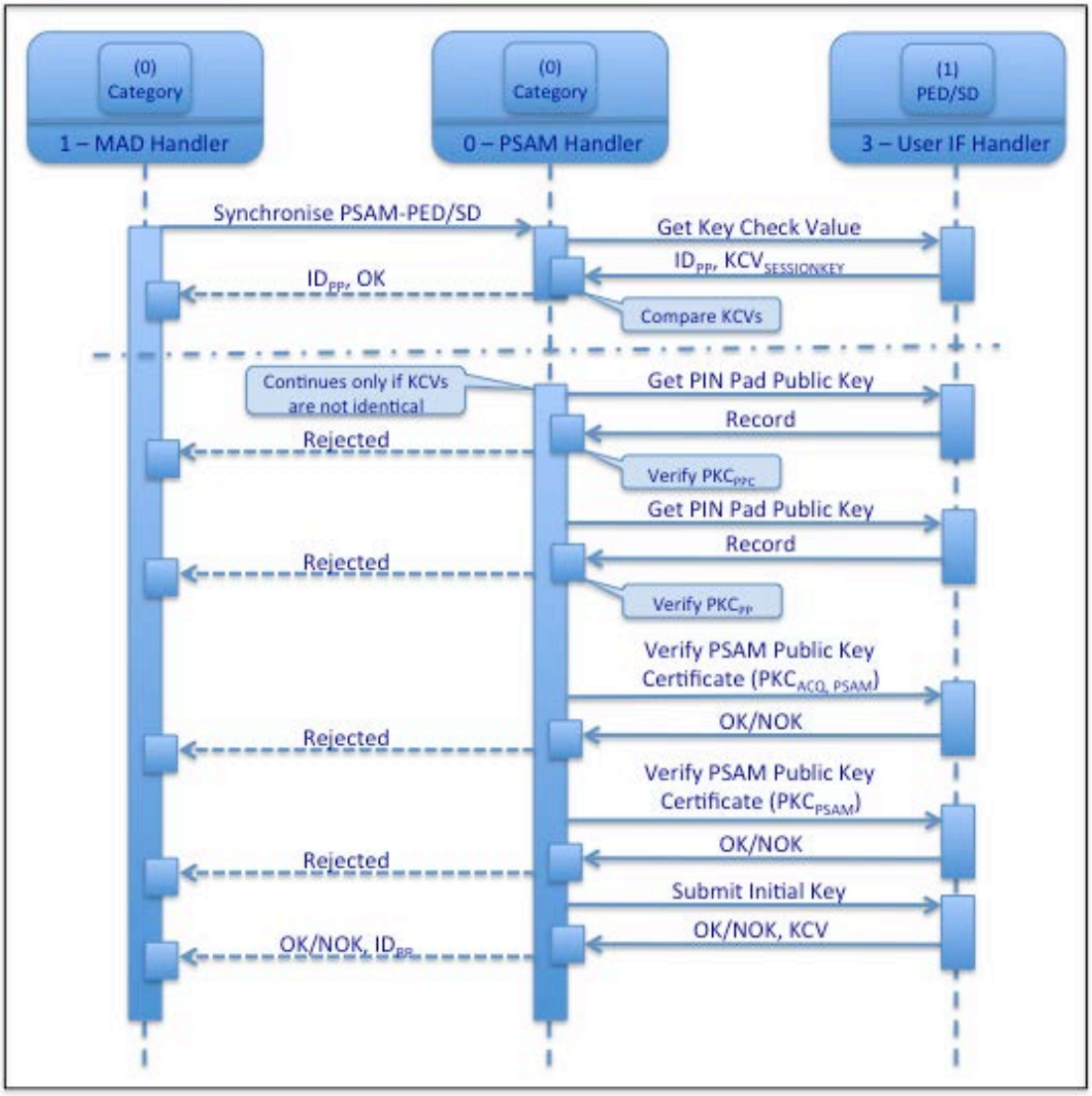

Figure 13: PSAM/PIN Pad Initialization

14.4 PIN Processing

14.4.1 Secure Cryptographic Device State

In a device with a PIN Pad, there are special security requirements that apply to the Secure Cryptographic Device (SCD). These requirements ensure that PINs are never revealed outside of the secure environment.

14.4.1.1 The SCD must have two possible states: Default State and PIN Entry State.

14.4.1.2 The SCD must be in Default State after terminal initialization.

14.4.1.3 The SCD must be put into PIN Entry State after the PIN Pad has received and authenticated a valid Initiate PIN Entry command. The SCD must not transition to PIN Entry State under any other circumstances.

14.4.1.4 The SCD must be returned to Default State when the PIN Pad receives a Terminate PIN Entry command. This command is not authenticated, and is not signed by the PSAM.

14.4.1.5 When the SCD is in Default State:

- The numeric keys on the PIN Pad must be disabled.
- No authentication is required on messages that send text to the User Display. Any application may freely send display messages to the User Interface Handler display.
- The Processor Card Reader must only accept plaintext Card commands sent using the ICC Command message. The Verify Offline PIN Command message must not be accepted.

14.4.1.6 When the SCD is in PIN Entry State:

- The numeric keys are enabled on the PIN Pad.
- The User Interface Handler must only accept “authorized” messages, which cause text to be displayed. An authorized message includes messages that contain a MAC generated by the PSAM, which sent the Initiate PIN Entry. Alternatively, it may be a command that supplies a numbered message that has been personalized into the SCD as valid.
- The Processor Card Reader may accept encrypted Card commands sent using the Verify Offline PIN Command message.

14.4.2 PIN Entry

14.4.2.1 The application must begin the process of PIN entry by sending The Initiate PIN Entry command to the PIN Pad. The PSAM must generate a new set of PIN Session Keys, and use the PIN MAC Session key ($KSES_{MAC}$) to sign the command.

This Initiate PIN Entry command is a “macro” command, which causes the PIN Pad to perform the following functions:

- The PIN Pad generates a new set of PIN session keys.
- The MAC on the command is validated using the new PIN session MAC key ($KSES_{MAC}$).
- The Secure Cryptographic Device is placed into PIN Entry State.
- The cardholder is prompted to enter a PIN by, for example, sending a Display Message with Message code ‘09’ to the User Interface Display.
- The PIN Pad may then respond to the Initiate PIN Entry command. (Depending on the implementation, the response may be sent before the consumer has finished performing the PIN entry.)

14.4.2.2 The application must send an authenticated Get PIN command to the PIN Pad in order to retrieve the PIN block. The PIN Pad will respond with the PIN Block encrypted under the current PIN Encryption Session Key ($KSES_{PIN}$). The PSAM must validate the MAC in the response, and decrypt the PIN Block.

14.4.2.3 During PIN entry, a symbol (for example, an asterisk character “*”) must be displayed at the user display instead of the PIN digit.

14.4.2.4 Error handling procedures, e.g. deletion of incorrect entered PIN digits, must be handled internally by the Secure Cryptographic Device.

In some environments, the application may send additional commands to the User Interface Handler while PIN Entry is in progress. As noted previously, any commands that require text to be displayed must be “authorized”.

In particular, the application may send a Confirm Amount message during this period. This command must be authenticated while in PIN Entry State.

The particular means of handling a Confirm Amount received during PIN Entry is environment specific. One possible implementation is to display the amount confirmation request along with the PIN entry request. In this case a single key press from the consumer will serve both as the amount confirmation and as the PIN entry.

The following three diagrams (Figure 14, Figure 15 and

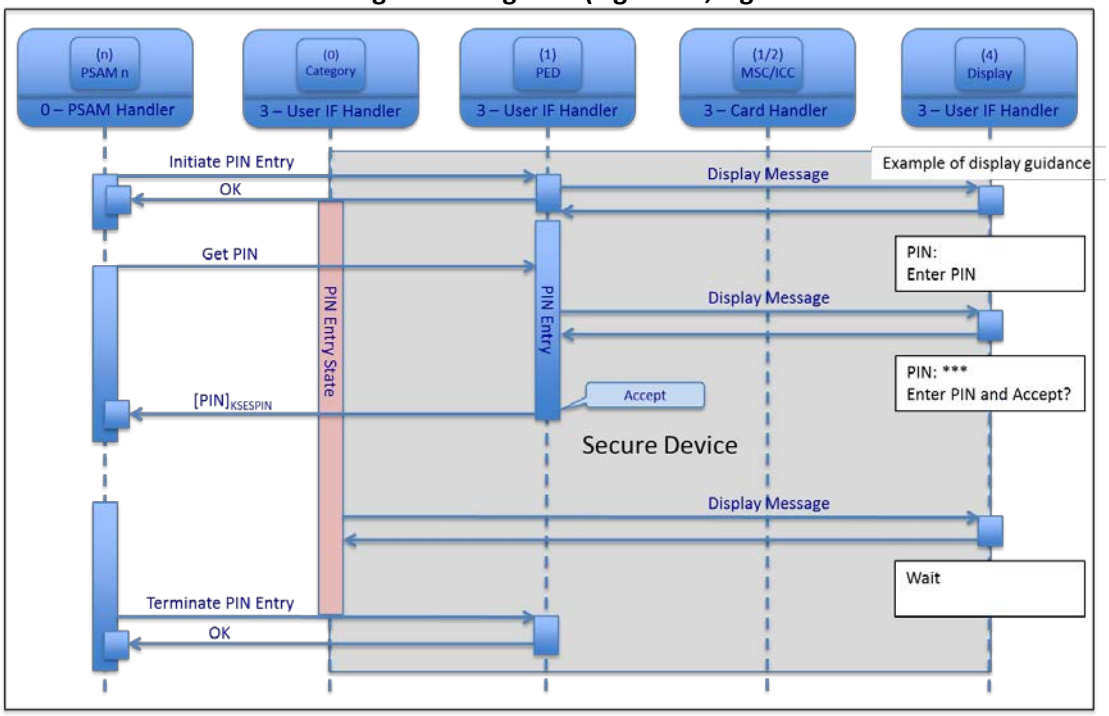

Figure 16) are examples of message flows within the Secure Cryptographic Device, and of the user interface, during PIN Entry. These sample diagrams illustrate the case where the application control is within the PSAM.

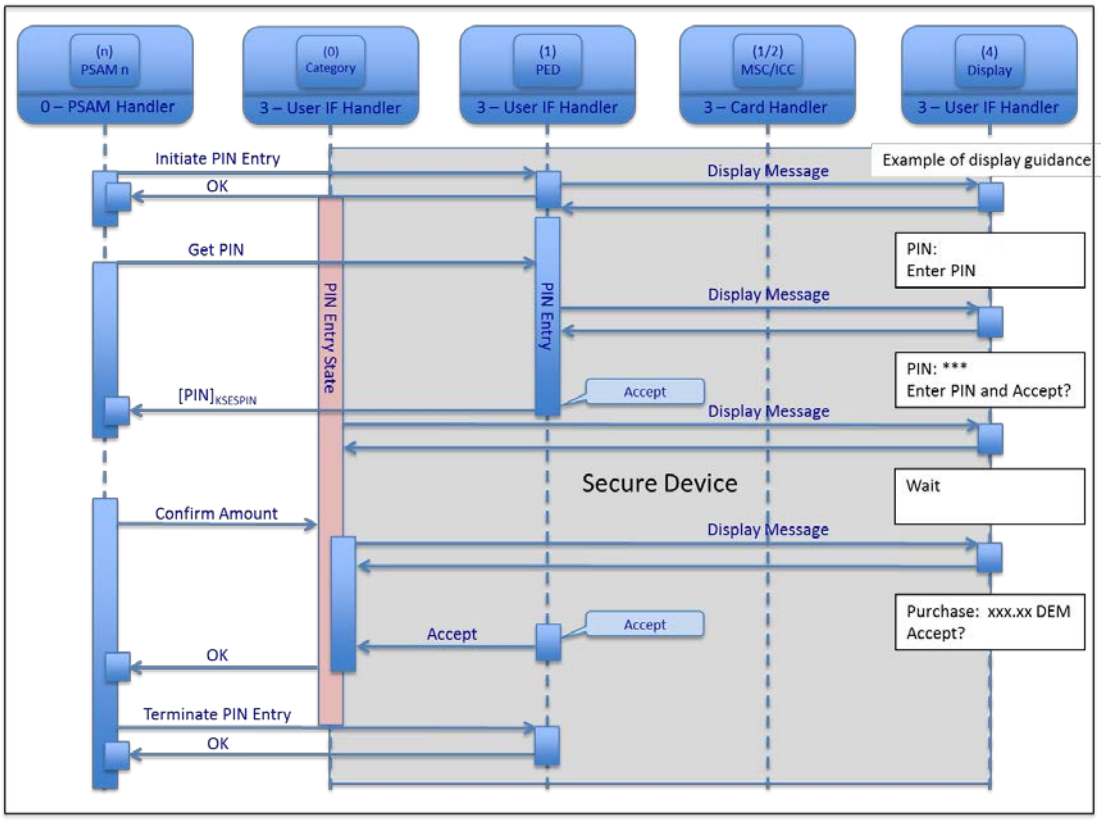

Figure 14: Separate PIN Entry and Amount Confirmation

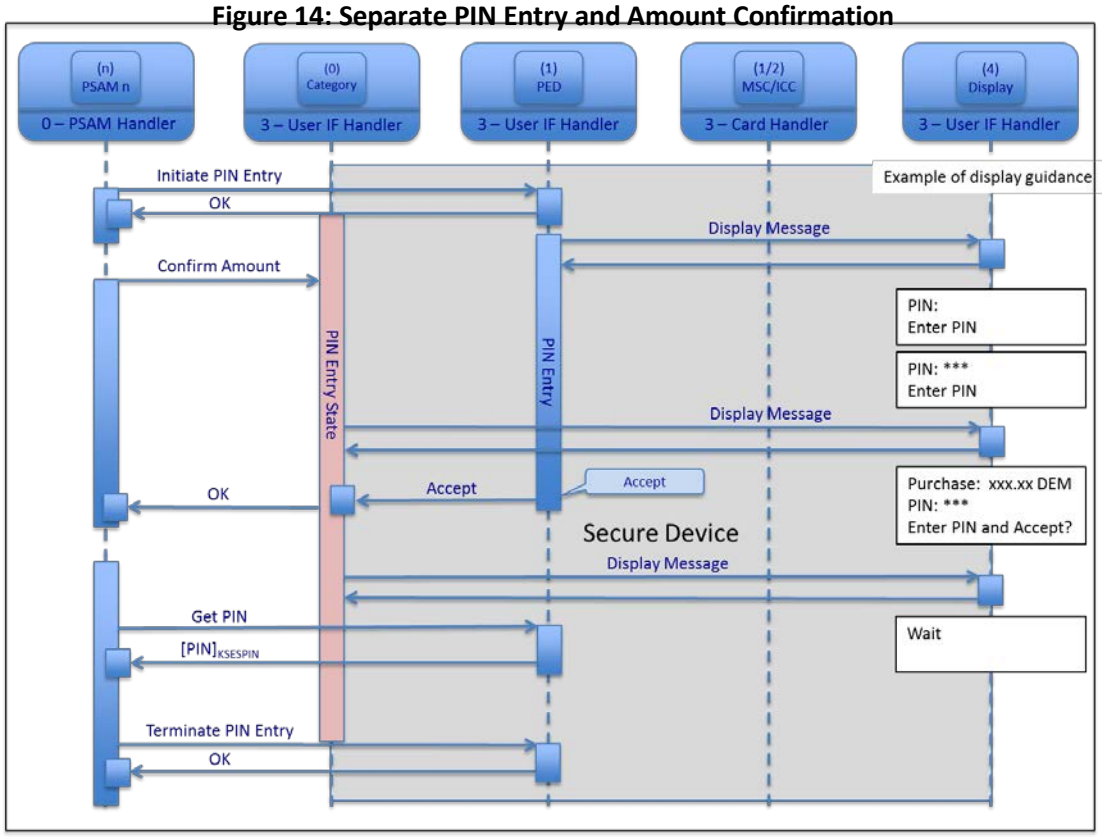

Figure 15: Combined PIN Entry and Amount Confirmation

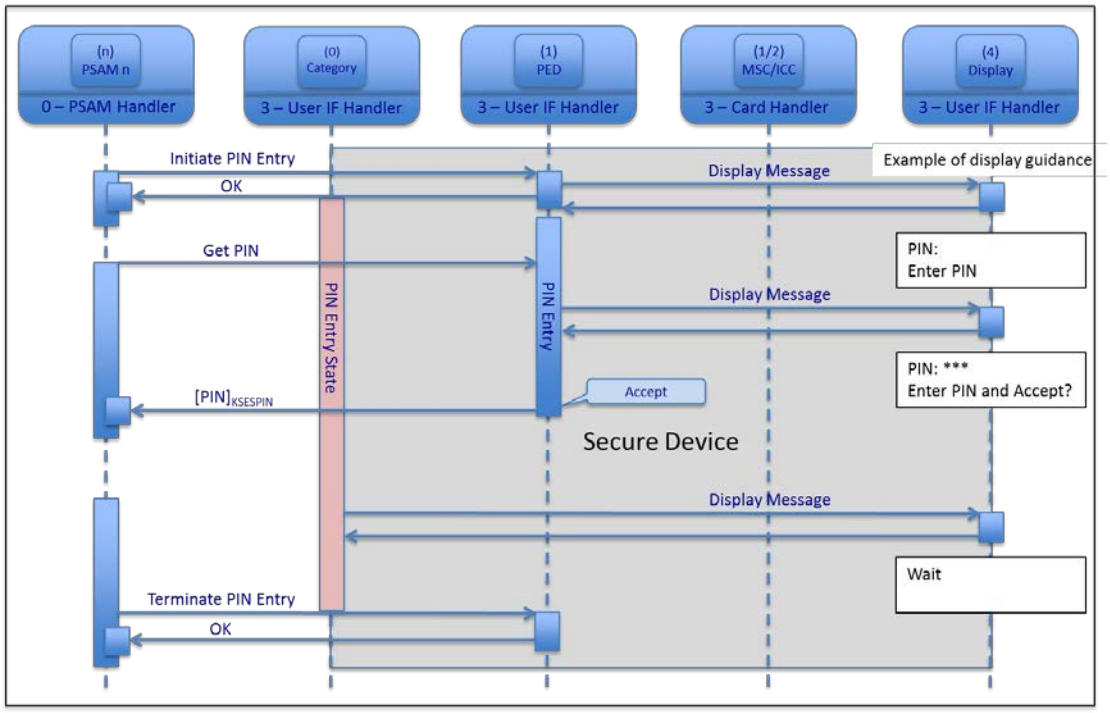

Figure 16: PIN Entry with no Amount Confirmation

14.5 PIN Verification

Depending on the requirements of the payment application, PIN verification may be performed in one of three ways:

1. Online PIN verification, where the PIN is sent encrypted to the acquirer for transmission to the card issuer.
2. Offline plaintext PIN verification, where the PIN is sent to the card for verification in the clear.
3. Offline encrypted PIN verification, where the PIN sent to the card for verification encrypted under a key known to the card.

14.5.1 Online PIN Verification

If online PIN verification is to be performed, the application sends the PIN to the Acquirer encrypted in accordance with the method implemented by the Acquirer. This might for example use a PIN Encryption key established between the PSAM and Acquirer. The approach taken, while it must comply with the PCI security requirements scheme, is specific to the Acquirer and outside the scope of this document.

14.5.1.1 The application must retrieve the PIN encrypted under a key specified by the Acquirer. In order to accomplish this, the PSAM must decipher the PIN block using the PIN Encryption Session Key ($KSES_{PIN}$), and then re-encipher it as specified by the Acquirer, using a PIN Encryption key shared between the PSAM and the acquirer.

Note that this function may be performed regardless of the state of the SCD.

Figure 17 illustrates an example of online PIN handling.

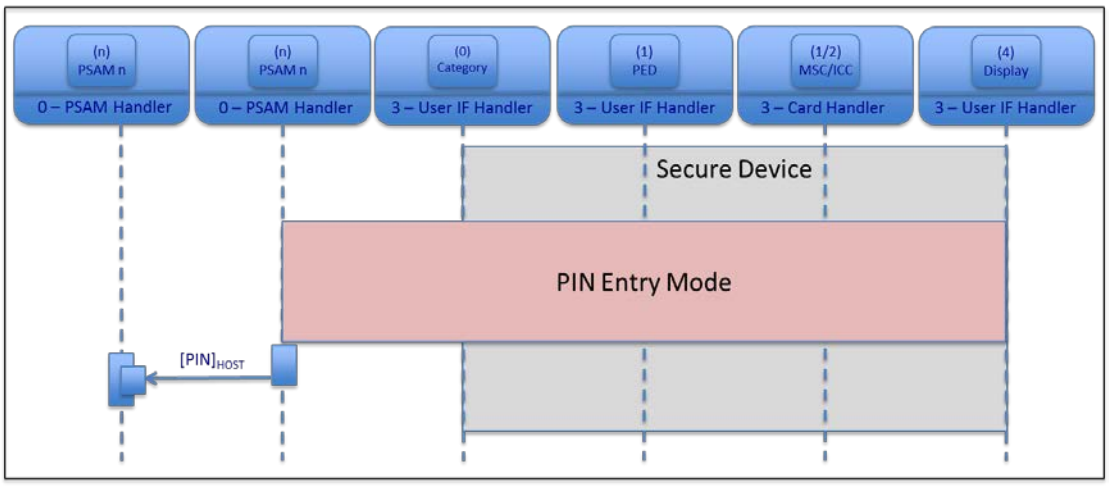

Figure 17: Online PIN Verification

14.5.2 Offline PIN Verification

When offline PIN verification is to be performed, the application sends a command containing the PIN to the card. This command may contain a plaintext PIN, or may contain a PIN that has been encrypted under a key known to the card. (For example, the Verify command is used in the EMV application. Depending on the card's requirements, the PIN may be encrypted under the card's public key.)

14.5.2.1 The PSAM must encrypt the PIN verification command APDU under the PIN session encryption key. The encrypted command must be sent to the Processor Card Reader using the Verify Offline PIN message. This message must be added a MAC using the PIN MAC Session Key ($KSES_{MAC}$).

The message is authenticated and the command APDU decrypted within the Secure Cryptographic Device. The C-APDU is then forwarded to the card.

14.5.2.2 The response to the Verify Offline PIN message contains the card application's response to the PIN verification command. The response message contains a MAC, which must be verified by the PSAM.

Note that, as previously specified; offline PIN verification may only be performed while the SCD is in PIN Entry State.

14.6 Security Requirements

14.6.1 Business Entities

14.6.1.1 The Primary Acquirer is the entity responsible for specifying, developing and maintaining the PIN Pads and (at least) one of the PSAMs (even if subcontracted to a third party). The Primary Acquirer may be the Certification Authority.

14.6.1.2 The Certification Authority (CA) is responsible for certifying the Acquirers` PIN processing systems (including Host systems, PSAMs and PIN Pads).

14.6.1.3 The certification of the Acquirer public keys must represent the approval by the CA of the Acquirer's PSAM and Host-based PIN processors.

14.6.1.4 The Card Schemes for which PIN processing is performed must approve the CA.

14.6.1.5 If a Secondary Acquirer introduces a new application (and PSAM), which requires PIN entering, it is the responsibility of the CA to certify that the level of security provided by the Secondary Acquirer is sufficient.

14.6.1.6 The Primary Acquirer may permit any number of (certified) Secondary Acquirer PSAMs to be installed in their terminal and thereby have access to the PIN Pad(s).

14.6.1.7 The Primary Acquirer must know the identities of all the PSAMs in each of their terminals.

14.6.1.8 The Primary Acquirer must know the identities of all the PIN Pads configured with each of their PSAMs.

14.6.1.9 The $ID_{PPCREATOR}$ and ID_{PP} must uniquely identify the PIN Pad to the Acquirer. There is no requirement that the PIN Pad be globally identifiable.

14.6.2 Physical Security Requirements

The requirements for the physical security of a payment and PIN handing terminal are governed by the PCI SSC in ref. 11: e "PCI PIN Transaction Security, PTS".

Consequently, said requirements are out of scope for this specification.

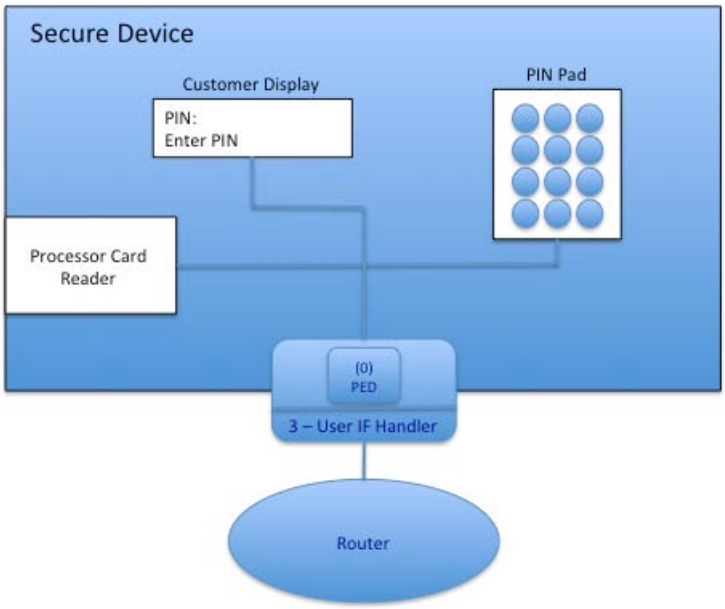

Figure 18: Secure Cryptographic Device

14.6.3 Logical Security Requirements

Also, the PCI SSC PTS requirements cover the logical security requirements.

In addition to the PTS logical security requirements, a number of requirements related to this application architecture are defined below.

14.6.3.1 The terminal and/or PSAM application must store the identities of each PSAM-PIN Pad configured pair. This must be available to the acquirer along with any other status information required by the acquirer.

14.6.3.2 The PIN Pad private RSA key must remain protected within the confines of the tamper responsive PIN Pad. All cryptographic operations using this key must be performed

within the tamper responsive PIN Pad.

14.6.3.3 The PSAM private RSA key must remain protected within the PSAM. All cryptographic operations using this key must be performed within the PSAM.

14.6.3.4 The set of PIN Session keys (including the KSES as well as the $KSES_{PIN}$ and $KSES_{MAC}$, which are derived from it), must be protected within the Secure Cryptographic Device and PSAM. The Initial Session Key, exchanged during synchronization, may only appear outside of the protected devices when encrypted as in the Submit Initial Key command.

In order to so protect the Initial Session Key it is necessary that the RSA and padding operations defined in sections 7.2.4.2 and 7.2.4.4 be performed within the protected devices.

14.6.3.5 The set of PIN Session keys must only be used in the manner specified within this document; they must not be used for any other purposes.

14.6.3.6 Only an authenticated Initiate PIN Entry command may cause the Secure Cryptographic Device to be put into PIN Entry state.

14.6.3.7 When in PIN Entry State the Display may only show messages authenticated by the PSAM. Authenticated messages includes generic write string messages sent from the PSAM with a MAC, as well as messages referenced by “Message Codes”, which have been personalized into the PIN Pad.

14.6.3.8 When in PIN Entry State, any commands that require authentication must not be accepted by the Secure Cryptographic Device, i.e. the PIN pad if they do not contain a MAC from the PSAM that sent the Initiate PIN Entry command.

14.6.4 Personalization Requirements

14.6.4.1 After personalization and initial synchronization, the PIN Pad must contain the data elements defined in Table 169.

14.6.4.2 After personalization and configuration the PSAM must contain the PIN related data elements defined in Table 170.

14.6.4.3 The PIN Pad and PSAM must each be personalized with at least one key/certificate hierarchy chain and one CA public key. Depending on the acquirer's requirements for the life span of the PIN Pad and PSAM and on the CA's requirements for migration to longer key lengths, additional chains and CA public keys may be inserted at personalization.

In the lists of data elements contained in the PIN Pad and in the PSAM, it is assumed that:

- The PIN Pad is personalized with N_1 key pairs and with M_2 PSAM CA public keys
- The PSAM is personalized with M_1 key pairs and with N_2 PIN PAD CA public keys.

Note that it is possible for N1 = N2 and/or M1 = M2.

Table 169: Data Elements contained in the PIN Pad

Description	Data element	Per PIN Pad/SCD	Per PSAM	Obtained at/with:
PIN Pad identifier	$ID_{PPCREATOR}$	1		Personalization
	ID_{PP}	1		Personalization
CA PP Key version	$VKP_{CA, PP}$	N_1		Personalization
PIN Pad Creator certificate	PKC_{PPC}	N_1		Personalization
PIN Pad Certificate	PKC_{PP}	N_1		Personalization
PIN Pads private key	SK_{PP}	N_1		Personalization
CA PSAM Key version	$VKP_{CA, PSAM}$	M_2		Personalization
CA PSAM public key	$PK_{CA, PSAM}$	M_2		Personalization
PSAM identifier	RID_{PSAM}		1	Submit Initial Key command

Description	Data element	Per PIN Pad/SCD	Per PSAM	Obtained at/with:
	$ID_{PSAMCREATOR}$		1	Submit Initial Key command
	ID_{PSAM}		1	Submit Initial Key command
Initial Session Key	$KSES_{INIT}$		1	Submit Initial Key command

Table 170: Data Elements contained in the PSAM

Description	Data element	Per PSAM	Per PIN Pad/SCD	Obtained at/with:
PSAM identifier	RID_{PSAM}	1		Personalization
	$ID_{PSAMCREATOR}$	1		Personalization
	ID_{PSAM}	1		Personalization
CA PSAM Key Version	$VKP_{CA, PSAM}$	M_1		Personalization
Acquirer certificate	PKC_{ACQ}	M_1		Personalization
PSAM Certificate	PKC_{PSAM}	M_1		Personalization
PSAM private key	SK_{PSAM}	M_1		Personalization
CA PP Key version	$VKP_{CA, PP}$	N_2		Personalization
CA PP public key	$PK_{CA, PP}$	N_2		Personalization
PIN Pad identifier	$ID_{PPCREATOR}$		1	PIN Pad Certificate
	ID_{PP}		1	PIN Pad Certificate
Initial Session Key	$KSES_{INIT}$		1	Randomly generated initial key

14.6.5 Minimum PSAM Requirements

14.6.5.1 The following PIN Pad processing functions must be performed by the PSAM and not by the application within the terminal:

- Generation of the $Challenge_{PSAM}$ presented in the Get Key Check Value command.

- Generation of the KCV_{PSAM}. The KCV_{PSAM} is used to validate the KCV_{PP} returned in the response to the Get Key Check Value command, and is sent to the PIN Pad in the Initiate PIN Entry command.
- Generation of the PS signature presented in the Submit Initial Key command.
- Generation of the MAC on all authenticated commands sent to the PIN Pad.
- Verification of the MAC on all authenticated responses received from the PIN Pad.
- Decryption and encryption of the PIN.
- Encryption of any commands used for PIN verification being sent to the IC card.

14.7 Cryptographic Requirements

14.7.1 Verifying a Certificate - General Requirements

This section defines the general requirements for certificate verification in accordance with ISO/IEC 9796-2. This corresponds to the process defined in Annex E.2.1.3 of EMV 3.1.1.

Verification of a certificate begins with recovery of the certificate data using the appropriate public key (either the CA public key or the key resulting from verification of the next higher level certificate) and its associated algorithm ALGP.

14.7.1.1 Recovery of the certificate data must be performed using the process described in Annex F.2.1 of EMV 3.1.1.

14.7.1.2 The recovery can only be performed if the length of the certificate is the same as the length of the modulus of the public key used in the verification. If the lengths are different, verification has failed.

14.7.1.3 After recovering the recoverable certificate data, the header (first byte) and the trailer (last byte) must be checked. The header must be '6A' (if there is an associated remainder field) or '4A' (if there is no associated remainder field)

and the trailer must be 'BC'. If this is not the case then verification of the certificate has failed.

14.7.1.4 If the public key algorithm indicator is not recognized then verification has failed.

14.7.1.5 The hash value is a 20 byte field immediately preceding the trailer (last byte) of the recovered certificate data and must be verified according to the following procedure:

The following data must be concatenated in order (left to right):

1. All data beginning with the format code in the recovered certificate data (which is always the second byte of the recovered certificate data) up to and including the last byte before the hash value, in the order in which it appears in the recovered certificate data.
2. The Public Key Remainder (PKR), if it exists.

14.7.1.6 The hash algorithm indicated in the certificate (SHA-1 is the only hash algorithm supported) must be applied to the concatenation, producing a 20-byte result. This result is compared to the hash value recovered from the certificate. If they are unequal, then certificate verification has failed.

14.7.2 Authentication of the PIN Pad Public Key

The PSAM must determine the correct set of certificates to be used for a transaction by examining the list of key versions ($VKP_{CA, PSAM}$) returned by the PIN Pad in its response to the Get Key Check Value command.

PSAM Authentication of the PIN Pad public key consists of:

1. Determination that a public key hierarchy in the PIN Pad can be processed by the PSAM.
2. Use of the Get PIN Pad Public Key Record command to retrieve certificates from the PIN Pad.
3. Verification of the certificates.

14.7.2.1 The PSAM must verify that a version ($VKP_{CA, PP}$) of the CA public key used to create the PIN Pad Creator certificate (and so identified in the response to the Get Key Check Value command) matches a version number of a $PK_{CA, PP}$ in the PSAM. If there is no match, the process is aborted.

14.7.2.2 The PSAM issues Get PIN Pad public key record commands to obtain certificate records from the PIN Pad. The PSAM must verify the certificates in sequence:

- The PIN Pad Creator certificate, using the $ALG_{CA,PP}$ and the $PK_{CA,PP}$ specified by the $VKP_{CA,PP.}$
- The PIN Pad certificate, using the ALG_{PPC} and the PK_{PPC} retrieved from the PIN Pad Creator certificate.

14.7.2.3 The general checks in Section 14.7.1 must be performed. If any of these fail then PIN Pad public key authentication has failed.

14.7.2.4 The PSAM must also check that:

1. The $ID_{PPCREATOR}$ retrieved from the PIN Pad Creator certificate is the same as that provided in the response to the Get Key Check Value command.
2. The Format code retrieved from the PIN Pad Creator certificate is equal to 'C2'.
3. $ID_{PPCREATOR}$ and ID_{PP} retrieved from the PIN Pad certificate are the same as those provided in the response to the Get Key Check Value command.
4. The Format code retrieved from the PIN Pad certificate is equal to 'C4'.
5. For each certificate, the month specified in the Certificate Expiration Date is equal to or later than the current month.

If any these checks fail, then the PIN Pad public

key has failed authentication.

14.7.3 Authentication of the PSAM Public Key

The PIN Pad contains one or more CA public keys ($PK_{CA,PSAM}$) for the purpose of authenticating the PSAM in the POS device. The PSAM must contain the necessary certificates for use with a $PK_{CA,PSAM}$ in the PIN Pad.

PIN Pad authentication of the PSAM public key consists of:

1. Determination by the PSAM that the PIN Pad can process a public key hierarchy in the PSAM.
2. Use of the Verify PSAM Public Key Certificate command to send certificates from the PSAM to the PIN Pad.
3. PIN Pad verification of the certificates.

14.7.3.1 After receiving and validating the response to the Get Key Check Value command (with KCVs not identical), the PSAM must verify that a version of the CA public key ($VKP_{CA,PSAM}$) in the response to the Get Key Check Value command matches a CA public key version under which the PSAM is certified. If there is no match, the process is aborted.

Certificates are provided to the PIN Pad using the Verify PSAM Public Key Certificate as shown in Section 7.2.3. The format of the certificate records is described in Section 14.7.9.

14.7.3.2 The PSAM must send and the PIN Pad must verify certificates in sequence:

- The Acquirer certificate, using the $ALG_{CA,PSAM}$ and the $PK_{CA,PSAM}$ specified by the $VKP_{CA,PSAM.}$
- The PSAM certificate, using the ALG_{ACQ} and the PK_{ACQ} retrieved from the Acquirer certificate.

14.7.3.3 The general checks in Section 14.7.1 must be performed. If any of these fail then PSAM public key authentication has failed.

14.7.3.4 The PIN Pad must also check that

1. $RID_{PSAMCREATOR}$ and $ID_{PSAMCREATOR}$ retrieved

from the PSAM certificate are the same as the $RID_{PSAMCREATOR}$ and $ID_{PSAMCREATOR}$ retrieved from the Acquirer Certificate, which in turn are the same as the $RID_{PSAMCREATOR}$ and $ID_{PSAMCREATOR}$ in the Get Key Check Value command.

2. The Format code retrieved from the Acquirer certificate is equal to 'A2'.

3. ID_{PSAM} retrieved from the PSAM certificate is the same as the ID_{PSAM} provided in the Get Key Check Value command.

4. The Format code retrieved from the PSAM certificate is equal to 'A4'.

5. For each certificate the last day of the month specified in the Certificate Expiration Date is equal to or later than the PIN Pad reference date (if it has one).

If any these checks fail, then the PSAM public key has failed authentication.

14.7.4 DES and Triple DES

DES and Triple DES are block ciphers standardized in FIPS PUB 46-3.

DES, denoted DES()[], operates on a 64-bit input block and a 64-bit key to produce a 64-bit output block. The number of effective key bits in a DES key is only 56 because every 8th bit of the 64-bit key takes on the value of a parity bit, thereby ensuring that there are an odd number of "1"s in each key byte.

Triple DES, denoted DES3()[], is implemented using three iterations of the DES block cipher with two independent DES keys K_1 and K_2.

Specifically, the cipher text Y of an 8-byte input block X is

$Y = DES3(K_1, K_2)[X] = DES(K_1)[DES^{-1}(K_2)[DES(K_1)[X]]]$.

Decryption is performed as

$X = DES3^{-1}(K_1, K_2)[Y] = DES^{-1}(K_1)[DES(K_2)[DES^{-1}(K_1)[Y]]]$

Note that for general encryption the padding and blocking process in Section 14.7.5 should be adhered to.

14.7.5 Encryption and Decryption

14.7.5.1 To encrypt any message, MSG, it must first be padded to the right with '80' and then with as many '00' bytes as necessary (possibly zero) until it is a multiple of 8 bytes:

X : = MSG||'80'||'00'||…||'00';

14.7.5.2 X is then divided into 8-byte blocks X_1, X_2, .., X_k and processed using Triple DES in Cipher Block Chaining mode:

Y_0 = '0000000000000000'

$Y_i = DES3(K_1,K_2)[X_i \oplus Y_{i-1}]$ for i = 1 to k

14.7.5.3 The encrypted message is

$Enc(K_1,K_2)[MSG] : = Y = Y_1 || \ldots || Y_k$

Note that this process always involves message padding so that when the message is an eight-byte PIN block the ciphertext will be 16-bytes long.

14.7.5.4 In order to decrypt a ciphertext message the encryption processed is merely reversed as shown below.

1. If the ciphertext is not a multiple of 8 bytes then decryption has failed.
2. Divide the ciphertext Y into 8 byte blocks:

   $Y : = Y_1 || \ldots || Y_k$
3. Compute the blocks X_i as follows:

   $X_i = DES3^{-1}(K_1,K_2)[Y_i] \oplus Y_{i-1}$ for i = 1 to k, where Y_0 = '0000000000000000'
4. Concatenate the blocks X_i to form
   $X : = X_1 || X_2 || \ldots || X_k$
5. Strip off all trailing zero bytes (possibly none) from X and then the final '80' byte to form MSG. If this last step is not possible then decryption has failed.
6. If all the preceding steps are successful then

$Dec(K_1,K_2)[Y] := MSG$

14.7.6 MAC computation

MACs are computed using DES in Cipher Block Chaining Mode with Triple DES applied to the last block.

The MAC computation is denoted by $MAC(K_1, K_2)[D]$. The computation conforms to ISO/IEC 9797-1:1999 Mechanism 3, using padding method 2 and DES as the block cipher. This is also described in EMV annex E1.2.

14.7.6.1 Input D to the MAC is first padded to the right with '80'. The result is then padded to the right with enough bytes of '00' (possibly none) to make the result a multiple of 8 bytes long.

X: = MSG||'80'||'00'||…||'00';

14.7.6.2 X is then divided into 8-byte blocks $X_1, X_2, .., X_k$ and processed using Single DES in Cipher Block Chaining mode:

Y_0 = '0000000000000000'

$Y_i = DES(K_1)[X_i \oplus Y_{i-1}]$ for i = 1 to k

14.7.6.3 Finally the 8-byte MAC is computed as

$MAC(K_1,K_2)[D] := DES(K_1)[DES^{-1}(K_2)[Y_k]]$

14.7.7 RSA Operations

14.7.7.1 All RSA operations must be performed as described in reference 6, EMV, annexes E and F.

The RSA encipher function corresponds to the Recover function defined in reference 6, EMV, Annex F.

The RSA decipher function corresponds to the Sign function defined in reference 6, EMV, Annex F.

14.7.8 RSA Padding

14.7.8.1 The process of RSA padding of data D of length 96 bytes (768 bits) to a length L bytes (where L $\geq$113) is as defined below.

1. Generate a 16 byte random number r and a 1-byte random number α whose most significant bit is forced to ”0”.

2. Form G(r) by concatenating

   SHA-1 (r || ‘00’) || SHA-1(r || ‘01’) || SHA-1(r || ‘02’) ||etc… until its length equals or exceeds L-17 bytes and then take the leftmost L-17 bytes as G(r).

3. Pad D to the left with (L-113) bytes of binary zeros. (D will now have a length of L-17 bytes).

4. Compute:

   PAD(D) := α || (D $\oplus$ G(r)) || (r $\oplus$ SHA (D $\oplus$ G(r), 16))

14.7.8.2 D is recovered from PAD(D) as follows:

1. Define PAD(D) := α || β || γ, where

   α is the first byte of PAD(D)
   β is the next L-17 bytes of PAD(D), and corresponds to D $\oplus$ G(r)
   γ is the next (and final) 16 bytes of PAD(D)

2. Skip the first byte, α;

3. Compute R:= SHA(β, 16) $\oplus$ γ

4. Compute G(R)

5. D is the rightmost 96 bytes of β $\oplus$ G(R)

14.7.9 Certificate Formats

14.7.9.1 The Acquirer Certificate must have the format defined in Table 171.

Table 171: Format of the Acquirer Certificate (PKC_{ACQ})

Field	Contents	Length
Header	Certificate Header '6A' - if there is an associated remainder field (PKR_{ACQ}), '4A' - if there is no associated remainder field	1
Format Code	Certificate Format ('A2')	1
RID_{PSAM}	RID of the PSAM Creator	5
$ID_{PSAMCREATOR}$	Identifies PSAM Creator / Acquirer	4
CED	Certificate expiration date (MMYY)	2
CSN_{ACQ}	Binary number unique to this certificate assigned by the certification authority (i.e. Primary Acquirer)	3
ALGH	Identifies the algorithm used to create the hash value. '01' indicates SHA-1.	1
ALG_{ACQ}	Identifies the algorithm used to verify the next lower level certificate	1
$LPKM_{ACQ}$	Length of the modulus of the acquirer public key	1
Filler	'00'	1
PKM_{ACQ}	Acquirer public key modulus or the leftmost bytes of the modulus. Padded to the right with 'BB' if the length of the modulus is less than $LPKM_{CA, PSAM}$ -41. If the length of the modulus is > $LPKM_{CA, PSAM}$ -41, the rightmost bytes (beginning in position $LPKM_{CA, PSAM}$ -40) are kept in PKR_{ACQ}.	$LPKM_{CA, PSAM}$ - 41
Hash Result	Hash of certificate data	20
Trailer	'BC'	1

14.7.9.2 The PSAM Certificate must have the format defined in Table 172.

Table 172: Format of the PSAM Certificate (PKC_{PSAM})

Field	Contents	Length
Header	Certificate Header '6A' - if there is an associated remainder field (PKR_{PSAM}), '4A' - if there is no associated remainder field	1

Field	Contents	Length
Format Code	Certificate Format ('A4')	1
RID_{PSAM}	RID of the PSAM Creator	5
$ID_{PSAMCREATOR}$	Identifies PSAM Creator / Acquirer	4
ID_{PSAM}	Identifier of the PSAM	4
CED	Certificate expiration date (MMYY)	2
CSN_{PSAM}	Binary number unique to this certificate assigned by the PSAM Creator	3
ALGH	Identifies the algorithm used to create the hash value. '01' indicates SHA-1, and is the only algorithm supported.	1
ALG_{PSAM}	Identifies the algorithm used to verify the dynamic signature created by the PSAM	1
$LPKM_{PSAM}$	Length of the modulus of the PSAM public key	1
Filler	'00'	1
PKM_{PSAM}	PSAM public key modulus or the leftmost bytes of the modulus. Padded to the right with 'BB' if the length of the modulus is less than $LPKM_{ACQ}$-45. If the length of the modulus is > $LPKM_{ACQ}$-45, the rightmost bytes (beginning in position $LPKM_{ACQ}$-44) are kept in PKR_{PSAM}.	$LPKM_{ACQ}$-45
Hash Result	Hash of certificate data	20
Trailer	'BC'	1

14.7.9.3 The PIN Pad Creator certificate must have the format defined in Table 173.

Table 173: Format of the PIN Pad Creator Certificate (PKC_{PPC})

Field	Contents	Length
Header	Certificate Header '6A' - if there is an associated remainder field (PKR_{PPC}), '4A' - if there is no associated remainder field	1
Format Code	Certificate Format ('C2')	1
$ID_{PPCREATOR}$	PIN Pad Creator ID	4

Field	Contents	Length
CED	Certificate expiration date (MMYY)	2
CSN_{PPC}	Binary number unique to this certificate assigned by the certification authority	3
ALGH	Identifies the algorithm used to create the hash value. '01' indicates SHA-1.	1
ALG_{PPC}	Identifies the algorithm used to verify the next lower level certificate	1
$LPKM_{PPC}$	Length of the modulus of the PIN Pad Creator public key	1
Filler	'00'	1
PKM_{PPC}	PIN Pad Creator public key modulus or the leftmost bytes of the modulus. Padded to the right with 'BB' if the length of the modulus is less than $LPKM_{CA, PP}$ - 36. If the length of the modulus is > $LPKM_{CA, PP}$ -36, the rightmost bytes (beginning in position $LPKM_{CA, PP}$ - 35) are kept in PKR_{PPC}.	$LPKM_{CA, PP}$ -36
Hash Result	Hash of certificate data	20
Trailer	'BC'	1

14.7.9.4 The PSAM Certificate must have the format defined in Table 174.

Table 174: Format of the PIN Pad Certificate (PKC_{PP})

Field	Contents	Length
Header	Certificate Header '6A' - if there is an associated remainder field (PKR_{PP}), '4A' - if there is no associated remainder field	1
Format Code	Certificate Format ('C4')	1
$ID_{PPCREATOR}$	Unique identifier of the PIN Pad Creator	4
ID_{PP}	PIN Pad Identifier	4
CED	Certificate expiration date (MMYY)	2

Field	Contents	Length
CSN_{PP}	Binary number unique to this certificate assigned by the certification authority (i.e. PIN Pad CA)	3
ALGH	Identifies the algorithm used to create the hash value. '01' indicates SHA-1, and is the only algorithm supported.	1
ALG_{PP}	Identifies the algorithm used to verify the dynamic signature created by the PIN Pad	1
$LPKM_{PP}$	Length of the modulus of the PIN Pad public key	1
Filler	'00'	1
PKM_{PP}	PIN Pad public key modulus or the leftmost bytes of the modulus. Padded to the right with 'BB' if the length of the modulus is less than $LPKM_{PPC}$ - 40. If the length of the modulus is > $LPKM_{PPC}$ - 40, the rightmost bytes (beginning in position $LPKM_{PPC}$ - 39) are kept in PKR_{PP}.	$LPKM_{PPC}$ - 40
Hash Result	Hash of certificate data	20
Trailer	'BC'	1

14.7.10 Expiration of Certificates

14.7.10.1 A certificate ceases to be valid after its Certificate Expiration Date. Acquirers must ensure that CA public keys are no longer used after their expiry date as dictated by the CA.

14.7.11 Replacement of Keys and Certificates

14.7.11.1 It must not be possible to change the PSAM and PIN Pad private keys (and associated public key certificates) after personalization.

Note that this may impact the minimum length of the PIN PAD Creator Public Key and the PIN Pad Public Key chosen.

14.7.12 Revocation of Certificates

The revocation of certificates is not described by this specification. If the acquirer's implementation permits certificate replacement, then that process may be used to replace revoked certificates.

14.7.13 Key Lengths

14.7.13.1 The minimum and maximum length of the public key modulus (LPKM) must be according to Table 175.

Table 175: Length of Public Key Modulus

	Minimum length (bits)	Maximum length (bits)
$LPKM_{CA, PSAM}$	1024	1952 (244 bytes)
$LPKM_{CA, PP}$	1024	1952 (244 bytes)
$LPKM_{ACQ}$	1024	1664 (208 bytes) < $LPKM_{CA, PSAM}$
$LPKM_{PPC}$	1024	1664 (208 bytes) < $LPKM_{CA, PP}$
$LPKM_{PSAM}$	1024	1536 (192 bytes) < $LPKM_{ACQ}$
$LPKM_{PP}$	1024	< $LPKM_{PPC}$

Note: Use of shorter RSA keys will limit the useful lifetime of the keys. Some implementations may require longer minimum key lengths than are specified here.

14.7.13.2 The key length (in bits) of the RSA moduli must always be an integer multiple of 16.

14.8 PIN Pad-less Secure Cryptographic Device

Terminals that do not support a PIN Pad, the keys protecting the exchange of data with the PSAMs need to be created in another Secure Cryptographic Device, e.g. a Secure Card Reader. This Secure Cryptographic Device may be created using the same PKC key management scheme as for a PIN Pad and must derive the keys necessary for protecting the data exchanged.

Other key management schemes may be used for the KEY_{CDP}. These are out of scope for this specification.

14.9 Response Codes

This section contains a summary of Response Codes that may be generated by various components of the terminal application.

Table 176: Summary of Response Codes

Response Code	Source Address	Description
Router		
'FFF0'	None	*Invalid Source Address*: the source address does not match the originator of the message.
'FFF1'	None	*Invalid Destination Address*: the message cannot be delivered because it contains an invalid destination address.
Common Handler Response Codes		
'0000'	Any	*Successful*
'FF34'	'0302'/'0304' '0402'/'0404'	*Unknown Message Code*
'FFF2'	Any	*Time-out*: the requested operation is valid, but some external event necessary for the proper execution failed to arrive in time.
'FFF3'	Any	*Handler Error*: generic message that an unspecified error has occurred.
'FFF4'	Any	*Handler must be initialized*: the Handler cannot perform the requested action until it has been initialized.
'FFF5'	Any	*Handler busy*: the Handler received the message but is unable to process it at this moment. The requesting Handler must try again later
'FFF6'	Any	*Insufficient resources*: the requested operation is valid, but insufficient resources exist to successfully execute the requested function.
'FFF7'	Any	*Handler must be opened*: the Handler is not in *open* status and therefore cannot perform the requested action.
'FFF8'	Any	*Handler is already open*
'FFF9'	Any	*Handler already closed*

Response Code	Source Address	Description
'FFFA'	Any	*Handler cannot be opened*: an error indicating that the Handler cannot be opened.
'FFFB'	Any	*Unsupported operation*: the Handler has received a command or an associated data set that was unrecognized or unsupported.
'FFFC'	Any	*Handler cannot be closed*: an error indicating that the Handler cannot be closed.
'FFFD'	Any	*Transaction interrupt request*: an interrupt indicating that the current transaction shall be terminated gracefully
'0XXX'	Any	Warning Codes: Reserved for Proprietary Use
'1XXX'	Any	Error Codes: Reserved for Proprietary Use
PSAM Handler (0)		
'FF23'	'0202'/'0203'	*Card did not respond*
'FF24'	'0202'/'0203'	*No card in reader*
'FF25'	'0202'/'0203'	*Unrecoverable Transmission error*
'FF26'	'0202'/'0203'	*Card buffer overflow*
'FF27'	'0202'/'0203'	*Unrecoverable Protocol error*
'FF28'	'0202'/'0203'	*Response has no status words*
'FF29'	'0202'/'0203'	Invalid buffer
'FF2A'	'0202'/'0203'	*Other card error*
Multi-Application Driver Handler (1)		
Card Handler (2)		

Response Code	Source Address	Description
'FF20'	'0201'	*Unrecoverable Transmission error between reader and magnetic stripe*
'FF21'	'0201'	*Output buffer overflow*
'FF22'	'0201'	*Write operation failed*
'FF23'	'0202'/'0203'	*Card did not respond*
'FF24'	'0202'/'0203'	*No card in reader*
'FF25'	'0202'/'0203'	*Unrecoverable Transmission error*
'FF26'	'0202'/'0203'	*Card buffer overflow*
'FF27'	'0202'/'0203'	*Unrecoverable Protocol error*
'FF28'	'0202'/'0203'	*Response has no status words*
'FF29'	'0202'/'0203'	*Invalid buffer*
'FF2A'	'0202'/'0203'	*Other card error*
'FF2B'	'0202'/'0203'	*Card partially in reader*
'FF82'	'0202'	*Authentication Error (MAC validation failed*
'FF87'	'0202'	*Secure Cryptographic Device not in PIN Entry State*
User Interface Handler (3)		
'FF30'	'0302'	*Out of border*
'FF31'	'0302'	*Printer out of paper*
'FF32'	'0302'	*Printer has signalled an error*
'FF33'	'0302'	*Printer does not appear to be connected and online*
'FF34'	'0302'/'0304'	*Unknown message code*
'FF35'	'0302'/'0304'	*Code Table not supported*
'FF80'	'0301'	*No KCV available, KSES not present*
'FF81'	'0301'	*Wrong PIN Pad ID*

Response Code	Source Address	Description
'FF82'	'0300'/'0301'/ '0302'	*Authentication Error (MAC validation failed)*
'FF83'	'0301'	*PSAM Identifier not recognized*
'FF84'	'0301'	*Parameters out of range*
'FF85'	'0301'	*Key Check values not identical, synchronization necessary*
'FF86'	'0301'	*PIN not available*
'FF87'	'0301'	*Secure Cryptographic Device not in PIN Entry State*
'FF88'	'0301'	*Termination Failed*
'FF89'	'0301'	*Record not found*
'FF8A'	'0301'	*Signature Error*
'FF8B'	'0301'	*Hash Error*
'FF8C'	'0301'	*Certificate Error*
'FF8D'	'0301'	*Hash algorithm not supported*
'FF8E'	'0301'	*PK Algorithm not supported*
'FF8F'	'0301'	*Hash result invalid*
'FF90'	'0301'	*RSA key mismatch. VKP not recognized*
'FF91'	'0301'	*Certificate format error*
'FF92'	'0301'	*Certificate expired*
'FF93'	'0301'	*Certificate ID mismatch*
Merchant Application Handler (4)		
'FF30'	'0402'	*Out of border*
'FF31'	'0402'	*Printer out of paper*
'FF32'	'0402'	*Printer has signalled an error*
'FF33'	'0402'	*Printer does not appear to be connected and online*

Response Code	Source Address	Description
'FF34'	'0402'/'0404'	*Unknown message code*
'FF35'	'0402'/'0404'	*Code Table not supported*
'FF40'	Any	*Invalid Currency*
'FF41'	Any	*Invalid Currency Exponent*
'FF42'	Any	*Invalid Transaction Results*
Data Store Handler (5)		
'FF50'	'0500'	*Invalid record pointer*. Record pointer outside the range defined for the current structure (Has not been added yet).
'FF51'	'0500'	*Invalid File ID*
'FF52'	'0500'	*Record too large*
'FF53'	'0500'	*Search key too large*
'FF54'	'0500'	*File creation error.*
'FF55'	'0500'	*File could not be accessed.*
'FF56'	'0500'	*File seek error.* A selected record could not be found.
'FF57'	'0500'	*File read error.*
'FF58'	'0500'	*File write error.*
'FF59'	'0500'	*Search key already existing*
Communication Handler (6)		
'FF60'	'0600'	*Invalid session setup parameters.*
'FF61'	'0600'	*No connection*
'FF62'	'0600'	*Connection in progress*
Event Handler (7)		
'FF72'	'0700'	*No Events in Queue*

Response Code	Source Address	Description
'FF73'	'0700'	*No Matching Events in Queue*

14.10 Message Codes

This section contains a list of Message codes that can be used to send pre-defined text messages to displays or to printers.

Table 177: Message Codes

Message Code	Text Message
01	(Amount)
02	(Amount) OK?
03	Approved
04	Call your Bank
05	Cancel or Enter
06	Card Error
07	Declined
08	Enter Amount
09	Enter PIN
0A	Incorrect PIN
0B	Insert Card
0C	Not Accepted
0D	PIN OK
0E	Please Wait
0F	Processing Error
10	Remove Card
11	Use Chip Reader
12	Use Mag Stripe

Message Code	Text Message
13	Try Again
14 – 3F	Reserved for future definition by EMV
40	System Error, Please Try Again
41	Invalid Card
42	Card out-of-order
43	Expired Card
44	Insufficient value
45	Card not present
46	Data Store full
47	Timed Out
48	Thank You
49	Not Available
4A	Print Receipt?
4B	Cancel
4C	Make Selection
4D	Incorrect Amount
4E	Welcome
4F	Signature
50	Application Menu
51	Transaction Menu
52	Purchase
53	Page
54	PIN Blocked
55	Enter new PIN
56	PIN Changed

Message Code	Text Message
57	PIN Unchanged
58	2 PINS not same
59	Confirm new PIN
5A	Change PIN
5B	Unblock PIN
5C	PIN not blocked
5D	PIN Unblocked
5E	Calling...
5F	Transmitting...
60	Receiving...
61	Comms Error
62	Disconnecting
63	Trans Log Upload
64	Retrying
65	Upload Done
66	Upload Failed
67	No Records
68	Debit :
69	Credit :
6A	Credit Reversal:
6B	Cash Load :
6C	Balance:
6D	New Balance:
6E	Specify Amount
6F	Recovery Needed

Message Code	Text Message
70	Insufficient Funds
71	Recovery Failed
72	Recovery Done
73	Money taken
74	Show Balance
75	Statement Review
76	by issuer
77	Upload Time
78	Start (HH:MM):
79	End (HH:MM):
7A	Prefix Nr
7B	Totals
7C	Auth X25 Nr
7D	Upload X25 Nr
7E	Nr Trials :
7F	Delay :
80	Onl Auth. Data
81	Onl Upload Data
82	Get cash
83	Unblock Appli.
84	Pre-Autho.
85	Pre Completion
86	Refund
87	Cancellation
88	D/C Menu

Message Code	Text Message
89	Precomp. Number
8A	GET Merchant PIN
8B	Data required in the database
8C	Interval (MM)
8D	Number Attempts
8E	Load Stop List
8F	Pick up Card
90	Denied:
91	View Balance?
92	Do not honour
93	Expired card
94	Suspected fraud
95	PIN exceeded
96	Refer Issuer
97	Not card number
98	Excessive amount
99	Counterfeit card
9A	Format error
9B	Card issuer or
9C	Switch inop.
9D	Bad Routing
9E	Sys malfunction
9F	Yes
A0	No
A1	Capture Card

Message Code	Text Message
A2	Money not taken
A3	Exp. date (YYMM)
A4	Enter PAN
A5	Enter Term ID
A6	Params Required
A7	Forced online
A8	Sale:
A9	Refund:
AA	Purse empty
AB	Set currency
AC	Currency changed
AD	Terminal id :
AE	Exceeds limit
AF	Invalid currency
B0-DF	Reserved for Future Use
E0-FF	Reserved for Proprietary Use

15. Data Elements

15.1.1 AID_N

Purpose: To indicate an AID that is supported by the PSAM, and which PSAM application is to be used to process transactions for that AID. Used during the Application Selection process.

Format: 5-16 bytes (binary).

Contents: RID || PIX where the RID is the five-byte global registered identifier as specified in ISO/IEC 7816-5 and the PIX (0-11 bytes) is at the scheme provider's discretion.

15.1.2 ALG

Purpose: To indicate the public key algorithm used in public key cryptography during the synchronization between a PIN Pad and a PSAM application. It also indicates the value of the public key exponent that is certified by this certificate.

Format: 1 byte (binary).

Content: See Table 178

Remarks: Used to identify the algorithm that is used to verify the next lower level certificate or signature. The subscript indicates the certificate containing the ALG.

Table 178: Coding of ALG

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	x	x	x				Value of public key exponent
0	0	0	0	1				2
0	0	0	1	0				3
1	0	0	0	0				$2^{16}+1$
All other values								RFU
					x	x	x	Usage of Public Key Algorithm
					0	0	1	- RSA Dynamic Authentication
					x	x	x	- (xxx ≠ 001) RFU

15.1.3 ALGH

Purpose: To indicate the algorithm used to produce a hash value in a public key certificate or signature.

Format: 1 byte (binary).

Content: '01' indicates SHA-1. SHA-1 is the only algorithm supported.

15.1.4 Amount Confirmed Indicator

Purpose: Indicates whether the amount has been confirmed.

Format: 1 byte (binary).

Content: '00' - the amount is confirmed
'01' - the amount has not been confirmed.
'02' – the transaction has been cancelled by the user

Remarks: Values other than '00', '01' and '02' are RFU.

15.1.5 Application Status Words (ASW1, ASW2)

Purpose: Indicates the results of a "process" command sent to the PSAM.

Format: 2 bytes (binary).

Content: As defined in section 10.3.2

15.1.6 ATR (Answer To Reset)

Purpose: Conveys the ATR returned by an IC card.

Format: Variable length, binary format

Content: As defined in ISO 7816-3 and EMV

15.1.7 [C-APDU]

Purpose: To hold an encrypted Card Command being sent to an ICC.

Format: Variable length

Content: A C-APDU encrypted under the current PIN session encryption key ($KSES_{PIN}$).

15.1.8 Card Command

Purpose: To convey a Command APDU being sent to an IC card.

Format: variable length

Content: A C-APDU as defined in reference 3, ISO/IEC 7816-4.

15.1.9 Card Response

Purpose: To convey a Response APDU from an IC card.

Format: variable length

Content: A complete R-APDU, including the Status Words, as defined in reference 3, ISO/IEC 7816-4.

15.1.10 CHALLENGE

Purpose: A number generated by the PIN Pad or PSAM, which allows each to authenticate that messages have been received from a valid device.

Format: 4 bytes (binary).

Content: Any non-repeating or random value.

Remarks: The subscript indicates whether the PIN Pad or the PSAM generated the Challenge.

15.1.11 CLA (Class byte)

Purpose: To form a command code.

Format: 1 byte (binary).

Remarks: CLA is the ISO assigned instruction class byte, which in conjunction with the INS field forms the command code. See ISO/IEC 7816-4 for a discussion of the Class byte.

15.1.12 CNT_{AID}

Purpose: To indicate the number of AIDs being returned in the response to a Get Supported AIDs command.

Format: 1 byte (binary).

Content: 8-bit value, coded as an unsigned integer.

Remarks: Used with the command "Get Supported AIDs".

15.1.13 $CNT_{SUBADDRESS}$

Purpose: To indicate the number of sub-addresses being returned in the response to the Get Handler Addresses command

Format: 1 byte (binary).

Content: Holds the number of sub-addresses returned.

Remarks: Used with the command Get Handler Addresses

15.1.14 Code Table Index

Purpose: Identifies the character set in which display, printer or key-entered data is coded.

Format: BCD, 1 byte

Content: '00' indicates the Command Character Set defined in reference 6, EMV Annex C.

'nn' indicates a code table as defined in reference 8, ISO 8859.

15.1.15 CSN (Certificate Serial Number)

Purpose: Unique number assigned to the certificate by the creator of the certificate.

Format: 3 bytes (binary).

Content: A unique number for a certificate.

15.1.16 CURR (Currency)

Purpose: Identifies the currency for a transaction

Format: BCD, 3 bytes in the form '0c cc 0e', where ccc is the code assigned to the currency by ISO 4217, and e is the exponent.

Content: CURR contains both the currency code (CURRC) and the exponent (CURRE)

15.1.17 CURRC (Currency Code)

Purpose: Identifies the currency for a transaction

Format: BCD, 2 bytes in the form '0c cc', where ccc is the code assigned to the currency by ISO 4217.

Content: CURRC contains only the currency code

15.1.18 CURRE (Currency Exponent)

Purpose: Identifies the exponent of the currency of a transaction

Format: BCD, 1 bytes in the form '0e'

Content: CURRE contains only the currency exponent

15.1.19 Destination Address (DAD)

Purpose: To identify the device to which a terminal message must be delivered.

Format: 2 bytes (binary).

Content: A handler address.

15.1.20 DS (Digital Signature)

Purpose: A digital signature created by the PSAM to allow the PIN Pad to authenticate the PSAM during synchronization

Format: Variable length (binary) (The length of the PSAM public key modulus ($LPKM_{PSAM}$) determines the length of DS.

Content: See section 7.2.4

15.1.21 $DTHR_{PDA}$ (Transaction date and time)

Purpose: To indicate a date and time.

Format: 5 bytes (BCD).

Content: 10 BCD digits: YYMMDDHHMM.

15.1.22 Enc($KSES_{PIN}$)[PIN]

Purpose: To hold an encrypted PIN block.

Format: 16 bytes (binary).

Content: Encrypted PIN block.

Remarks: See Table 83 for the format of the PIN block.

15.1.23 Error Response Data

Purpose: To hold application specific response data.

Format: Variable length.

Content: Application-specific error response data.

15.1.24 Event Type Code

Purpose: To indicate the type of event.

Format: 1 byte (binary).

Content: See Table 157.

15.1.25 Event Location

Purpose: To indicate the handler that posted an event.

Format: 2 bytes (binary).

Content: A handler address.

15.1.26 File Identifier (ID_{FILE})

Purpose: Identifies a particular file that has been created in the data store.

Format: 2 bytes (binary).

Content: The unique identifier of a file.

Remarks A value of zeros (‘0000’) is not valid

15.1.27 Filler

Purpose: Used to fill out a record or block

Format: 1 byte (binary).

15.1.28 Format Code

Purpose: To indicate the type of certificate.

Format: 1 byte (binary).

Content: 8-bit value, unsigned integer.

Remarks: Examples of Format Codes: 'C2' = PKC_{PPC}, 'C4' = PKC_{PP}, 'A2' = PKC_{ACQ}, 'A4' = PKC_{PSAM}

15.1.29 Handler Category Address

Purpose: Indicates the address of a Handler Category

Format: 1 byte (binary).

Content: Each device handler category is assigned a one-byte address. Individual handlers are located by their handler category address, followed by a sub-handler address.

15.1.30 Handler Sub-Address

Purpose: In combination with the Handler category address, identifies a particular handler.

Format: 1 byte (binary)

Content: Each device handler is assigned a two-byte address. Individual handlers are located by their handler category address, followed by a sub-handler address.

15.1.31 Historical Bytes

Purpose: The historical bytes are an optional element in the Answer-to-Reset from an IC card. The historical bytes designate general information, for example, the card manufacturer, the chip inserted in the card, the masked ROM in the chip, the life-cycle state of the card.

Format: 0-15 bytes (binary).

Remarks: The contents of the historical bytes are at the discretion of the card issuer. See reference 3, ISO/IEC 7816-4 for additional information.

15.1.32 ID_{PP} (PIN Pad ID)

Purpose: To identify a PIN Pad

Format: 4 bytes (binary).

Content: Serial number of the PIN Pad assigned by the PIN Pad Creator.

Remarks: Assigned by the entity identified by $ID_{PPCREATOR}$. With the $ID_{PPCREATOR}$ identifies a PIN Pad within a POS device.

15.1.33 $ID_{PPCREATOR}$ (Identifier for the Creator of a PIN Pad)

Purpose: To identify the system or entity that generates personalization data for PIN Pads and assigns the ID_{PP}.

Format: 4 bytes (binary).

Content: Number assigned to the PIN Pad Creator.

Remarks: Assigned by the primary acquirer. With the ID_{PP} identifies a PIN Pad within a POS device.

15.1.34 ID_{PSAM} (Identifier for a PSAM)

Purpose: To identify a PSAM.

Format: 4 bytes (binary).

Content: Serial number of the PSAM assigned by the PSAM Creator.

Remarks: Assigned by the entity identified by $ID_{PSAMCREATOR}$. With the RID_{PSAM} and the $ID_{PSAMCREATOR}$ uniquely identifies a PSAM.

15.1.35 $ID_{PSAMAPP}$ (TAPA PSAM Application Identifier)

Purpose: To identify a particular PSAM application

Format: 2 bytes (binary).

Content:

- The first nibble is coded as follows:‘0’-‘7’ = application specification defined by the international payment schemes.
  - ‘0’ = CEP application, ‘1’-‘7’ = RFU
- ‘8’-‘F’ = proprietary application specification.

15.1.36 $ID_{PSAMCREATOR}$ (Identifier for the Creator of the PSAM)

Purpose: To identify the system or entity which generates personalization data for PSAMs and assigns the ID_{PSAM}.

Format: 4 bytes (binary).

Remarks: Assigned by the owner of the RID_{PSAM}. With the RID_{PSAM}, uniquely identifies the entity creating a PSAM.

15.1.37 ID_{SCHEME} (Acquirer reference number)

Purpose: Used to identify a particular AID or scheme supported by the Acquirer

Format: 1 byte (binary).

Remarks: Assigned by the Acquirer or processor.

15.1.38 INS (Instruction byte)

Purpose: To identify a command.

Format: 1 byte.

Remarks: Used in conjunction with CLA to identify a command. See ISO/IEC 7816-4.

15.1.39 KCV (Key Check Value)

Purpose: To verify the status of the session key shared between a PSAM and a PIN Pad.

Format: 3 bytes (binary).

Content: The 3 most significant bytes of the result of a triple-DES encryption of an 8-byte block of binary zeros.

Remarks: The subscript indicates whether the PSAM or the PIN Pad computed the KCV.

15.1.40 KEK_{CDP}

Purpose: A key encryption key used to protect the master session key used during transfer from a secure Card Reader to the PSAM. Part of an independent key chain used for Cardholder Data Protection (CDP) between the PSAM and a Secure Cryptographic Device in a Terminal without a PIN Pad. The initial value of the key is loaded into the PSAM during configuration.

Format: 16 bytes (binary).

15.1.41 Key Data

Purpose: The data to be used as a unique key for a data store record.

Format: LEN_{SKEY} bytes (binary).

Content: Any.

Remarks: Used when searching, adding and deleting data in records in files.

15.1.42 KEY_{CDP}

Purpose: A master session key used to derive session keys that are used for Cardholder Data enciphering and MACs between a Secure Cryptographic Device and the PSAM.

Format: 16 bytes (binary).

15.1.43 KSES

Purpose: A master session key used to derive session keys that are used for PIN encryption and MACs between the PIN Pad and the PSAM.

Format: 16 bytes (binary).

15.1.44 $KSES_{CDP}$

Purpose: To encipher Cardholder Data when transferred between the PSAM and the Terminal.

Format: 16 bytes (binary).

Remarks: Generated by the PSAM based on the KEY_{CDP} delivered from a Secure Cryptographic Device.

15.1.45 $KSES_{INIT}$

Purpose: To provide an initial key which is used to derive the first session key.

Format: 16 bytes (binary).

Remarks: Generated by the PSAM during synchronization with the PIN Pad.

15.1.46 $KSES_{MAC}$

Purpose: A session key used to authenticate messages exchanged between the PSAM and the PIN Pad.

Format: 16 bytes (binary).

Remarks: Derived from the current master session key (KSES).

15.1.47 $KSES_{PIN}$

Purpose: A session key used to encrypt PIN data being exchanged between the PSAM and the PIN Pad.

Format: 16 bytes (binary).

Remarks: Derived from the current master session key (KSES).

15.1.48 L_c (Data length)

Purpose: To indicate the number of bytes present in the data field of a command.

Format: 1 byte (binary).

Content: 8-bit value, coded as an unsigned integer.

Remarks: Omitted if no data is sent in a command. See ISO/IEC 7816-4.

15.1.49 L_e (Expected data length)

Purpose: To indicate the maximum number of bytes expected in a response to a command.

Format: 1 byte (binary).

Content: 8-bit value, coded as an unsigned integer.

Remarks: A value of '00' indicates that the card must return all available data. Please see ISO/IEC 7816-4.

15.1.50 L_{DATA} (Data field length)

Purpose: To indicate the length of a data field in a Terminal Message.

Format: 2 bytes (binary).

Remarks Message data lengths of at least 512 bytes must be supported. Terminal applications must only rely on the ability to send longer messages in a proprietary environment.

15.1.51 LEN

Purpose: To indicate a length.

Format: 2 bytes (binary).

Content: 16-bit value, unsigned.

Remarks: Used to indicate a number of bytes to read.

15.1.52 $LEN_{AID,N}$

Purpose: To indicate the length of the N'th AID in the response to the Get Supported AIDs PSAM command.

Format: 1 byte (binary).

Content: 8-bit value, coded as an unsigned integer.

15.1.53 LEN_{REC}

Purpose: To indicate the length of a record.

Format: 2 bytes (binary).

Content: 16-bit value, coded as an unsigned integer.

Remarks: If the LEN_{REC} is coded as '0000' in an Add Record message, the maximum record size must be reserved. The actual record size is maintained as '0000' until a subsequent "Update Record" message

15.1.54 LEN_{SKEY}

Purpose: Indicates the length of the search key assigned to a file in the data store

Format: 1 byte (binary).

Content:

Remarks:

15.1.55 Length

Purpose: To indicate a length.

Format: 1 byte (binary).

Content: 8-bit value, coded as an unsigned integer.

15.1.56 LPKE (Length of a Public Key Exponent)

Purpose: To indicate the length of a Public Key Exponent.

Format: 1 byte (binary).

Remarks: The subscript indicates the exponent being referred to.

15.1.57 LPKM (Length of Public Key Modulus)

Purpose: To indicate the length in bytes of a Public Key Modulus.

Format: 1 byte (binary).

Remarks The subscript indicates the modulus being referred to.

15.1.58 MAC

Purpose: A MAC providing authentication of a data exchange between a PSAM and PIN Pad.

Format: 8 byte (binary).

Remarks: The subscript indicates the particular message in which the MAC is located. The data being MAC'ed is specified in each applicable message.

15.1.59 Magnetic Stripe Data

Purpose: To hold a set of data from a track.

Format: Variable

Content: Any.

Remarks: Includes u, Len and track Data.

15.1.60 Message Code

Purpose: To indicate a pre-defined message to be displayed or printed

Format: 1 byte (binary).

Content: See Table 177

15.1.61 Message Data

Purpose: The data portion of a Terminal message.

Format: Variable

Content: Specific to the Message Type

15.1.62 Message Type

Purpose: To identify the type of message.

Format: 1 byte (binary).

Content: SeeTable 4, Table 5 and Table 6.

Remarks: If Message Type is 'FF' the message is a response, any other value indicates the message type of a command.

15.1.63 NUMFILE

Purpose: To indicate the number of files to create.

Format: 1 byte (binary).

Content: 8-bit value, coded as an unsigned integer.

15.1.64 Pad Pattern

Purpose: Padding bytes required in a public key certificate or signature

Format: variable length (binary).

Content: Successive bytes containing 'BB'

15.1.65 PK (Public Key)

Purpose: A public key is used by an entity to verify a certificate or signature created by the owner of the public key. The public key of a PIN Pad is used by the PSAM for the purpose of encrypting messages to the PIN Pad.

Format: Variable length (binary).

Remarks: The subscript indicates the entity to which the public key belongs. A public key consists of the modulus (PKM) and the exponent (indicated in the ALG field).

15.1.66 PKC (Public Key Certificate)

Purpose: A public key certificate is created by the next higher entity in the certificate hierarchy.

Format: Variable length (binary). The length is the same of the length of the signing public key modulus.

Remarks: The subscript indicates the entity to which the certificate belongs.

15.1.67 PKM (Public Key Modulus)

Purpose: A public key modulus is a component of a public key.

Format: Variable length (binary).

Remarks: The subscript indicates the entity to which the modulus belongs.

15.1.68 PKR (Public Key Remainder)

Purpose: Contains the rightmost bytes of the Public Key Modulus when the entire modulus will not fit into the public key certificate.

Format: Variable length (binary).

Remarks: The subscript indicates the associated public key.

15.1.69 P1, P2 (Parameter bytes)

Purpose: To set up parameters for a PSAM command

Format: 2 bytes.

Remarks: See ISO/IEC 7816-4.

15.1.70 Pointer Orientation

Purpose: Indicates the starting location for a Get File Record, and “next record” pointer that must be returned.

Format: 1 byte (binary).

Content: ‘01’-‘03’, See Table 131

Remarks: Values other than ‘01’-‘03’ are reserved for future use

15.1.71 Message Code

Purpose: To identify a predefined text for display or printing.

Format: 1 byte (binary).

Content: See Table 177

Remarks: Used by the "Print Message" and Display commands.

15.1.72 PIN Pad Identifier

Purpose: Unique PIN Pad identifier.

Format: 8 bytes (binary).

Content: $ID_{PPCREATOR}$ || ID_{PP}.

15.1.73 PS

Purpose: A digital signature used to provide mutual authentication between a PSAM and a PIN Pad, and to exchange the Initial Session Key ($KSES_{INIT}$).

Format: Variable, binary

Content: Encrypted digital signature of the PSAM. See 7.2.4

15.1.74 PSAM Identifier

Purpose: Unique PSAM identifier.

Format: 13 bytes (binary).

Content: RID_{PSAM} || $ID_{PSAMCREATOR}$ || ID_{PSAM}.

15.1.75 PSAM sub-address

Purpose: Identifies the Handler sub-address assigned to the PSAM.

Format: 1 byte (binary).

Content: The PSAM sub-address.

15.1.76 Record Data

Purpose: The data stored in a record in a file.

Format: LEN_{REC} bytes.

Content: Data contained in a record in a given file.

15.1.77 Record Pointer

Purpose: To identify a record in a file.

Format: 2 bytes (binary).

Content: 16-bit value, unsigned.

Remarks A value of zeros (‘0000’) is not valid.
Once the record pointer has been assigned, it cannot be changed.

15.1.78 Record Tag

Purpose: Indicates the contents of a record read from the PIN Pad

Format: 1 byte.

Content: ‘85’

15.1.79 Response Code (RC)

Purpose: The Handler or Device uses the Response Code to indicate a problem handling a Terminal Message command.

Format: 2 bytes (binary).

Content: See Table 176

15.1.80 Response Data

Purpose: Contains the data being returned in response to a card or PSAM command.

Format: Variable

Content: The contents are specific to the command being responded to.

15.1.81 Returned String

Purpose: A string of data returned in response to a Read String message.

Format: Len bytes (binary).

15.1.82 *RID_{PSAM} (Registered Identifier Of The Entity Assigning PSAM Creator Ids)*

Purpose: To make the identifier of a PSAM Creator unique.

Format: 5 bytes (binary).

Remarks: The identifier of the entity that assigns identifiers to certified PSAM Creators ($ID_{PSAMCREATOR}$), assigned as specified in ISO/IEC 7816-5.

15.1.83 Search Type

Purpose: To allow the application to search for a particular type of event (for example, card inserted), and/or an event that occurred at a specific device (for example, the customer keypad).

Format: 1 byte (binary).

Contents See Table 179.

Table 179: Search Type Coding

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	X	x	x	x	x			RFU
						1		Search by Event Type
							1	Search by Event Location

15.1.84 Session Data

Purpose: To hold data necessary to initiate a communication session.

Format: Var.

Content: Any.

15.1.85 SK (Private Key)

Purpose: An asymmetric private key used by a PSAM for the purpose of generating signatures and by a PIN Pad for the purpose of decryption.

Format: Variable length, binary

Remarks: The subscript identifies the entity to which the key belongs

15.1.86 Source Address (SAD)

Purpose: To identify the source of a given command or response.

Format: 2 bytes (binary).

Content: A handler address.

15.1.87 Status Words (SW1, SW2)

Purpose: To indicate the result of a command sent to an ICC.

Format: 2 bytes (binary).

Content: See ISO/IEC 7816-4.

Remarks: A value of ’96 01’ is used by the PSAM to indicate that additional response data is available. The additional data is retrieved with a Get Next command.

15.1.88 ID_{THREAD} (Thread Identifier)

Purpose: To identify a particular thread in a multi-threading terminal

Format: 1 byte (binary).

Content: The Thread Identifier can have any value in the range '00' - 'FF'.

Remarks: The MAD Handler assigns the Thread Identifier. A value may be re-used when a thread is completed.

15.1.89 Time

Purpose: To specify a time-out value.

Format: 4 bytes (binary).

Content: The time-out value in milliseconds.

Remarks: Time indicates the maximum time after which either data or an error response must be returned

15.1.90 Timer Flag

Purpose: To indicate that a time-out value is specified.

Format: 1 byte (binary).

Content: '00' - the message is not timed
'80' - the message is timed.

15.1.91 Track Data

Purpose: To hold track data from a magnetic stripe.

Format: Variable

Content: Depends on the track being read

15.1.92 Transaction Amount

Purpose: To indicate the transaction amount.

Format: 4 bytes (binary).

Content: The transaction amount is unsigned.

Remarks: The value represents the lowest denominator for the corresponding Currency Code, e.g. for USD, amounts are represented in 1/100 USD units, i.e. cents.

15.1.93 Transaction Results

Purpose: To indicate to the Merchant Application Handler the results of a transaction, thus allowing the merchant equipment to dispense goods or take other action as required.

Format: 1 byte (binary).

Content: '00' - the transaction was successful

'01' - the transaction failed.

15.1.94 u

Purpose: To identify the track(s) on the magnetic stripe to be read.

Format: 1 byte (binary).

Content: The ISO identifier of the track to be read.

Remarks: u can have the values '01', '02', '03', '0C', '0D', '17' and '7B' representing track ISO 1, ISO 2, ISO 3, ISO 1&2, ISO 1&3, ISO 2&3 and ISO 1&2&3. The highest bit is set if the track data to be returned are to be enciphered.

15.1.95 $VKP_{CA, xx}$

Purpose: Indicates the version of the CA public key used to produced the PIN Pad Creator or PSAM Creator certificate

Format: 1 byte (binary).

Remarks: The subscript indicates which CA key is referenced

16. Acronyms

Acronym or Data Element	Description
AID	Application Identifier
APDU	Application Protocol Data Unit
ASW	Application Status Words
ATR	Answer-to-Reset
BCD	Binary Coded Decimal
CA	Certification Authority
CDP	Cardholder Data Protection
CEPS	Common Electronic Purse Specifications
CLA	Class Byte of a Command APDU
DES	Data Encryption Standard
DES3	Triple DES
DS	Digital Signature
EMV	Europay, MasterCard and Visa
ICC	Integrated Circuit Card
IEC	International Electrotechnical Commission
INS	Instruction Byte of the Command Message
ISO	International Organization for Standardization
KCV	Key Check Value
L_c	Length of Command Data Field
L_e	Expected Length of the Response Data Field
MAC	Message Authentication Code
MAD	Multi-Application Driver
MT	Message Type
P1	Parameter1
P2	Parameter2
PED	PIN Entry Device (PIN pad)
PIN	Personal Identification Number
PK	Public Key
PS	Public-key digital signature
PSAM	Purchase Secure Application Module

Acronym or Data Element	Description
RC	Response Code
RFU	Reserved for Future Use
RID_{PSAM}	Registered identifier of the entity assigning PSAM Creator IDs
RSA	Rivest, Shamir and Adleman (Cryptographic Algorithm)
SCD	Secure Cryptographic Device
SHA	Secure Hash Algorithm
SK	Private Key
SW1-SW2	Status Words
TAPA	Terminal Architecture for PSAM Applications
TED	Tamper Evident Device
var	Variable