EMV®
Contactless Specifications for Payment Systems

Book C-6

Kernel 6 Specification

Version 2.10
March 2021

Legal Notice

Revision Log – Version 2.10

This Revision Log applies to:

EMV Contactless Specifications for Payment Systems, Book C-6, Kernel 6 Specification, Version 2.9, March 2020 [Book C-6]

The following changes have been made to Book C-6 since the publication of Version 2.9.

1. Introduced Data Storage, Extended Logging, Tearing Recovery features.
2. Removed support for MS Mode and Legacy MS Mode

Contents

Figures

Tables

[This page is intentionally left blank.]

1 Introduction

This *EMV Contactless Specifications for Payment Systems, Book C-6, Kernel 6 Specification* (this Specification) addresses the requirements applicable to Kernel 6, including the:

- Overview of Kernel 6 Approach
- Terminal Transaction Processing Steps and
- Application Protocol Data Unit (APDU) Command Description

Baseline functionalities are addressed in [EMV Book B], which should be read as a companion document and is hereby incorporated into this specification by reference. These baseline functionalities include, but are not limited to:

- Preliminary Transaction Processing (i.e., Pre-Processing)
- Protocol Activation
- Combination Selection
- Kernel Activation
- Outcome Processing and
- Data Element Processing.

1.1 Scope

This Specification describes one of several Kernels defined for use with Entry Point.

1.2 Audience

This Specification is intended for use by system designers in payment systems and financial institution staff responsible for implementing financial applications.
Note: Throughout this Specification, Contactless Cards and Consumer Devices (such as mobiles) are referenced as Contactless Cards unless Contactless and Consumer Devices must be addressed separately.

1.3 Volumes of Contactless Specifications

This Specification is part of an EMVCo ten-volume set, including:

- Book A: Architecture and General Requirements [EMV Book A]
- Book B: Entry Point Specification [EMV Book B]
- Book C-1: Kernel 1 Specification
- Book C-2: Kernel 2 Specification
- Book C-3: Kernel 3 Specification
- Book C-4: Kernel 4 Specification
- Book C-5: Kernel 5 Specification
- Book C-6: Kernel 6 Specification
- Book C-7: Kernel 7 Specification
- Book D: Contactless Communication Protocol [EMV Book D]

1.4 Reference Specifications

In addition to the volumes cited in section 1.3, the following specifications and standards contain provisions that are referenced in this Specification. The latest version shall apply unless a publication date is explicitly stated.
If any provision or definition in this Specification differs from those in the listed specifications and standards, the provision or definition herein shall take precedence.

Abbreviation	Title
[EMV Book 1]	EMV Integrated Circuit Card Specifications for Payment Systems. Book 1 - Application Independent ICC to Terminal Interface Requirements.
[EMV Book 2]	EMV Integrated Circuit Card Specifications for Payment Systems. Book 2 - Security and Key Management.
[EMV Book 3]	EMV Integrated Circuit Card Specifications for Payment Systems. Book 3 - Application Specification.
[EMV Book 4]	EMV Integrated Circuit Card Specifications for Payment Systems. Book 4 - Cardholder, Attendant, and Acquirer Interface Requirements.
[EMV Book A]	EMV Contactless Specifications for Payment Systems: Book A Architecture and General Requirements
[EMV Book B]	EMV Contactless Specifications for Payment Systems: Book B Entry Point Specification
[ISO/IEC 639-1]	Codes for the representation of names of languages – Part 1: Alpha-2 Code.
[ISO/IEC 3166-1]	Codes for the representation of names of countries and their subdivisions Part 1 Country Codes.
[ISO/IEC 4217]	Codes for the representation of currencies and funds.
[ISO/IEC 7810]	Information technology – Identification cards – Physical characteristics.

Abbreviation	Title
[ISO/IEC 7813]	Information technology – Identification cards – Financial transaction cards.
[ISO/IEC 7816-4]	Information technology – Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange.
[ISO/IEC 7816-5]	Identification cards – Integrated circuit cards – Part 5: Registration of application providers.
[ISO/IEC 8583-1]	Financial transaction card originated messages – Interchange message specifications – Part 1: Messages, data elements and code values.
[ISO/IEC 8825-1]	Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).
[ISO/IEC 8859-1]	Information Technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1.
[ISO/IEC 9797-1]	Information technology – Security techniques - Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher.
[ISO/IEC 10116]	Information technology – Security techniques - modes of operation for an n-bit block cipher.
[ISO/IEC 13239]	Information technology – Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures.
[ISO/IEC 14443-1]	Identification cards – Contactless integrated circuit cards – Proximity cards – Part 1: Physical characteristics.
[ISO/IEC 14443-2]	Identification cards – Contactless integrated circuit cards – Proximity cards – Part 2: Radio frequency power and signal interface.
[ISO/IEC 14443-3]	Identification cards – Contactless integrated circuit cards – Proximity cards – Part 3: Initialization and anti-collision.
[ISO/IEC 14443-4]	Identification cards – Contactless integrated circuit cards – Proximity cards – Part 4: Transmission protocol.

1.5 Overview

This volume includes the following chapters and annexes:

- **Chapter 1:** contains general information regarding this Specification.
- **Chapter 2:** provides a high-level description of the Kernel 6 approach, including configuration, transaction processing flows, terminal and configuration requirements and options, and supported APDU commands.
- **Chapter 3:** specifies detailed transaction processing flows for Kernel 6.
- **Chapter 4:** lists and describes the APDU commands used by Kernel 6.
- **Annex A:** is a glossary of terms and abbreviations used in this specification.
- **Annex B:** contains Kernel 6 Transaction Parameters and Outcomes.
- **Annex C:** lists data elements required for offline only, online only, and offline / online transactions.
- **Annex D:** is a dictionary of data elements for Kernel 6.
- **Annex E:** specifies the Track 2 Equivalent Data.
- **Annex F:** provides guidance for Data Storage security.

1.6 Data Conventions

The data conventions included in Table 1-1 are used throughout this Specification.

Table 1-1: Data Conventions

Convention	Description
'…'	Single quotation marks surround a decimal or hexadecimal digit. For example '3A' or '9901'
[…]	Squared brackets surround a reference document. For example: [EMV Book A].
A	Alphanumeric
b	Binary
n	Numeric
Var	Variable length

For data elements that have multiple bytes, the first byte or byte 1 is the leftmost byte, while the last byte is the rightmost byte.

1.7 Terminology

The terminology used in this specification is presented in Table 1-2.

Table 1-2: Terminology

Term	**Definition**
Shall, must, or required	Denotes a mandatory requirement
Should	Denotes a recommendation
May or optional	Denotes an optional feature

[This page is intentionally left blank.]

2 Overview of Kernel 6 Approach

A high-level description of the Kernel 6 approach is provided in the following sections:

2.1 Configuration

The Kernel 6 approach supports only EMV transactions and it allows the processing of transactions offline or online (based on Terminal capabilities).

Access to configurations supporting EMV transactions are enabled by a designated Application Identifier (AID) of ‘A0000001523010’.

2.1.1 EMV Transactions (Offline, Online or Deferred Authorization Transactions)

EMV transactions conform to the EMV requirements supporting offline and online capabilities including the functionalities and commands for:

- Selecting the application, initializing transaction processing, and reading records to obtain the application data items
- Performing Cardholder Verification and Offline Data Authentication (ODA) for offline transactions
- Enabling Processing Restrictions, Terminal Action Analysis, Online Processing, and Issuer Update Processing for online transactions, and
- Supporting security parameters, including an optimized cryptographic process that enables the generation of an Application Cryptogram (AC) or Combined Dynamic Data Authentication / Application Cryptogram Generation (CDA) using the Unpredictable Number (UN) provided by the reader in the Processing Data Objects List (PDOL).

Kernel 6 may optionally be configured to support Deferred Authorization for EMV transactions in environments where real time online authorization is not possible.

Kernel 6 supports optional capabilities for Data Storage, Extended Logging and Tearing Recovery, where supported by the card.

Unlike ***contact*** EMV-based transactions, the reader performs certain checks before the card enters the Radio Frequency (RF) field as specified in [EMV Book B] and after the card has left the RF field. Additional information regarding EMV transaction processing is provided in Section 3 of this Specification.

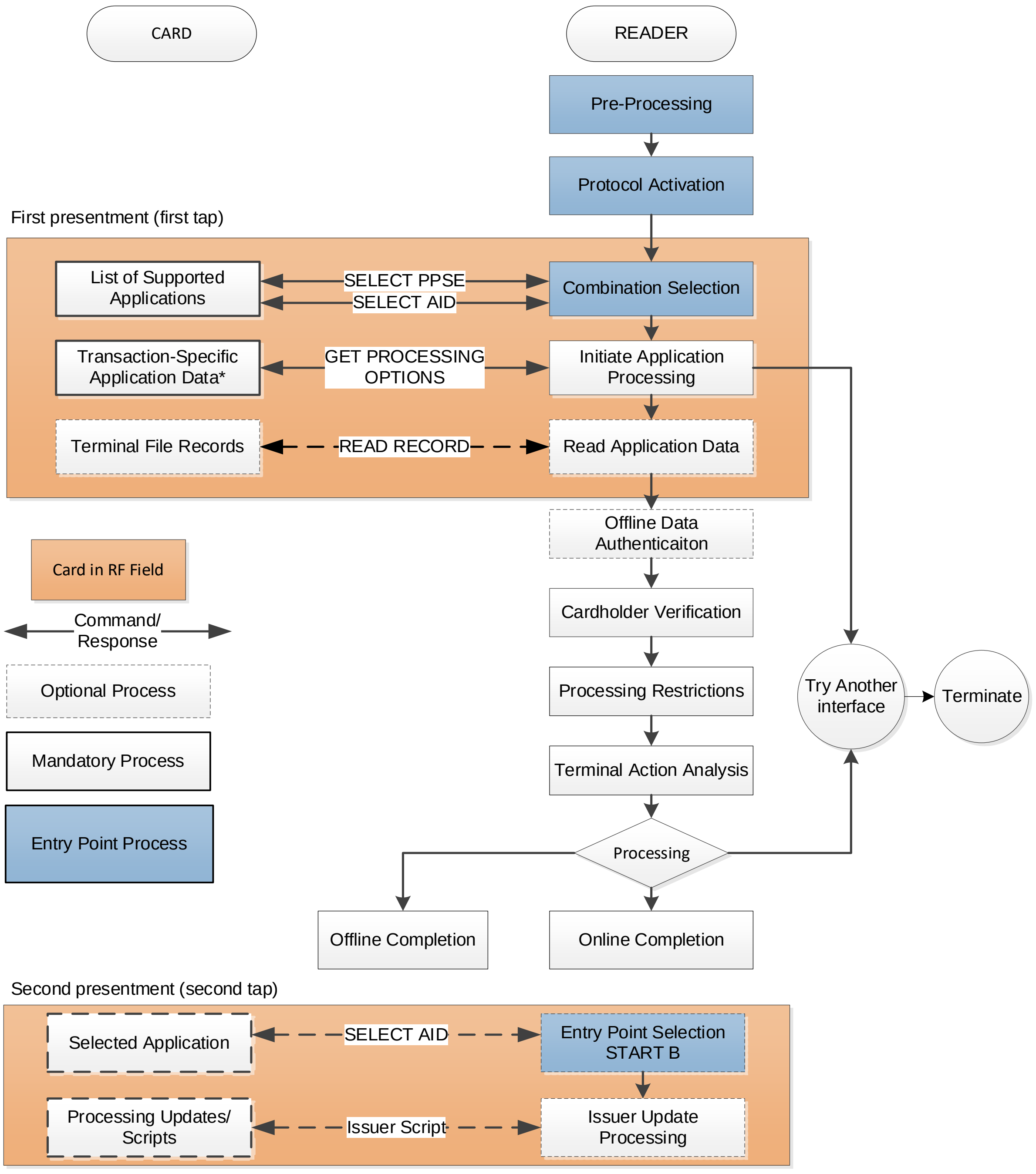
2.2 EMV Transaction Processing Flows

The Kernel will initiate the EMV transaction when EMV is supported by the card and the transaction passes the required PDOL checks.
Terminal entry point shall perform the pre-processing prior to the activation of the contactless interface of the reader and before the cardholder is invited to present a Contactless Card. Combination Selection shall be processed after card presentment. Because the interaction (i.e., the command-response) between the card and the Terminal can only occur when the card is in the RF field, the process requires adaptations to permit the:

- Reduction of the number of commands and adjustment of the transaction flow for both the first and second presentment processes
- Definition of data elements taking into account events relevant for contactless transactions
- Usage of Entry Point as defined for [EMV Book B], to select the application and activate the corresponding Contactless Kernel 6
- Optional identification of and recovery for torn contactless transactions
- Optional retrieval of data items from on-card Data Storage before transaction initiation
- Optional writing of data items to on-card Data Storage or Extended Logging during transaction initiation
- Generation of a CDA Cryptogram or an Application Cryptogram during the transaction initiation
- Execution of Terminal Risk Management before the card is placed in the field and after the card is removed from the field, and
- If required, execution of cardholder verification after the card is removed from the field.

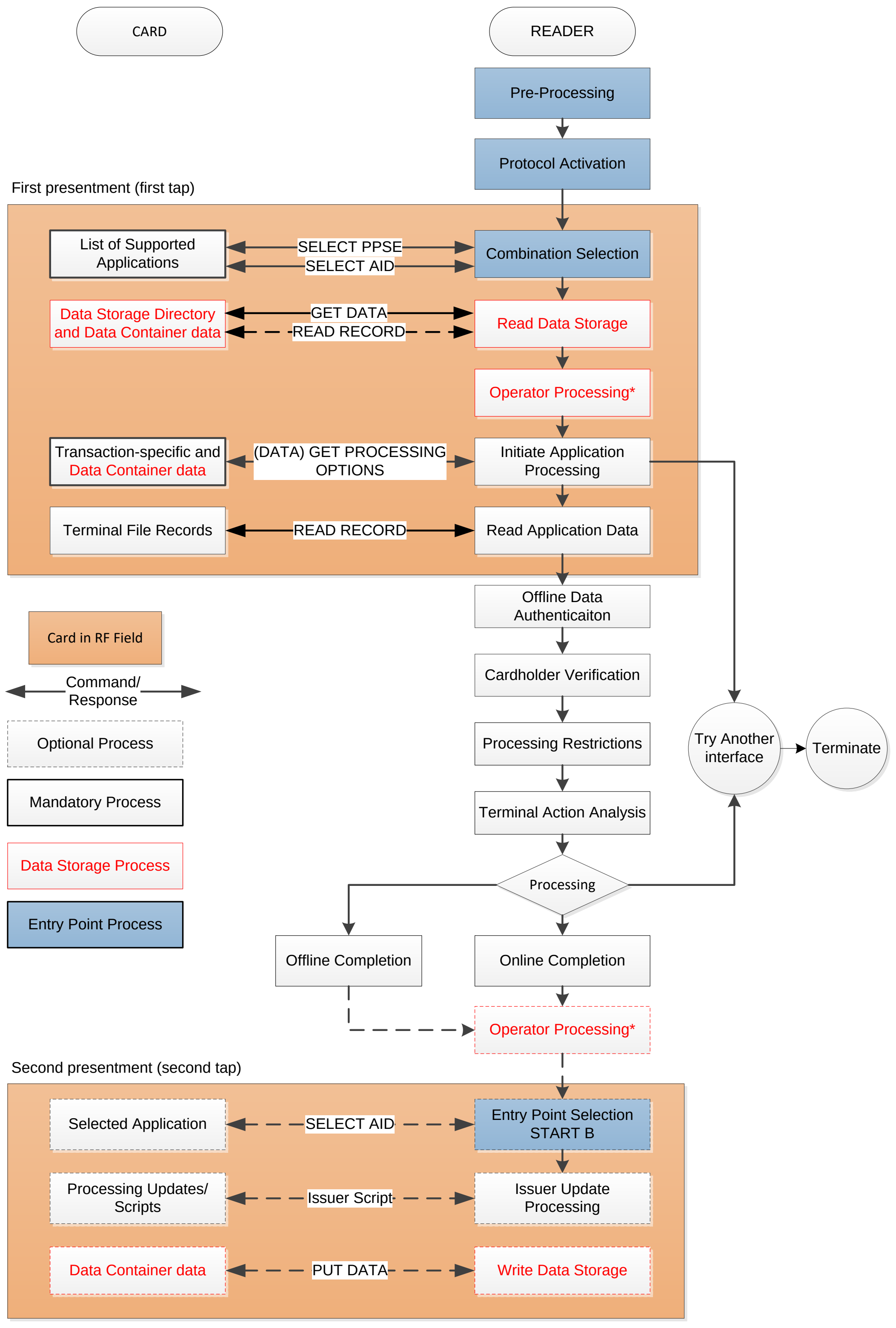
Issuers may allow a Contactless Card to be presented a second time (via a second presentment) for updating application data or Data Storage. The need for a second presentment is determined by the contactless reader by checking for the presence of Issuer scripts included in the authorization request response and / or Data Storage data prepared by the Terminal. The reader shall store the fields that are required for a second presentment in transient memory as specified in [EMV Book B]. Entry Point uses this transient memory to select the Combination used in the previous Final Combination Selection to deliver the Issuer Update Processing commands and / or Data Storage updates.
The EMV configuration uses the Outcome parameters defined in Annex B and provides data for online messages and clearing records.
The EMV transaction flow is illustrated in Figure 2-1. The EMV Mode transaction flow with optional Data Storage is illustrated in Figure 2-2, and with optional Tearing Recovery in Figure 2-3.

Figure 2-1: Kernel 6 Contactless EMV Transaction Flow



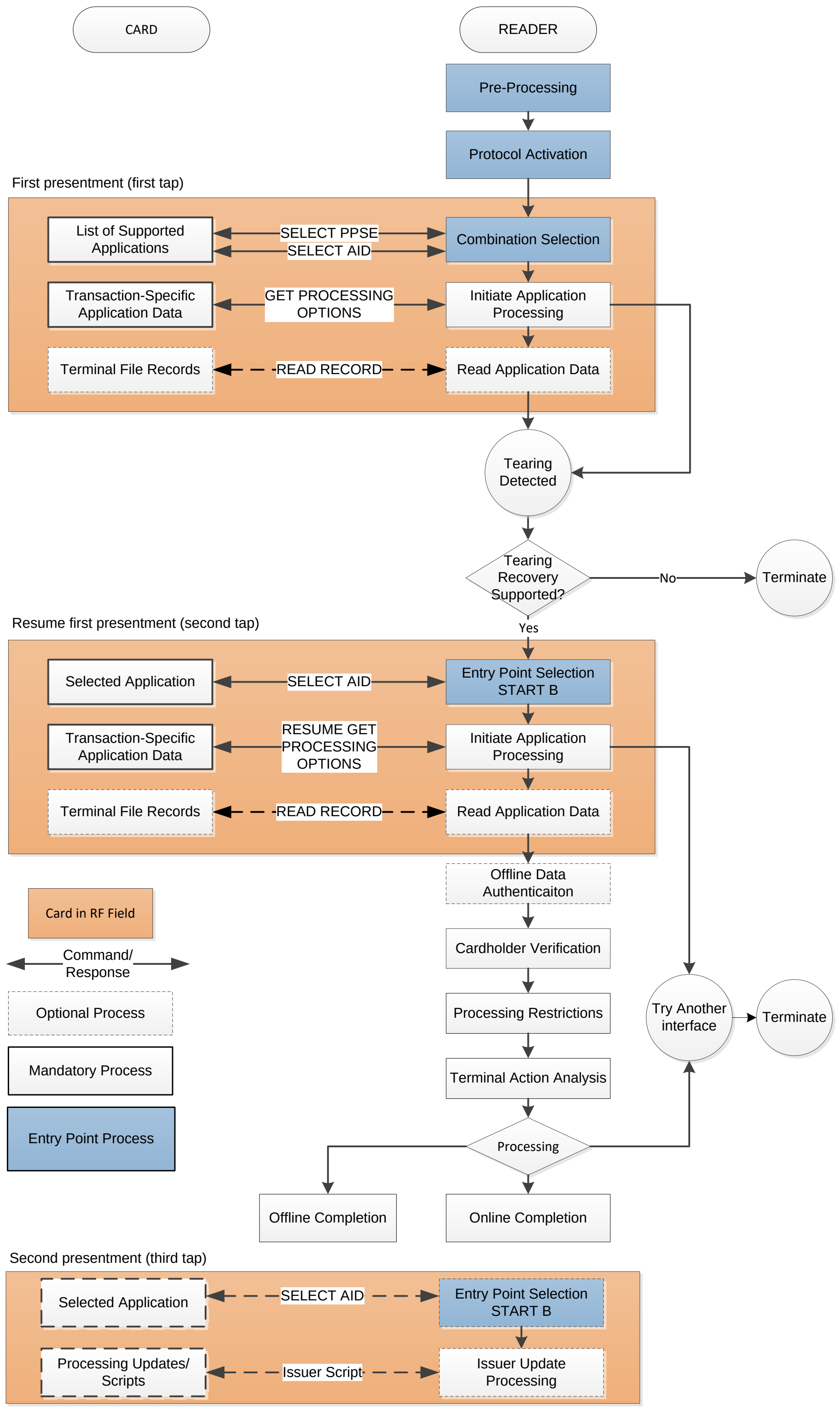
*Transaction-Specific Application Data are provided in Table 4-4.

Figure 2-2: Kernel 6 Contactless EMV Transaction Flow with Data Storage



*Operator-specific business logic (terminal functionality).

Figure 2-3: Kernel 6 Contactless EMV Transaction Flow with Tearing Recovery



Note: Tearing Recovery is also supported for EMV transactions using Data Storage.

2.3 Configuration, Functionality, and APDU Command Support

Section Summary:

2.3.1 POS System Configuration Options

The POS System Configuration options are used by the Kernel and must be set as described in [EMV Book A] and [EMV Book B]. In addition, the following configuration options are specific to Kernel 6.

1. **Acceptance Environment:** Reader will be enabled to support the EMV transactions
2. **Kernel AID:** Discover AIDs will be configured to be used with Kernel 6.

2.3.2 Functionality

Specific Contactless functionalities are included in Table 2-1 and classified based on the following definitions:

- **Mandatory:** This functionality must be present and must conform to the behavior described in this Specification
- **Conditional:** The presence of the functionality is dependent on another function
- **Optional:** This functionality must be present in the Terminal application but the Acquirer may choose not to use the function.

Table 2-1: Contactless Functionality

Function	**Kernel 6 Terminal Support**	**Notes**
Entry Point Processing based on [EMV Book B]	Mandatory	The Terminal shall support Entry Point processing as defined in [EMV Book B]
Combination Selection Post Processing	Mandatory	See Section 3.1: Combination Selection Post Processing
Tearing Recovery	Mandatory	
Read Data Storage	Mandatory	
Operator Processing	Mandatory	Invocation of proprietary business logic for Data Storage or Extended Logging.
Initiate Application Processing	Mandatory	If a data requested by the PDOL is not present in the Terminal, then the Terminal shall set the value to zeroes.
Cardholder Verification Method (CVM) • Online PIN • Signature • Consumer Device CVM • No CVM	Conditional	Mandatory for EMV transactions except as determined by the payment system.
Read Application Data • EMV Data in Terminal File records, identified by Short File Identifier (SFI) and record number	Conditional	Mandatory for Offline capable EMV readers
Offline Data Authentication (ODA) • CDA	Conditional	Mandatory for Offline capable readers and Data Storage (must be performed before Termination Action Analysis)
Processing Restrictions • Application Expiration Date • Application Effective Date • Application Version Number • Application Usage Control* • Exception List	Mandatory	*Conditional – Mandatory for offline transactions
Terminal Action Analysis	Mandatory	

Function	Kernel 6 Terminal Support	Notes
Online Processing	Conditional	Mandatory for Online capable EMV readers
Completion • Offline • Online	Mandatory	
Issuer Update Processing • Application Block • Application Unblock • Put Data • Update Record	Conditional	Mandatory for Online capable EMV readers
Write Data Storage	Mandatory	

2.3.3 APDU Command Support

The APDU commands included in Table 2-2 are designated as “Mandatory”, “Conditional”, or “Not supported” by the Contactless Kernel 6 Terminal based on the following definitions:

- **Mandatory**: The command must be supported.
- **Conditional**: The command shall be supported if the corresponding functionality is enabled
- **Not supported**: The command is not supported.

Table 2-2: APDU Command Support

Command	Kernel 6 Support
APPLICATION BLOCK	Mandatory[1]
APPLICATION UNBLOCK	Mandatory[1]
CARD BLOCK	Not supported
DATA GET PROCESSING OPTIONS	Conditional[2]
EXTERNAL AUTHENTICATE	Not Supported
GENERATE APPLICATION CRYPTOGRAM (AC)	Not Supported
GET CHALLENGE	Not Supported
GET DATA	Mandatory
GET PROCESSING OPTIONS	Mandatory
INTERNAL AUTHENTICATE	Not Supported
PIN CHANGE / UNBLOCK	Not Supported
PUT DATA	Mandatory[1]
READ RECORD	Mandatory
RESUME GET PROCESSING OPTIONS	Conditional[3]
SELECT	Mandatory
UPDATE RECORD	Mandatory[1]

Notes:

1. Terminal to only pass supported Issuer Script commands to card.
2. Mandatory if Data Storage is supported.
3. Mandatory if Tearing Recovery is supported.

2.4 Deferred Authorization

Deferred Authorization enables POS Systems to be used in environments where real time online authorization is not possible.
With Deferred Authorization, when an EMV transaction requires online authorization (and after successful Offline Data Authentication) the Kernel will provide an Approved Outcome. The POS System or back-office may then request authorization at a later time.
Notes:

- Even though real time online authorization is not used, Terminals shall request online cryptograms from cards (by setting TTQ B2b8 = ‘1’). Consequently, Terminals must be configured as online capable, with TTQ B1b4 = ‘0’ (Online capable contactless reader) and a Terminal Type (tag ‘9F35’) of “Online only” or “Offline with online capabilities”
- Online PIN cardholder verification is not supported. Readers must be configured with TTQ B1b3 = ‘0’ (Online PIN not supported)
- The reader must be configured to support Offline Data Authentication, with TTQ B1b1 = ‘1’ (ODA for online authorization supported), and
- Issuer Update Processing is not supported, the reader must be configured with TTQ B3b8 = ‘0’ (Issuer update processing not supported).

2.5 Data Storage

Data Storage provides transit operators, merchants, and other business entities (referred to generically as *operators* herein) with the ability to read and write custom merchant data on cards during EMV contactless transactions.
Data Storage is an optional feature for card and terminal applications, and may be used as part of an EMV transaction when supported by both.

2.5.1 General Features

Data Storage has the following features:

- Use of on-card Data Containers, in which operators can store their own data. Three types of Data Container are supported:
    - Transient Containers – freely readable and writable by any operator
    - Operator Containers – freely readable and with write access for the allocated operator only, and
    - Issuer Containers – freely readable with issuer-only write access.
- Use of Container IDs to uniquely identify and address individual Data Containers
- Use of an on-card Data Storage Directory that records the allocation of Data Containers on the card. The Data Storage Directory is a dynamic data object (comprising a sequence of Directory Entries, one for each allocated Data Container) that can be read by operators using the GET DATA command
- Mapping of Data Containers to card file records, allowing the retrieval of their contents by operators using the READ RECORD command
- Use of Data Exchange by Kernel to invoke operator-specific* processing in the Terminal prior to Initiate Application Processing
- Use of the DATA GET PROCESSING OPTIONS command (which extends the GET PROCESSING OPTIONS command) to update Data Container content during Initiate Application Processing
- Optional use of the PUT DATA command by operators and issuers to update Data Container content during card second presentment
- The capability for issuers to allocate Operator Containers to operators, either during card personalization or via issuer scripting
- Use of Offline Data Authentication, container Write Counters and Integrity Codes for operators to authenticate cards and to guard against skimming or replay of container content.

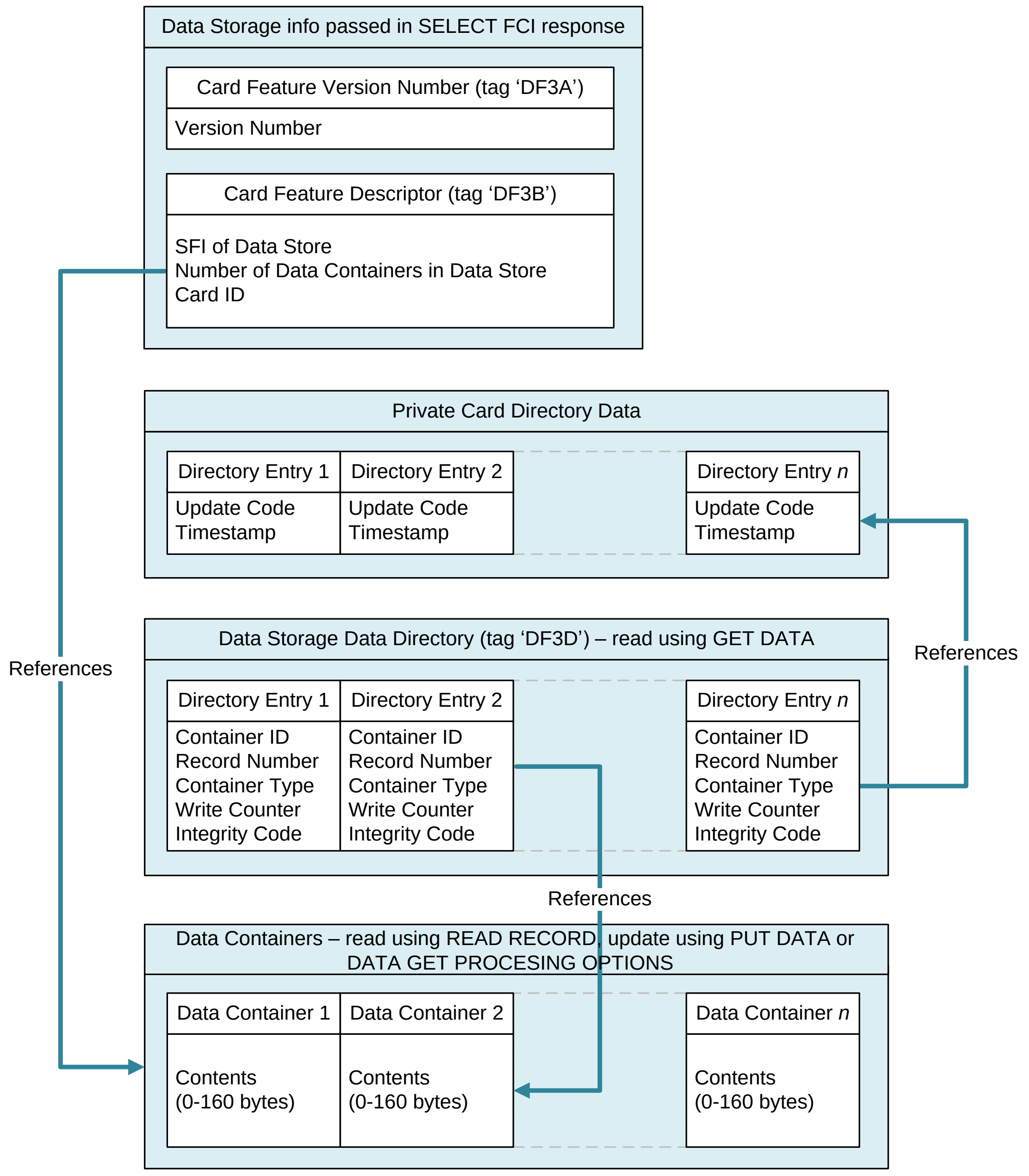
* The use of operator-specific logic is proprietary and is outside the scope of this specification.

2.5.2 Data Structure Overview

Figure 2-4 shows the logical data structure used by the card for data storage.
Summary data storage information is passed from card application to terminal as part of the application selection response. This information includes the version number, the number and location of records in the data store, and the card application's unique identity.
The Kernel can retrieve public Data Storage Directory information from the card using the GET DATA command. This comprises a sequence of Directory Entries, one for each allocated Data

Container. A container's Directory Entry identifies its Container ID, the record holding its content and other attributes.
The Kernel can read a Data Container's content using the READ RECORD command.

Figure 2-4: Logical Data Storage Data Structure



2.5.3 Data Containers

A Data Container is a logical unit of data storage, addressable by its Container ID. A Data Container must be allocated on a card before an operator can use it to store data. During allocation a mapping is created on the card between the Container ID and the file record that stores the container's content.

There are three types of Data Container, all are freely readable but each have differing write-access and allocation approaches, as shown in Table 2-3.

Table 2-3: Data Container Properties

Property	Data Container Type		
	Transient Container	**Operator Container**	**Issuer Container**
Allocation	By card application	By issuer	By issuer
Read-access	Any operator	Any operator	Any operator
Write-access	Any operator and the issuer	Allocated operator (using Update Codes) and the issuer only	Issuer only

Transient Containers are allocated as needed by the card application during a transaction at an operator's terminal. The card will persist the contents of Transient Containers between transactions and until overwritten, either by the same operator, a different operator, or the issuer. When allocating Transient Containers, the card application will select either an unallocated Transient Container (if one is free), or the oldest of its allocated Transient Containers (overwriting its current Container ID and content).
Operator Containers are allocated by the issuer, either during card personalization or via issuer scripting. An Operator Container may only be updated by the operator to which it is allocated or the issuer. Operator write-access is enforced using Update Codes; the card will update container content only when the correct Update Code is used by the operator (see below).
Issuer Containers are allocated by the issuer, either during card personalization or via issuer scripting. Issuer Containers may only be updated by the issuer (via secure messaging).
Card applications implementing data storage must be personalized with a minimum of three Transient Containers.

2.5.4 Operator Container Update Codes

The card holds a hidden Update Code for each Operator Container, and will only allow an update to an Operator Container if the terminal can prove that it knows the same Update Code. This works as follows:

- To update an Operator Container, the operator computes a hash over the data to be updated, data provided by the card application, and its copy of the Update Code. This hash value is called the Update Code Hash. The computed Update Code Hash is sent to the card application as part of the update request
- The card application computes a hash over the same data items, but uses its copy of the Update Code
- The card application then compares its computed Update Code Hash with the given Update Code Hash. If they do not match, then the update request will be rejected, otherwise the card will:
    - Write the new content into the container's file record
    - Update the container's Write Counter, Integrity Code and Timestamp.

Operators may also replace the stored Update Code with a new value as part of the update, if required. The operator is free to use any suitable strategy for managing its Update Codes (see Annex E for guidance). Initial Update Code values must be agreed between operators and issuers.

2.5.5 Container Identifiers

Each allocated Data Container is uniquely identified on a card by its Container ID.
Only one Data Container may be allocated per Container ID per card.
An operator may be assigned one or more Container IDs by the Data Storage Registration Authority.
Any operator requiring simultaneous use of multiple Data Containers will require multiple Container IDs, one for each Data Container needed.
The Container ID of zero ('00000000') is reserved and is used by card and terminal to indicate an unallocated Data Container.
The Data Storage Registration Authority can assign shared Container IDs between operators. This allows groups of two or more operators to share the same Data Container. For example, a group of co-operating transit operators may each be assigned a unique Container ID for their own individual use, plus a shared Container ID for use as a group.
The card and terminal make no explicit distinction between unique and shared Container IDs.

2.6 Extended Logging

Extended Logging provides merchants with a simple mechanism for recording custom data in a card's transaction log during EMV contactless transactions.
If Extended Logging is enabled, the Kernel will use Data Exchange to invoke operator-specific* processing in the Terminal prior to performing Initiate Application Processing. The Terminal response may provide an Extended Logging Data (tag 'DF3C') data element. This data element can contain up to 32 bytes of custom data and, if provided, is appended to the GET PROCESSING OPTIONS command data. The card will include the data in its own transaction log.
The card's cyclic transaction log can be read later at a special terminal using the READ RECORD command.
Support for Extended Logging is optional for card and Kernel, and shall be used only when supported by both.
* The use of operator-specific logic is proprietary and is outside the scope of this specification.

2.7 Tearing Recovery

For contactless transactions, interaction between card and reader can be interrupted if the card is removed prematurely from the RF field or when a protocol error occurs. This can result in "transaction tearing", where the card and terminal have differing transaction outcomes.
With Tearing Recovery, if an EMV transaction tears, the Kernel can invite the cardholder to present the card again. The Kernel may then request the card resume the previous transaction, rather than starting a new transaction.
Support for Tearing Recovery is optional for card and Kernel, and shall be used only when supported by both.

2.7.1 Process Overview

At the start of transaction processing, the Kernel determines if it needs to recover from a previous torn transaction or begin a new transaction.
This begins with the Kernel examining the settings in the card's optional Card Feature Descriptor to determine its support for Tearing Recovery and to extract its unique Card ID.

To support Tearing Recovery, the Kernel keeps a record of transaction details in a Tearing Log; populating the log at the start of Initiate Application Processing, then emptying it at the end of successful Read Application Data processing.

If the card supports Tearing Recovery and its Card ID matches that recorded in the Tearing Log, then:

- The Kernel uses the transaction details recorded in the Tearing Log to send the RESUME GET PROCESSING OPTIONS (RESUME GPO) command to the card
- The card compares the PDOL data with that of the previous GET PROCESSING OPTIONS (GPO) or DATA GET PROCESSING OPTIONS (DATA GPO) command (recorded in its own internal log), and:
    - If there is a match, the card returns the same response message as the previous GPO or DATA GPO command (without updating its CRM counters and accumulators, or incrementing ATC), or
    - If there is no match, the card processes the command as a new GPO or DATA GPO command (incrementing its ATC and updating CRM counters and accumulators).

Else, the Kernel starts a new transaction, and:

- Records the new transaction details in the Tearing Log (if Tearing Recovery enabled)
- Sends the GPO or DATA GPO command to the card.

If there is a timeout or transmission error during the Initiate Application Processing or Read Application Data processing steps, and the Tearing Log is not empty, then the Kernel can invite the cardholder to present the card again. Otherwise, the transaction can proceed and the Kernel can clear the Tearing Log.

[This page is intentionally left blank.]

3 Terminal Transaction Processing Steps

This section addresses the Kernel 6 transaction processing requirements, including process flows, in the following sections:

- 3.1: Combination Selection Post Processing
- 3.2: Read Data Storage
- 3.3: Initiate Application Processing
- 3.4: Read Application Data
- 3.5: Tearing Analysis
- 3.6: Offline Data Authentication (ODA) for Offline Transactions
- 3.7: Cardholder Verification
- 3.8: Processing Restrictions
- 3.9: Terminal Action Analysis
- 3.10: Online Processing
- 3.11: Completion
- 3.12: Issuer Update Processing
- 3.13: Write Data Storage

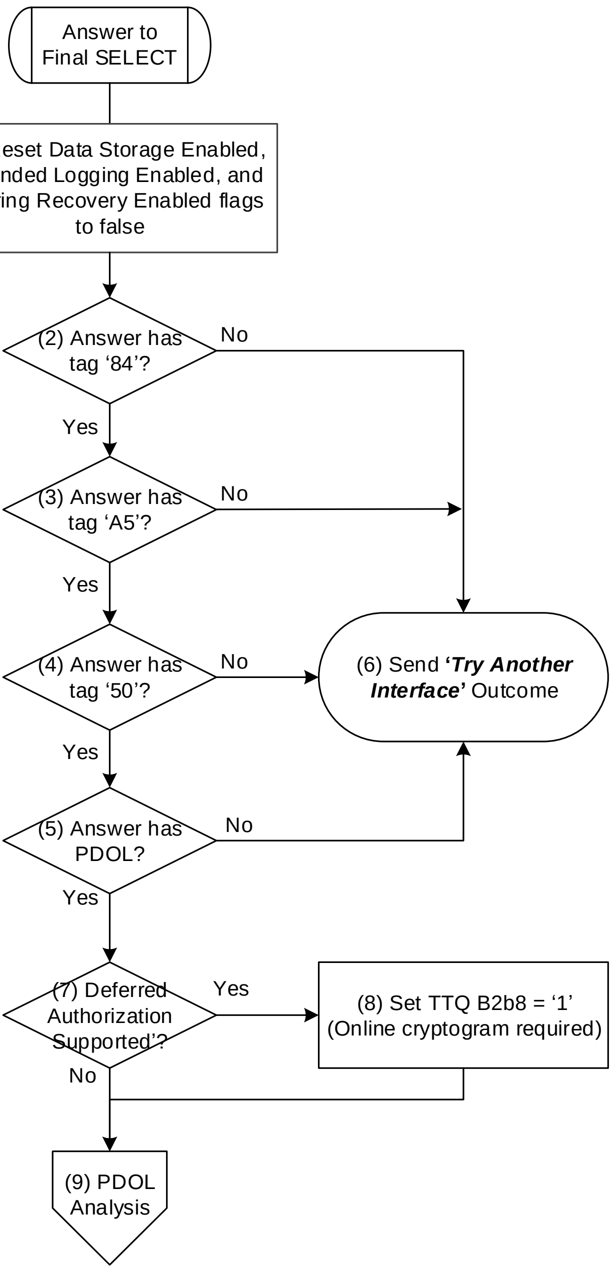
Details on each APDU command referenced in this section are provided in Section 4, and additional information regarding Outcomes and Outcome Parameters are included in Annex B.

3.1 Combination Selection Post Processing

The transaction starts in the Kernel at the handover from Entry Point, which has performed EMV transaction Pre-Processing, Protocol Activation, Combination Selection, and Kernel Activation), or
After Kernel Activation, the selected Kernel must check the response sent by the card to the SELECT command. The operations to be performed to validate the answer sent by the card are described in the following sections:

- Figure 3-1: Processing Flow (Format Analysis)
- Figure 3-2: Processing Flow (PDOL Mandatory Check)
- Figure 3-3: Processing Flow (PDOL Optional Check)
- Figure 3-4: Processing Flow (Optional Checks)
- Figure 3-5: Processing Flow (Optional Feature Checks)
- Figure 3-6: Processing Flow (Tearing Recovery Check)

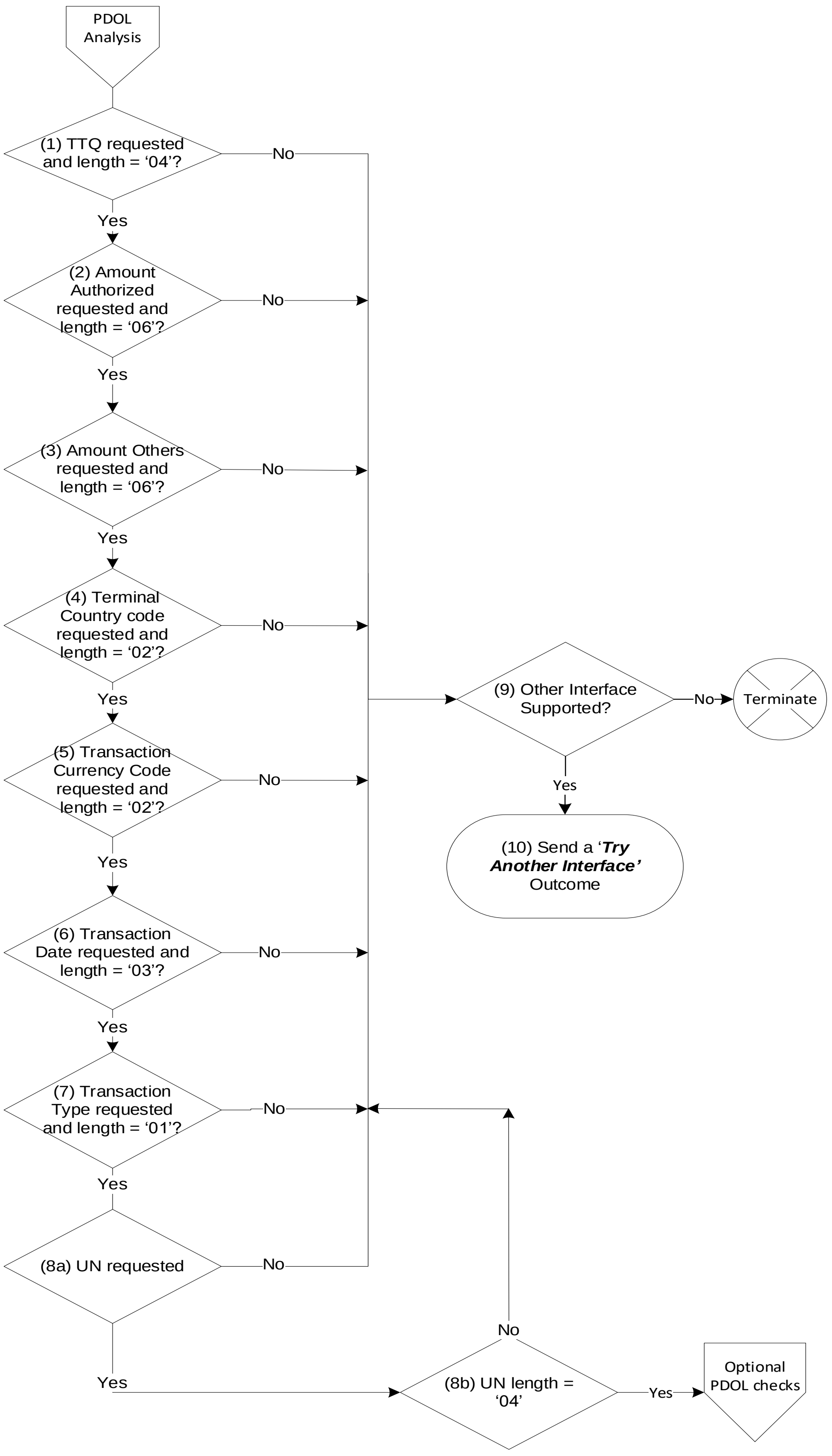
Figure 3-1: EMV Application Selection – Processing Flow (Format Analysis)


Answer to Final SELECT
(1) Reset Data Storage Enabled, Extended Logging Enabled, and Tearing Recovery Enabled flags to false
(2) Answer has tag '84'?
No
Yes
(3) Answer has tag 'A5'?
No
Yes
(4) Answer has tag '50'?
No
(6) Send '***Try Another Interface***' Outcome
Yes
(5) Answer has PDOL?
No
Yes
(7) Deferred Authorization Supported'?
Yes
(8) Set TTQ B2b8 = '1' (Online cryptogram required)
No
(9) PDOL Analysis

#	Description
1	Kernel shall reset its transient Data Storage Enabled, Extended Logging Enabled, and Tearing Recovery Enabled flags to false.
2	The Kernel shall begin the checks on the response received for the final SELECT command. Kernel shall check if "DF name" (tag '84') of the application is provided by the card. If the tag is not present, then go to Step 6. Else continue.

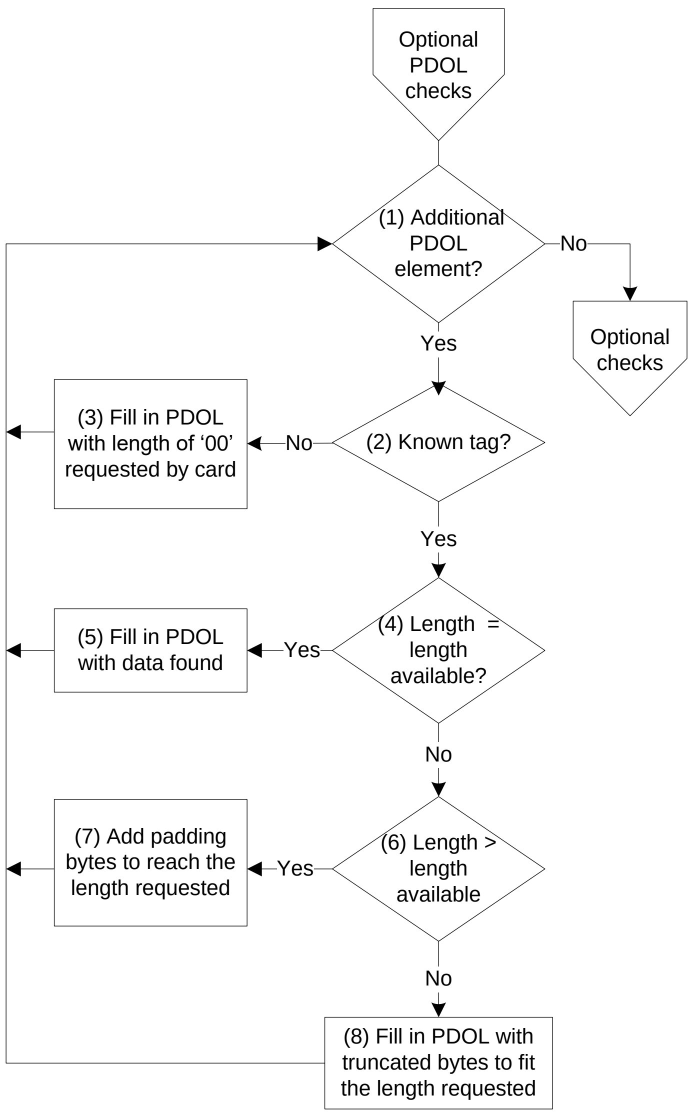
#	**Description**
3	Kernel shall check if FCI Proprietary Template (tag 'A5') is present. If the tag is not present, then go to Step 6. Else continue.
4	Kernel shall check if Application Label (tag '50') is present in the answer provided by the card and is under tag 'A5'. If the tag is not present as specified, then go to Step 6. Else continue.
5	Kernel shall check if Processing Options Data Object List (PDOL) (tag '9F38') is present in the answer provided by the card and is under tag 'A5'. If the tag is not present as specified, then go to Step 6. Else go to Step 7.
6	Kernel shall send the Outcome as '***Try Another Interface***' to request the use of another interface (if one is supported) to the Entry Point.
7	Kernel shall check if it is configured for Deferred Authorizations (true if 'Deferred Authorization Supported' flag is present and set to '1'). If yes, go to Step 8. Else go to Step 9.
8	Terminal shall indicate in the TTQ that it wishes to process the transaction online (by setting the TTQ B2b8 = '1')
9	Kernel shall go to Figure 3-2 to perform PDOL Analysis.

Figure 3-2: EMV Application Selection – Processing Flow (PDOL Mandatory Check)


PDOL Analysis
(1) TTQ requested and length = '04'?
No
Yes
(2) Amount Authorized requested and length = '06'?
No
Yes
(3) Amount Others requested and length = '06'?
No
Yes
(4) Terminal Country code requested and length = '02'?
No
Yes
(5) Transaction Currency Code requested and length = '02'?
No
Yes
(6) Transaction Date requested and length = '03'?
No
Yes
(7) Transaction Type requested and length = '01'?
No
Yes
(8a) UN requested
No
Yes
(9) Other Interface Supported?
No
Terminate
Yes
(10) Send a '***Try Another Interface'*** Outcome
(8b) UN length = '04'
No
Yes
Optional PDOL checks

#	**Description**
1	Kernel shall check if the card requests a TTQ (tag ‘9F66’) with a length of 4 bytes. If the TTQ is not requested or the TTQ length is different from 4, then the Kernel shall go to Step 9.
2	Kernel shall check if the card requests Amount Authorized (tag ‘9F02’) with a length of 6 bytes. If Amount Authorized is not requested or the Amount Authorized length is not equal to 6, then the Kernel shall go to Step 9.
3	Kernel shall check if the card requests Amount Other (tag ‘9F03’) with a length of 6 bytes. If Amount Others is not requested or the Amount Others length is not equal to 6, then the Kernel shall go to Step 9.
4	Kernel shall check if the card requests Terminal Country Code (tag ‘9F1A’) with a length of 2 bytes. If Terminal Country Code is not requested or the Terminal Country Code length is not equal to 2, then the Kernel shall go to Step 9.
5	Kernel shall check if the card requests Transaction Currency Code (tag ‘5F2A’) with a length of 2 bytes. If the transaction Currency Code is not requested or the Transaction Currency Code length is not equal to 2, then the Kernel shall go to Step 9.
6	Kernel shall check if the card requests Transaction Date (tag ‘9A’) with a length of 3 bytes. If the Transaction Date is not requested or the Transaction Date length is not equal to 3, then the Kernel shall go to Step 9.
7	Kernel shall check if the card requests Transaction Type (tag ‘9C’) with a length of 1 byte. If the Transaction Type is not requested or the Transaction Type length is not equal to 1, then the Kernel shall go to Step 9.
8	a. Kernel shall check if the card requests Unpredictable Number (UN) (tag ‘9F37’). If not present, the Kernel shall go to Step 9. b. Kernel will check for UN length of ‘04’ bytes. If yes, the Transaction will go to Figure 3-3 to perform optional PDOL checks. If no, go to Step 9. **Note**: If the UN is not requested or the UN length is not correct, then the Kernel shall go to Step 9.
9	Kernel shall check if it supports another interface. • If Terminal does not support another interface, the Terminal shall terminate the transaction with ‘End Application’ (for Processing Error) Outcome. ELSE go to Step 10.
10	If another interface is supported by the Terminal, the Kernel shall provide the Outcome ‘***Try Another Interface***’ to Entry Point.

Figure 3-3: EMV Application Selection – Processing Flow (PDOL Optional Check)

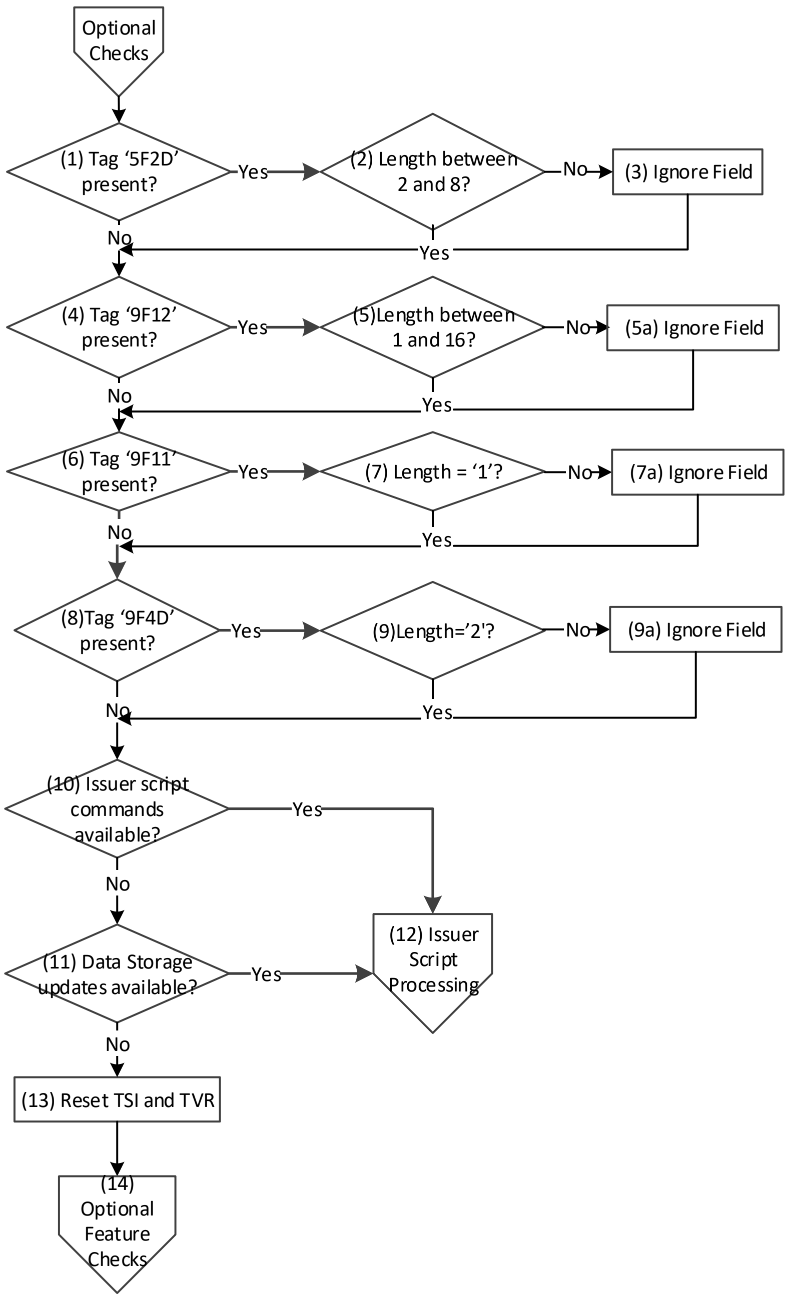

Optional PDOL checks
(1) Additional PDOL element?
No
Optional checks
Yes
(2) Known tag?
No
(3) Fill in PDOL with length of '00' requested by card
Yes
(4) Length = length available?
Yes
(5) Fill in PDOL with data found
No
(6) Length > length available
Yes
(7) Add padding bytes to reach the length requested
No
(8) Fill in PDOL with truncated bytes to fit the length requested

#	**Description**
1	Kernel shall check if the PDOL requested by the card contains additional data to be sent with the GET PROCESSING OPTION. If… / Then… Yes / Go to Step 2. No / Go to Figure 3-4 to perform optional checks.

If…	Then…
Yes	Go to Step 2.
No	Go to Figure 3-4 to perform optional checks.

#	Description		
2	Kernel verifies if the data element requested by the card is a Discover D-PAS tag or EMV tag.		
		If…	Then…
		Yes	Go to Step 4.
		No	Go to Step 3.
3	If the requested data is unknown by the Kernel, the Kernel shall add the number of ‘00’ bytes requested by the card to data that will be sent to the card. Then, go to Step 1.		
4	Kernel shall check If the length requested by the card concerning the known element is equal to the length available in the Kernel.		
		If…	Then…
		Yes	Go to Step 5.
		No	Go to Step 6.
5	Kernel inserts requested data into the PDOL that will be sent to the card. Then, go to Step 1.		
6	Kernel checks if the length requested by the PDOL entry is greater than the length of available data inside the Kernel.		
		If…	Then…
		Yes	Go to Step 7.
		No	Go to Step 8.
7	Kernel shall add padding bytes (as per [EMV Book 3] Section 5.4 padding rules) to reach the size requested by the card and add the result to the PDOL that will be sent to the card. Then, go to Step 1.		
8	If the length of data requested by the card in the PDOL entry is lower than the length of available data inside the Kernel, the Kernel shall include truncated bytes to match the requested length (as per [EMV Book 3] Section 5.4). Then, go to Step 1.		

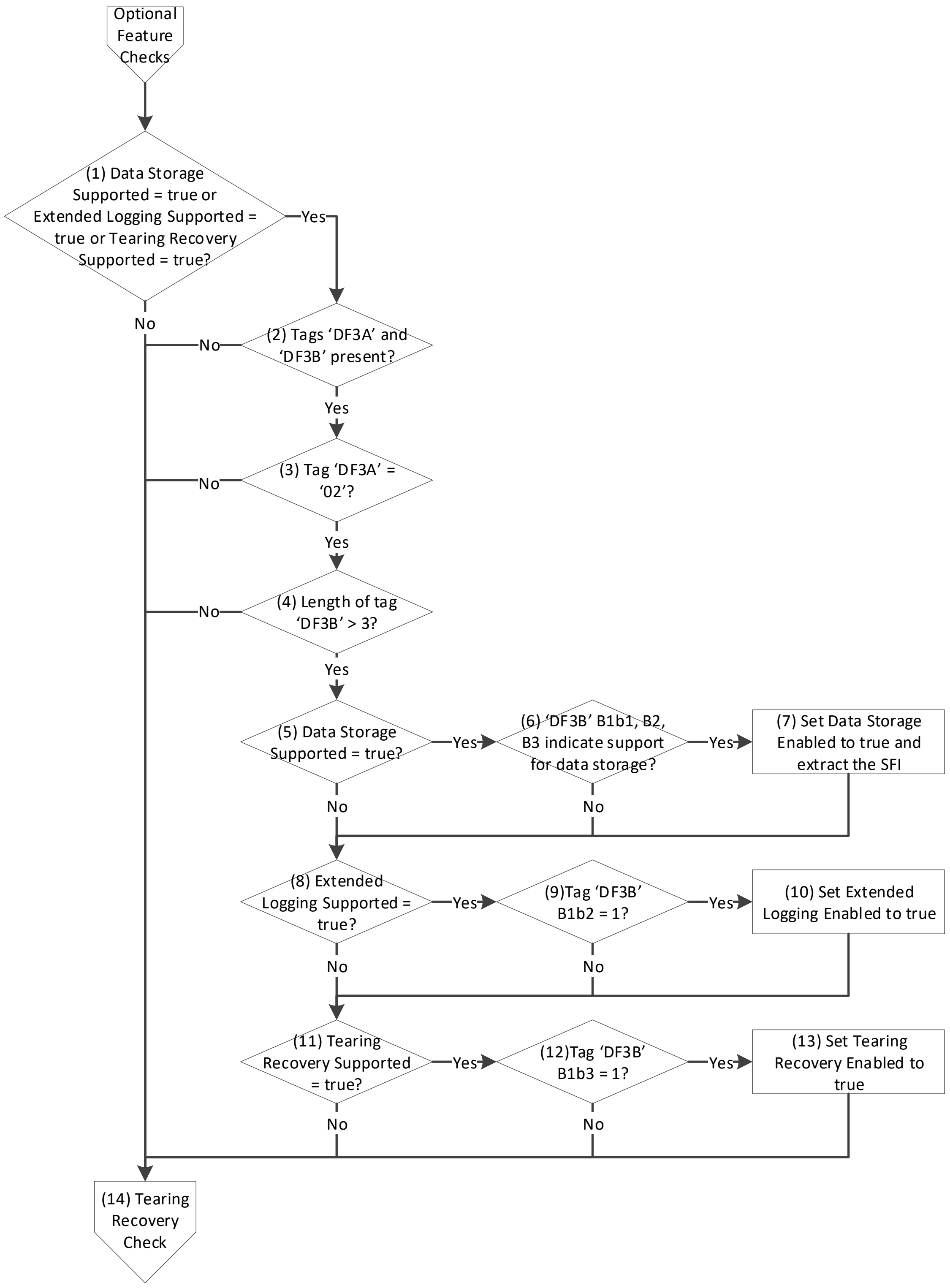
Figure 3-4: EMV Application Selection – Processing Flow (Optional Checks)



#	**Description**
1	Kernel shall check if the card has returned Language Preference (tag ‘5F2D’). If data is not present, the Kernel shall continue the optional checks. If… | Then… Yes | Go to Step 2. No | Go to Step 4.
2	Kernel verifies if Language Preference (tag ‘5F2D’) length is coded between 2 and 8 bytes and is a multiple of 2. If… | Then… Yes | Go to Step 4. No | Go to Step 3.
3	If the length returned by the card is not the expected one, the Kernel shall ignore the field and continue the optional checks.
4	Kernel shall check if Application Preferred Name (tag ‘9F12’) is sent by the card. If data is not present, the Kernel shall continue the optional checks (Step 6).
5	Kernel verifies if the Application Preferred Name (tag ‘9F12’) length is between 1 and 16 bytes. If the length returned by the card is not the expected one, the Kernel shall ignore the field (Step 5a) and continue the optional checks (Step 6).
6	Kernel shall check if Issuer Code Table Index (tag ‘9F11’) is sent by the card. If data is not present, the Kernel shall continue the optional checks (Step 8).
7	Kernel verifies if Issuer Code Table Index (tag ‘9F11’) length is 1 byte. If the length returned by the card is not the expected one, the Kernel shall ignore the field (Step 7a) and continue the optional checks (Step 8).
8	Kernel shall check if Log Entry (tag ‘9F4D’) is sent by the card. If data is not present, the Kernel shall continue the optional checks (Step 10).
9	Kernel verifies if Log Entry (tag ‘9F4D’) length is 2 bytes. If the length returned by the card is not the expected one, the Kernel shall ignore the field (Step 9a) and continue the optional checks (Step 10).
10	Kernel shall check if it received Issuer script commands sent by Issuer (indicating card second presentment). If… | Then… Yes | Go to Step 12. No | Go to Step 11.
11	Kernel shall check if it received a Write Data Storage Template (tag ‘BF11’) from the Terminal (indicating card second presentment). If… | Then… Yes | Go to Step 12. No | Go to Step 13.
12	Kernel shall go to Figure 3-20 to process card second presentment.

#	**Description**
13	New transaction is being processed. All information concerning the previous transaction (TVR and TSI) shall be reset.
14	The transaction will be performed using EMV processing. The Kernel shall continue to Figure 3-5 to perform Optional Feature Checks.

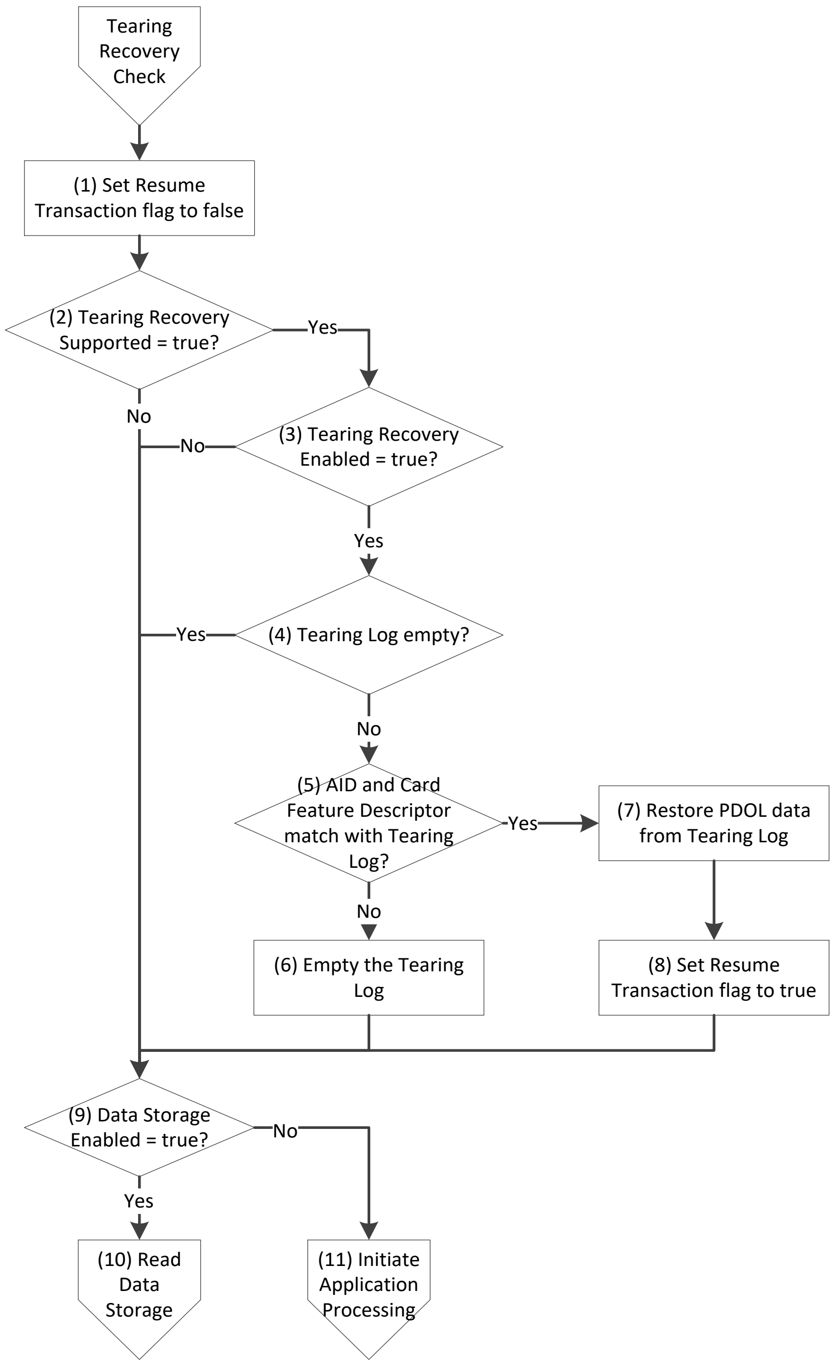
Figure 3-5: EMV Application Selection – Processing Flow (Optional Feature Checks)


Optional Feature Checks
(1) Data Storage Supported = true or Extended Logging Supported = true or Tearing Recovery Supported = true?
Yes
No
(2) Tags 'DF3A' and 'DF3B' present?
No
Yes
(3) Tag 'DF3A' = '02'?
No
Yes
(4) Length of tag 'DF3B' > 3?
No
Yes
(5) Data Storage Supported = true?
Yes
(6) 'DF3B' B1b1, B2, B3 indicate support for data storage?
Yes
(7) Set Data Storage Enabled to true and extract the SFI
No
No
(8) Extended Logging Supported = true?
Yes
(9)Tag 'DF3B' B1b2 = 1?
Yes
(10) Set Extended Logging Enabled to true
No
No
(11) Tearing Recovery Supported = true?
Yes
(12)Tag 'DF3B' B1b3 = 1?
Yes
(13) Set Tearing Recovery Enabled to true
No
No
(14) Tearing Recovery Check

#	Description
1	Kernel verifies if it is configured to support Data Storage, Extended Logging, or Tearing Recovery. If… | Then… Yes | Go to Step 2. No | Go to Step 14.
2	Kernel shall check if Card Feature Version Number (tag 'DF3A') and Card Feature Descriptor (tag 'DF3B') are sent by the card. If… | Then… Yes | Go to Step 3. No | Go to Step 14.
3	Kernel shall check if Card Feature Version Number (tag 'DF3A') equals '02'. If… | Then… Yes | Go to Step 4. No | Go to Step 14.
4	Kernel shall verify that the Card Feature Descriptor (tag 'DF3B') has a length of more than 3 bytes. If… | Then… Yes | Go to Step 5. No | Go to Step 14.
5	Kernel shall check if it is configured to support Data Storage – Data Storage Supported flag is set to true. If… | Then… Yes | Go to Step 6. No | Go to Step 8.
6	Kernel shall check if the card is configured to support Data Storage – Card Feature Descriptor (tag 'DF3B') B1b1 is set to '1' and both Card Feature Descriptor (tag 'DF3B') B2 and B3 are not all zeroes. If… | Then… Yes | Go to Step 7. No | Go to Step 8.
7	Kernel shall set its transient Data Storage Enabled flag to true. The Kernel shall extract the Data Storage SFI from the Card Feature Descriptor (see Table 4-18).
8	Kernel shall verify if it is configured to support Extended Logging – Extended Logging Supported flag is set to true. If… | Then… Yes | Go to Step 9. No | Go to Step 11.

#	Description
9	Kernel shall check if the card is configured to support Extended Logging – Card Feature Descriptor (tag 'DF3B') B1b2 is set to '1'. If… → Then… Yes → Go to Step 10. No → Go to Step 11.
10	Kernel shall set its transient Extended Logging Enabled flag to true.
11	Kernel shall verify if it is configured to support Tearing Recovery – Tearing Recovery Supported flag is set to true. If… → Then… Yes → Go to Step 12. No → Go to Step 14.
12	Kernel shall check if the card is configured to support Tearing Recovery – Card Feature Descriptor (tag 'DF3B') B1b3 is set to '1'. If… → Then… Yes → Go to Step 13. No → Go to Step 14.
13	Kernel shall set its transient Tearing Recovery Enabled flag to true.
14	Kernel shall continue to Figure 3-6 to perform Tearing Recovery Check.

Figure 3-6: EMV Application Selection – Processing Flow (Tearing Recovery Check)

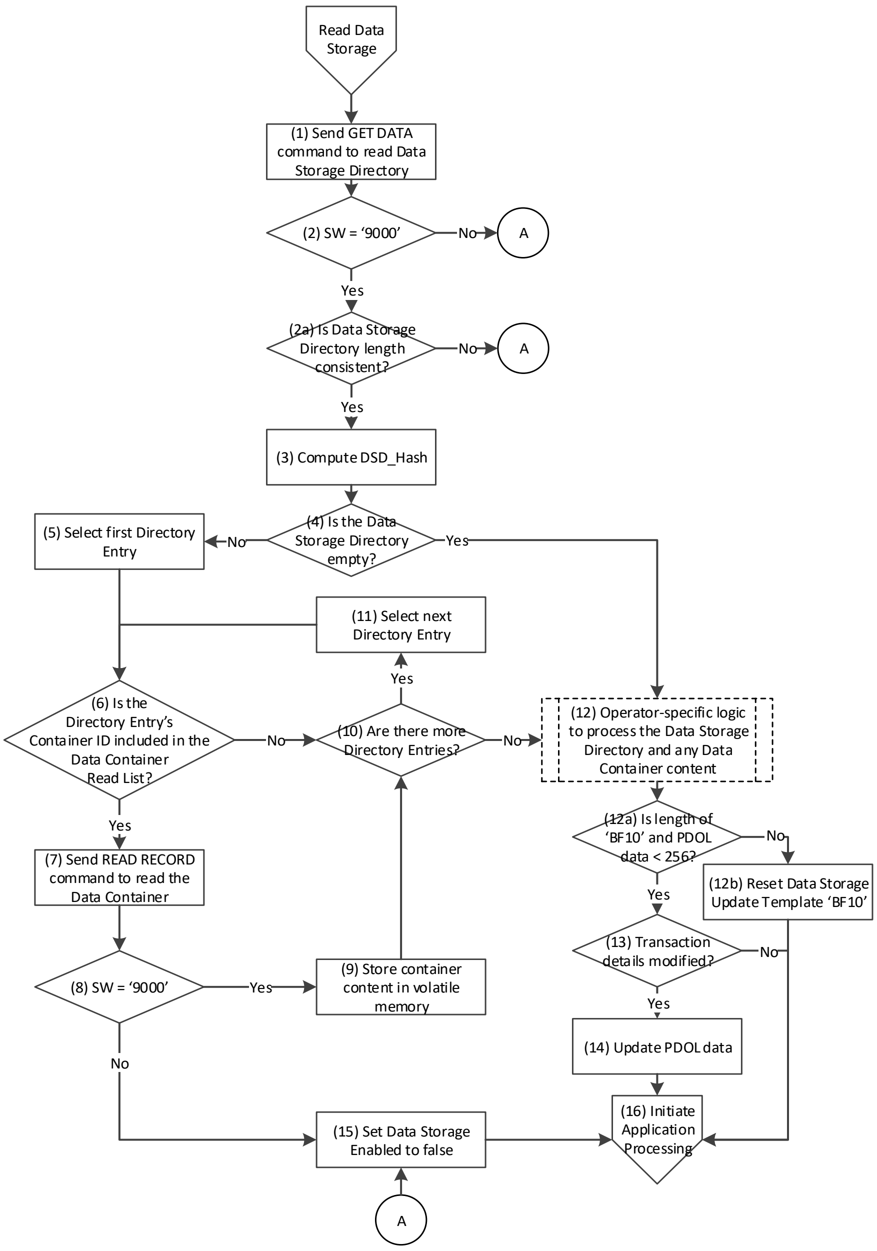

Tearing Recovery Check
(1) Set Resume Transaction flag to false
(2) Tearing Recovery Supported = true?
Yes
No
(3) Tearing Recovery Enabled = true?
No
Yes
(4) Tearing Log empty?
Yes
No
(5) AID and Card Feature Descriptor match with Tearing Log?
Yes
(7) Restore PDOL data from Tearing Log
No
(6) Empty the Tearing Log
(8) Set Resume Transaction flag to true
(9) Data Storage Enabled = true?
No
Yes
(10) Read Data Storage
(11) Initiate Application Processing

#	**Description**		
1	Kernel shall set Resume Transaction flag to false.		
2	Kernel verifies if it is configured to support Tearing Recovery.		
		If…	Then…
		Yes	Go to Step 3.
		No	Go to Step 9.
3	Kernel shall check if Tearing Recovery is enabled for the current transaction (Tearing Recovery Enabled = true).		
		If…	Then…
		Yes	Go to Step 4.
		No	Go to Step 9.
4	Kernel shall check if the Tearing Log is empty		
		If…	Then…
		Yes	Go to Step 5.
		No	Go to Step 9.
5	Kernel shall check if the same card application is being re-presented (true if the selected AID and the card's Card Feature Descriptor and Card Feature Version Number match the values recorded in the Tearing Log).		
		If…	Then…
		Yes	Go to Step 7.
		No	Go to Step 6.
6	Kernel shall clear the Tearing Log (a different card is present). Then go to Step 9.		
7	Kernel shall extract transaction details from the Tearing Log (i.e. use the PDOL data values for each of the data elements listed in the PDOL). **Note**: This shall include the logged value for Unpredictable Number (tag '9F37').		
8	Kernel shall set Resume Transaction flag to true.		
9	Kernel shall check if Data Storage is enabled for the current transaction (Data Storage Enabled = true)		
		If…	Then…
		Yes	Go to Step 10.
		No	Go to Step 11.
10	Kernel shall go to Figure 3-7 to perform Read Data Storage processing.		
11	Kernel shall go to Figure 3-8 to perform Initiate Application Processing.		

3.2 Read Data Storage

Read Data Storage is an optional step that is processed *only if the Data Storage Enabled flag is set*. During this step, the Kernel retrieves the Data Storage Directory from the card (using the GET DATA command), and may read Data Container content (using READ RECORD commands), as described in this section.

Figure 3-7: Read Data Storage – Processing Flow


Read Data Storage
(1) Send GET DATA command to read Data Storage Directory
(2) SW = '9000'
No
A
Yes
(2a) Is Data Storage Directory length consistent?
No
A
Yes
(3) Compute DSD_Hash
(4) Is the Data Storage Directory empty?
No
(5) Select first Directory Entry
Yes
(11) Select next Directory Entry
Yes
(6) Is the Directory Entry's Container ID included in the Data Container Read List?
No
(10) Are there more Directory Entries?
No
(12) Operator-specific logic to process the Data Storage Directory and any Data Container content
Yes
(7) Send READ RECORD command to read the Data Container
(8) SW = '9000'
Yes
(9) Store container content in volatile memory
No
(12a) Is length of 'BF10' and PDOL data < 256?
No
(12b) Reset Data Storage Update Template 'BF10'
Yes
(13) Transaction details modified?
No
Yes
(14) Update PDOL data
(15) Set Data Storage Enabled to false
(16) Initiate Application Processing
A

#	Description		
1	Kernel shall send the GET DATA command to retrieve the Data Storage Directory (tag 'DF3D') from the card.		
2	Kernel shall verify that the command has been successfully processed by the card (by confirming that the Status Word is set to '9000').	If…	Then…
		Yes	Go to Step 3.
		No	Go to Step 15.
2a	Kernel shall verify that the length of the Data Storage Directory (tag 'DF3D') is consistent with having a DSD_UN of 8 bytes and zero or more directory entries of 10 bytes.	If…	Then…
		Yes	Go to Step 3.
		No	Go to Step 15.
3	Kernel shall compute a SHA-1 hash across the contents of the Data Storage Directory object. The computed hash value is the Data Storage Directory Hash (DSD_Hash) and shall be used during Offline Data Authentication.		
4	Kernel shall check if the Data Storage Directory is empty (i.e., contains no Directory Entries, see Figure 4-3).	If…	Then…
		Yes	Go to Step 12.
		No	Go to Step 5.
5	Kernel shall select the first Directory Entry from the Data Storage Directory.		
6	Kernel shall check if the selected Directory Entry's Container ID (see Table 4-22) is included in the Data Container Read List.	If…	Then…
		Yes	Go to Step 6.
		No	Go to Step 9.
7	Kernel shall send the READ RECORD command to retrieve the Data Container from the card, using the record number from the Directory Entry and the SFI extracted from the Card Feature Descriptor.		
8	Kernel shall verify that the command has been successfully processed by the card (by confirming that the Status Word is set to '9000').	If…	Then…
		Yes	Go to Step 9.
		No	Go to Step 15.
9	Kernel shall store the container's content in volatile memory.		

#	Description
10	Kernel shall check if there are more Directory Entries in the Data Storage Directory. If… → Then… Yes → Go to Step 11. No → Go to Step 12.
11	Kernel shall select the next Directory Entry from the Data Storage Directory. Then go to Step 6.
12	Kernel shall clear reset the Data Storage Updates Required flag and use a Data Exchange mechanism to invoke operator-specific processing[1] in the Terminal to analyze the retrieved data and decide: • If Data Container content is authentic[2], • If any Data Container updates are required[3], and/or • Whether to modify any of the transaction details[4]. The request parameters shall include: • PDOL (tag '9F38') and PDOL data elements • Resume Transaction flag • Card Feature Version Number (tag 'DF3A') • Card Feature Descriptor (tag 'DF3B') • Data Storage Directory (tag 'DF3D') • Data Container record(s), if any were read The response parameters may include: • A Data Storage Update Template (tag 'BF10') and commit-control settings (see Table 4-6)[3] • Modified PDOL data elements[4] **Notes**: 1. The use and function of any operator-specific business logic is out of scope of this specification. 2. It is expected that operators will include a MAC within their Data Container content with which to verify its authenticity (see Annex E.2 for guidance). 3. If any Data Container updates are required during Initiate Application Processing. 4. If any changes to transaction details are required (such as changing the Amount, Authorized based on the presence of a loyalty coupon stored in a Data Container, for example), then the corresponding values in the PDOL data shall be modified. The value of Unpredictable Number (UN) must not be changed.

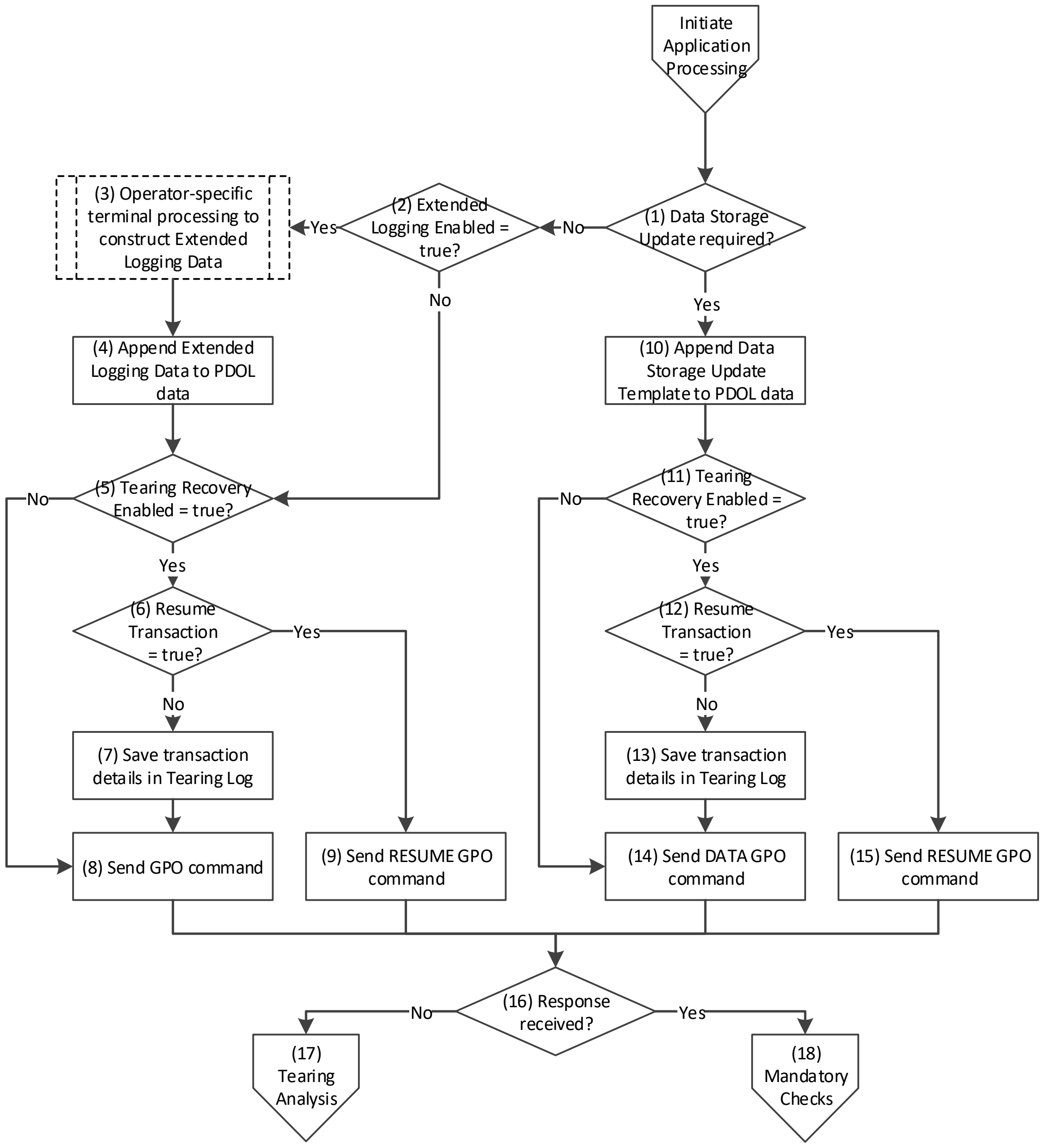
#	**Description**
12a	Verify that the length of the PDOL data when combined with the Data Storage Update Template (tag 'BF10') is less than 256. If… | Then… Yes | Go to Step 13. No | Go to Step 12b.
12b	Reset Data Storage Update Template (tag 'BF10'), removing the data from the template. Kernel may use a Data Exchange mechanism to invoke additional operator-specific processing to inform the operator of the error. This functionality is outside the scope of this specification. Go to Step 16.
13	Kernel shall check if the operator-specific processing modified any of the data elements listed in the PDOL. If… | Then… Yes | Go to Step 14. No | Go to Step 16.
14	Kernel shall update the PDOL data with the modified data element values, and shall go to Step 16. The value of Unpredictable Number (UN) must not be updated.
15	Kernel shall set the Data Storage Enabled flag to false.
16	Kernel shall continue to Figure 3-8 to Initiate Application Processing.

3.3 Initiate Application Processing

At the Initiate Application Processing stage, the Kernel indicates to the card that either a new transaction is beginning or a torn transaction is being resumed. The Kernel does this by sending a GET PROCESSING OPTIONS (GPO), a DATA GET PROCESSING OPTIONS (DATA GPO), or a RESUME GET PROCESSING OPTIONS (RESUME GPO) command as described in this section.

As part of this step, the Kernel will only check if the answer to the command is well formatted and will analyze data later during the Terminal Action Analysis step. Different checks must be performed depending on the type of transaction being processed. The diagrams included in this section show the checks performed by the Kernel when it receives the response to the GPO, DATA GPO or RESUME GPO command.

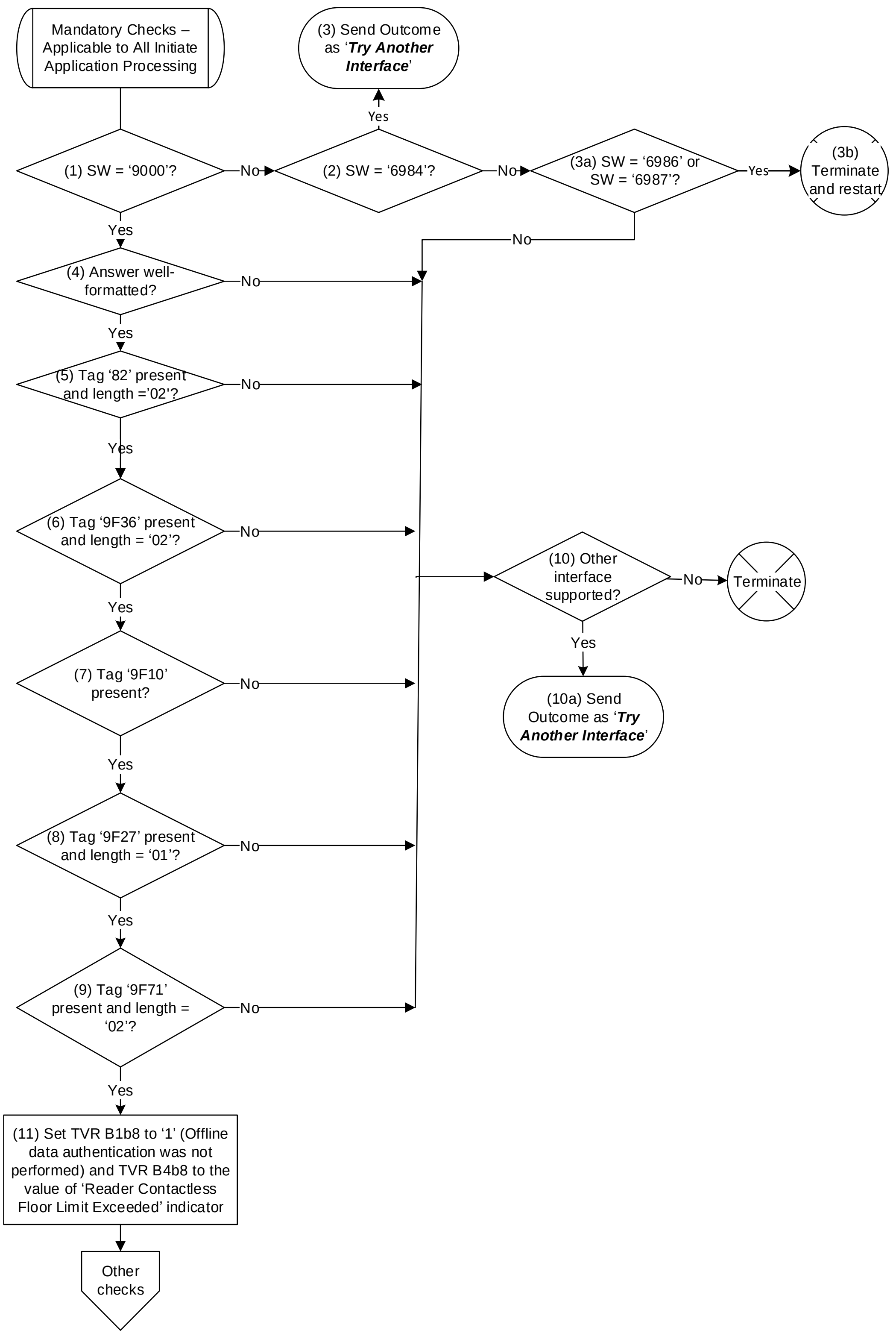
Figure 3-8: Initiate Application Processing – Processing Flow


Initiate Application Processing
(1) Data Storage Update required?
No
(2) Extended Logging Enabled = true?
Yes
(3) Operator-specific terminal processing to construct Extended Logging Data
No
Yes
(4) Append Extended Logging Data to PDOL data
(10) Append Data Storage Update Template to PDOL data
No
(5) Tearing Recovery Enabled = true?
No
(11) Tearing Recovery Enabled = true?
Yes
Yes
(6) Resume Transaction = true?
Yes
(12) Resume Transaction = true?
Yes
No
No
(7) Save transaction details in Tearing Log
(13) Save transaction details in Tearing Log
(8) Send GPO command
(9) Send RESUME GPO command
(14) Send DATA GPO command
(15) Send RESUME GPO command
No
(16) Response received?
Yes
(17) Tearing Analysis
(18) Mandatory Checks

#	Description
1	Kernel shall check if the Data Storage Update Template (and commit-control setting) is available. If… Then… Yes Go to Step 10. No Go to Step 2.
2	Kernel shall check if Extended Logging Enabled is true (indicating that both card and terminal support Extended Logging). If… Then… Yes Go to Step 3. No Go to Step 5.
3	Kernel shall use Data Exchange mechanism to invoke operator-specific processing[1] in the Terminal. The request parameters shall include: • PDOL (tag ‘9F38’) and PDOL data elements • Card Feature Version Number (tag ‘DF3A’) • Card Feature Descriptor (tag ‘DF3B’) The response parameters shall include: • Extended Logging Data (tag ‘DF3C’) **Notes**: The use and function of any operator-specific business logic is out of scope of this specification.
4	Kernel shall append the Extended Logging Data data element to the PDOL data.
5	Kernel shall check if Tearing Recovery Enabled is true (indicating that both card and terminal support Tearing Recovery). If… Then… Yes Go to Step 6. No Go to Step 8.
6	Kernel shall check if Resume Transaction flag is true (indicating that this transaction is recovering from a tear). If… Then… Yes Go to Step 9. No Go to Step 7.
7	Kernel shall record transaction details in the Tearing Log. The details shall include the PDOL data to be used with the GET PROCESSING OPTIONS command. Kernel shall set the value of P1 to ‘00’ in the Tearing Log.
8	Kernel shall send the GET PROCESSING OPTIONS command to the card. Then go to Step 16.

#	Description
9	Kernel shall send the RESUME GET PROCESSING OPTIONS command to the card, with APDU parameter P1 set to '00'. Then go to Step 16.
10	Kernel shall append the Data Storage Update Template to the PDOL data.
11	Kernel shall check if Tearing Recovery Enabled is true (indicating that both card and terminal support Tearing Recovery). If… → Then… Yes → Go to Step 12. No → Go to Step 13.
12	Kernel shall check if Resume Transaction flag is true (indicating that this transaction is recovering from a tear). If… → Then… Yes → Go to Step 15. No → Go to Step 13.
13	Kernel shall record transaction details in the Tearing Log. The details shall include the PDOL data and P1 parameter settings to be used with the DATA GET PROCESSING OPTIONS command. Kernel shall set the value of P1 bit 8 to '1' in the Tearing Log.
14	Kernel shall send the DATA GET PROCESSING OPTIONS command to the card. Then go to Step 16.
15	Kernel shall send the RESUME GET PROCESSING OPTIONS command to the card, with APDU parameter P1 set to the value stored in the Tearing Log. Then go to Step 16.
16	Kernel shall check that a command response is received from the card. If… → Then… Yes → Go to Step 18. No → Go to Step 17.
17	Kernel shall go to Figure 3-13 to process Tearing Analysis.
18	Kernel shall continue to Figure 3-9 to process the GPO / DATA GPO / RESUME GPO command response.

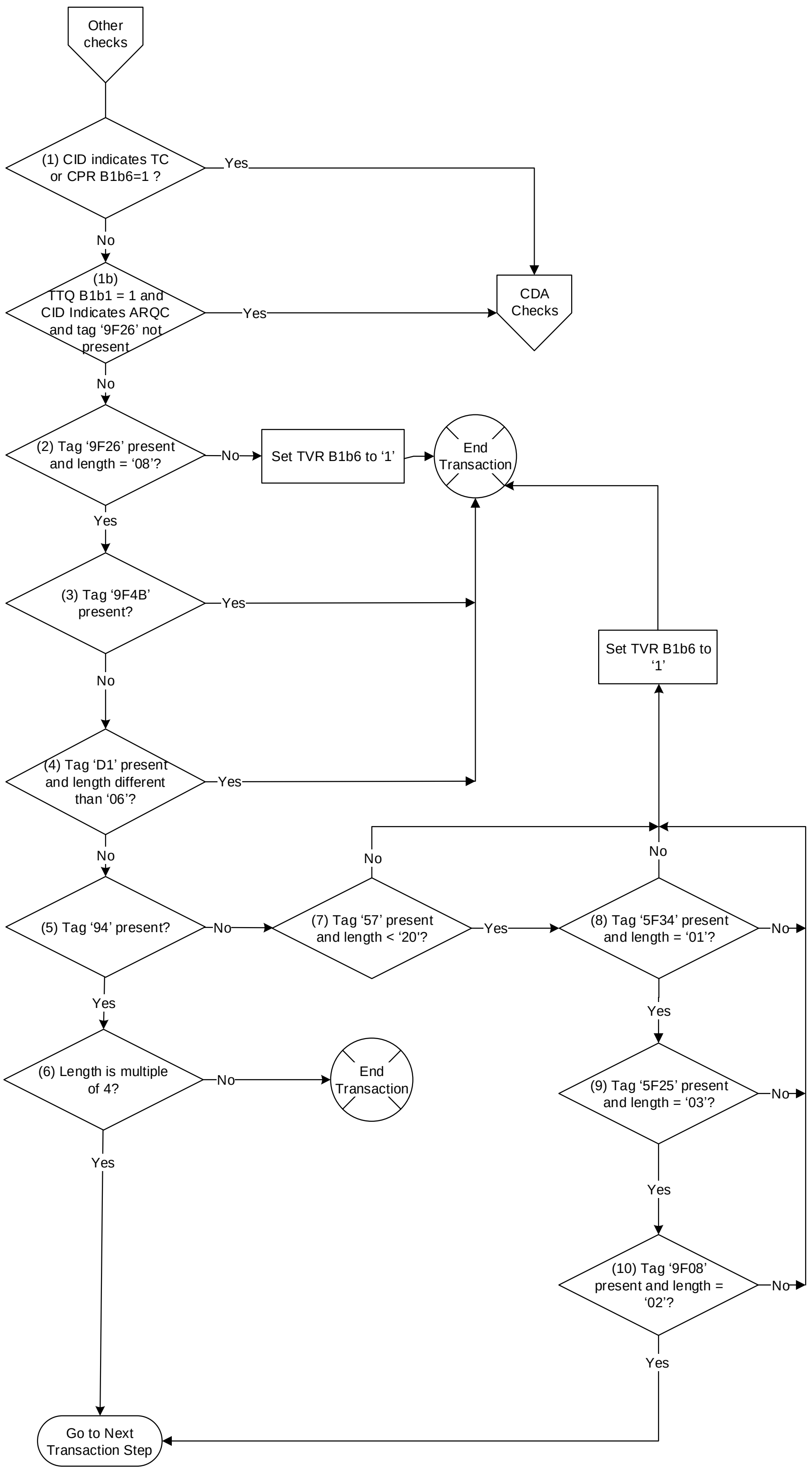
Figure 3-9: Initiate Application Processing (Mandatory Data Checks: All Transactions)


Mandatory Checks – Applicable to All Initiate Application Processing
(1) SW = '9000'?
No
(2) SW = '6984'?
Yes
(3) Send Outcome as '*Try Another Interface*'
No
(3a) SW = '6986' or SW = '6987'?
Yes
(3b) Terminate and restart
No
Yes
(4) Answer well-formatted?
No
Yes
(5) Tag '82' present and length ='02'?
No
Yes
(6) Tag '9F36' present and length = '02'?
No
Yes
(7) Tag '9F10' present?
No
Yes
(8) Tag '9F27' present and length = '01'?
No
Yes
(9) Tag '9F71' present and length = '02'?
No
Yes
(10) Other interface supported?
No
Terminate
Yes
(10a) Send Outcome as '*Try Another Interface*'
(11) Set TVR B1b8 to '1' (Offline data authentication was not performed) and TVR B4b8 to the value of 'Reader Contactless Floor Limit Exceeded' indicator
Other checks

#	Description
1	Kernel shall verify that the command has been successfully processed by the card (by confirming that the Status Word is set to '9000'). If… / Then… Yes: Go to Step 4. No: Go to Step 2.
2	Terminal shall check if the card application requests to use another interface. Note that the card's application has verified that the Terminal has another interface before returning the SW '6984'. • If status word is not equal to '6984', the Terminal shall go to Step 3a. ELSE go to Step 3.
3	The Terminal, the Kernel shall provide the Outcome '***Try Another Interface***' to Entry Point.
3a	Kernel shall check if the card application returns status words '6986' or '6987'. • If status word is not equal to '6986' and not equal to '6986', the Terminal shall go to Step 10 • ELSE go to Step 3b. Note that the SW '6986' only applies to consumer devices when a passcode is not entered and verified. Note that the SW '6987' only applies to consumer devices when a biometric authentication mechanism is not performed and verified.
3b	Terminal shall terminate and restart the existing transaction
4	Kernel shall check if the answer returned by the card is formatted according to [EMV Book 3]. If no, go to Step 10.
5	Kernel shall check if Tag 82 is present. If… / Then… Yes: Go to Step 6. No: Go to Step 10.
6	Kernel shall check if the ATC (tag '9F36') is present and coded on two bytes. If no, go to Step 10.
7	Kernel shall check if application has returned Issuer Application Data (tag '9F10') to the GPO / DATA GPO / RESUME GPO command. If no, go to Step 10.
8	Kernel shall verify that Cryptogram Information Data (CID) (tag '9F27') is present as the answer to the GPO / DATA GPO / RESUME GPO command and is coded on one byte. If no, go to Step 10.
9	Kernel shall check that the application returns Card Processing Requirement (CPR) (tag '9F71') coded on two bytes. If no, go to Step 10.
10	Kernel shall check if it supports another interface. • If Terminal does not support another interface, the Terminal shall terminate the transaction with '***End* Application**' (for Processing Error) Outcome. • ELSE go to Step 9.

#	**Description**
10a	If another interface is supported by the Terminal, the Kernel shall provide the Outcome ‘***Try Another Interface***’ to Entry Point.
11	The Terminal shall set TVR B1b8 = ‘1’ (Offline data authentication was not performed) and set TVR B4b8 to the value of the “Reader Contactless Floor Limit Exceeded” indicator resulting from pre-processing.

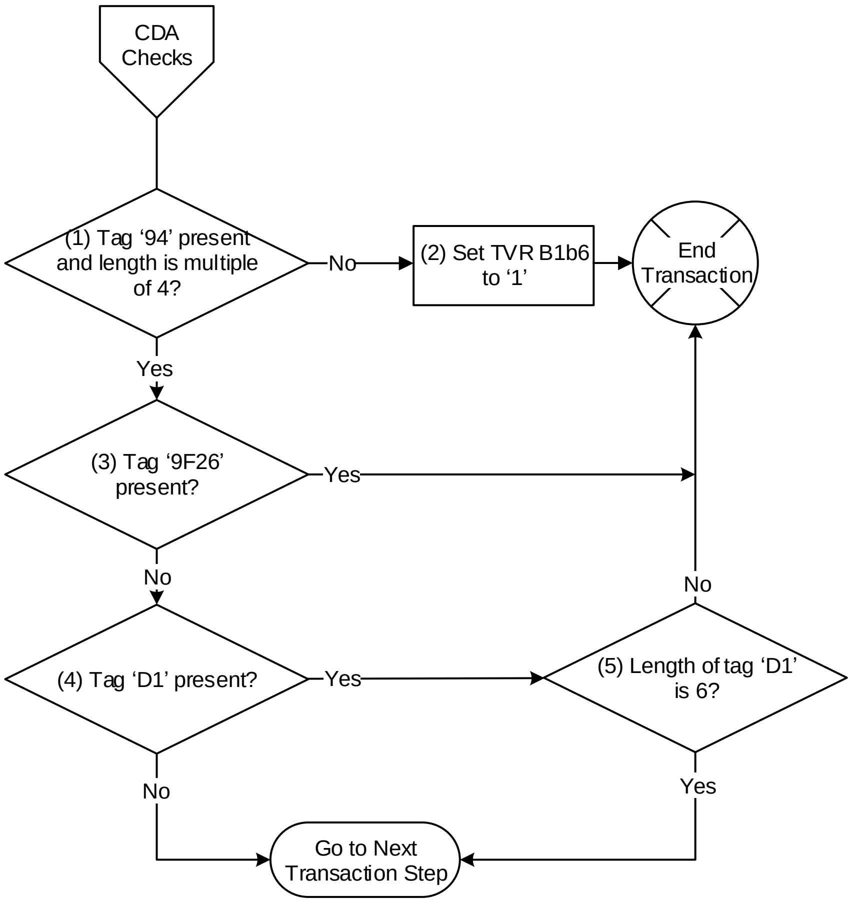
Figure 3-10: Initiate Application Processing (Checks for Online and Decline Decision – No CDA)


Other checks
(1) CID indicates TC or CPR B1b6=1 ?
Yes
No
CDA Checks
(1b) TTQ B1b1 = 1 and CID Indicates ARQC and tag '9F26' not present
Yes
No
(2) Tag '9F26' present and length = '08'?
No
Set TVR B1b6 to '1'
End Transaction
Yes
(3) Tag '9F4B' present?
Yes
No
Set TVR B1b6 to '1'
(4) Tag 'D1' present and length different than '06'?
Yes
No
No
No
(5) Tag '94' present?
No
(7) Tag '57' present and length < '20'?
Yes
(8) Tag '5F34' present and length = '01'?
No
Yes
Yes
(6) Length is multiple of 4?
No
End Transaction
(9) Tag '5F25' present and length = '03'?
No
Yes
Yes
(10) Tag '9F08' present and length = '02'?
No
Yes
Go to Next Transaction Step

#	Description
1	Kernel shall check if the card has returned Cryptogram Information Data (i.e., a Transaction Certificate (TC) or if the Card Processing Requirement (CPR) B1b6 = 1 to indicate that the Program Identifier (PID) Limit is reached. These 2 conditions should result in the checking of CDA. If… | Then… Yes | Complete CDA checks. No | Go to Step 2.
1b	If configured to support Offline Data Authentication for Online Authorizations (TTQ B1b1 is set to '1'), then the Kernel shall check if the card has returned Cryptogram Information Data (CID) indicating that it is ARQC, and that an AC (tag '9F26') is not provided. If… | Then… Yes | Complete CDA checks. No | Go to Step 2.
2	Kernel shall verify that an AC (tag '9F26') is provided by the card and is coded on 8 bytes. If the AC is not present or is not 8 bytes long, the Kernel shall set TVR B1b6 to '1' (ICC Data missing) and end the transaction with '***End Application***' (for Processing Error) Outcome to Entry Point.
3	Kernel shall check if Signed Dynamic Application Data (SDAD) (tag '9F4B') is returned by the application. If SDAD is present in the answer, the Kernel shall end the transaction with '***End Application***' (for Processing Error) Outcome to Entry Point.
4	Kernel shall verify that Offline Balance (tag 'D1') is present in the answer from the card and has a length that is different than 6 bytes. If Offline balance is present and has a length different than '06', the Kernel shall return the transaction with '***End Application***' (for Processing Error) Outcome to Entry Point.
5	Kernel shall check if Application File Locator (AFL) (tag '94') is returned by the card. If… | Then… Yes | Go to Step 6. No | Go to Step 7.
6	Kernel shall check if the AFL is coded on a multiple of 4 bytes. • If AFL does not have a length that is a multiple of 4 bytes, the Kernel shall end the transaction with '***End Application***' (for Processing Error) Outcome. • ELSE go to the next transaction process step.
7	If AFL (tag '94') is not present in the answer of the GPO / DATA GPO / RESUME GPO command, the Kernel shall verify that the answer from the card contains Track 2 Equivalent Data (tag '57') at a length of less than 20 bytes. • If neither Track 2 Equivalent Data (at a length less than 20 bytes) nor the AFL is present in the answer, the Terminal shall set TVR B1b6 to '1' (ICC Data missing) and end the transaction. • ELSE go to Step 8.

#	**Description**
8	Kernel shall verify that the answer from the card contains PAN Sequence Number (tag '5F34') at a length equal to '01'. • If no, set TVR B1b6 to '1' (ICC data missing) and end the transaction with '***End Application***' (for Processing Error) Outcome. • ELSE go to Step 9.
9	Kernel shall verify that the answer from the card contains Application Effective Date (tag '5F25') at a length equal to '03'. • If no, set TVR B1b6 to '1' (ICC data missing) and end the transaction with '***End Application***' (for Processing Error) Outcome. • ELSE go to Step 10.
10	Kernel shall ensure that the answer from the card contains Application Version Number (tag '9F08') at a length equal to '02'. • If no, set TVR B1b6 to '1' (ICC data missing) and end the transaction with '***End Application***' (for Processing Error) Outcome. • ELSE go to the next transaction process step.

Figure 3-11: Initiate Application Processing (Checks for CDA)



#	**Description**
1	Kernel shall check if Application File Locator (AFL) (tag ‘94’) is present in the answer from the card and the associated length is a multiple of 4 bytes.

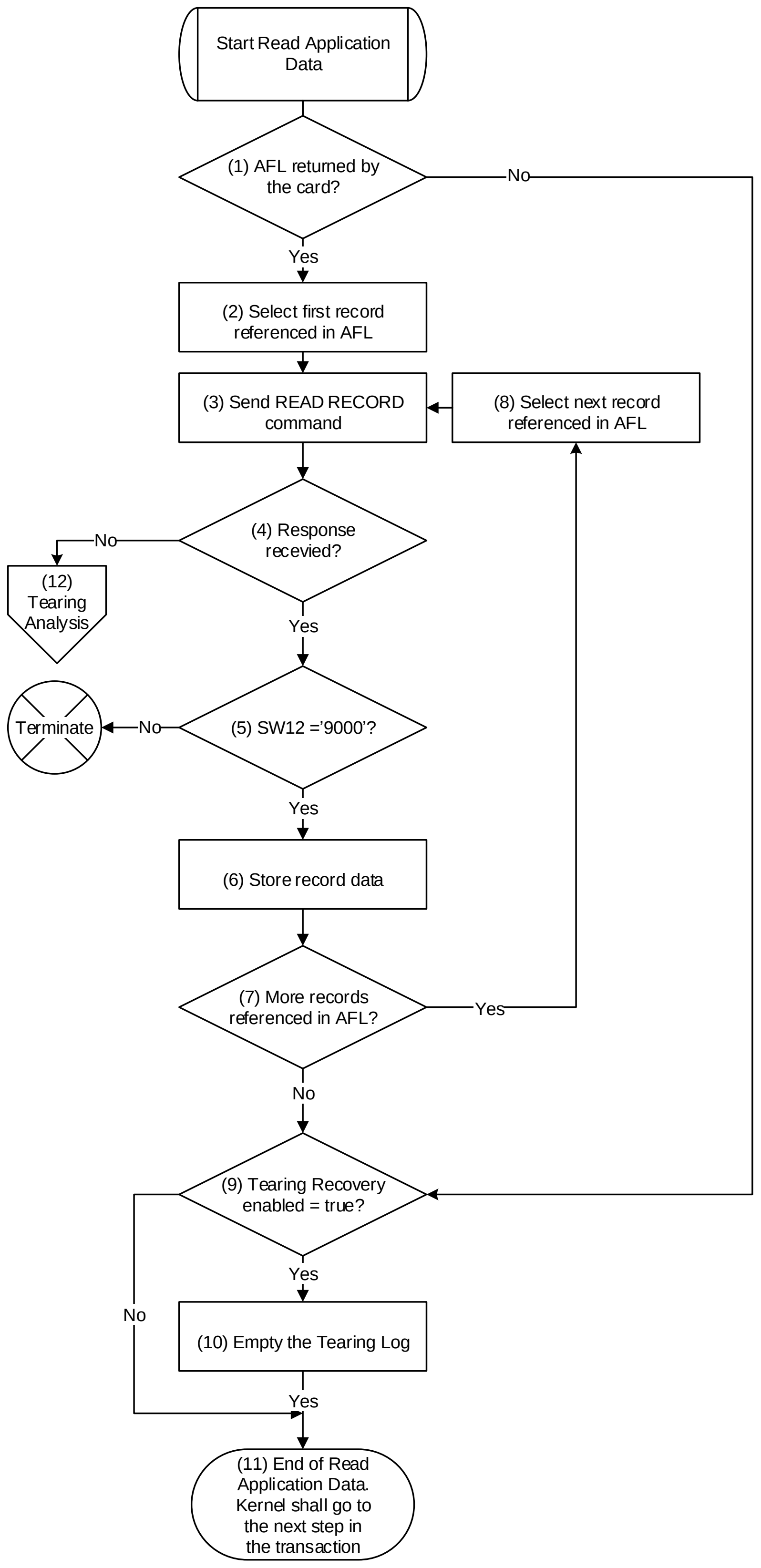
If…	Then…
Yes	Go to Step 3.
No	Go to Step 2.

#	**Description**
2	If AFL is not present or its length is not multiple of 4, the Kernel shall set TVR B1b6 to ‘1’ (ICC data missing) and end the transaction with ‘***End Application***’ (for Processing Error) Outcome.
3	Kernel shall check if Application Cryptogram (AC) (tag ‘9F26’) is returned in the answer to GPO / DATA GPO / RESUME GPO. • If data is present, the Kernel shall end the transaction. • ELSE go to Step 4.
4	Kernel shall verify if Offline Balance (tag ‘D1’) is present in the answer to GPO / DATA GPO / RESUME GPO command. • If data is not present, the Kernel shall end Initiate Application processing and start to perform the next transaction process step. • ELSE go to Step 5.
5	Kernel shall check If the length of Offline Balance has a length of 6 bytes. • If the length is not 6, the Kernel shall end the transaction with ‘***End Application***’ (for Processing Error) Outcome. • ELSE the Kernel shall end Initiate Application processing and start to perform the next transaction process step.

3.4 Read Application Data

The Kernel shall perform Read Application Data if the card returns an Application File Locator (AFL) to the GET PROCESSING OPTIONS (GPO), DATA GET PROCESSING OPTIONS (DATA GPO) or RESUME GET PROCESSING OPTIONS (RESUME GPO) command and data is well-formatted (i.e., the length related to AFL shall be a multiple of 4 bytes). The aim is to read all records as referenced in the AFL.
The Read Application Data process is illustrated in Figure 3-12.

Figure 3-12: Read Application Data Process

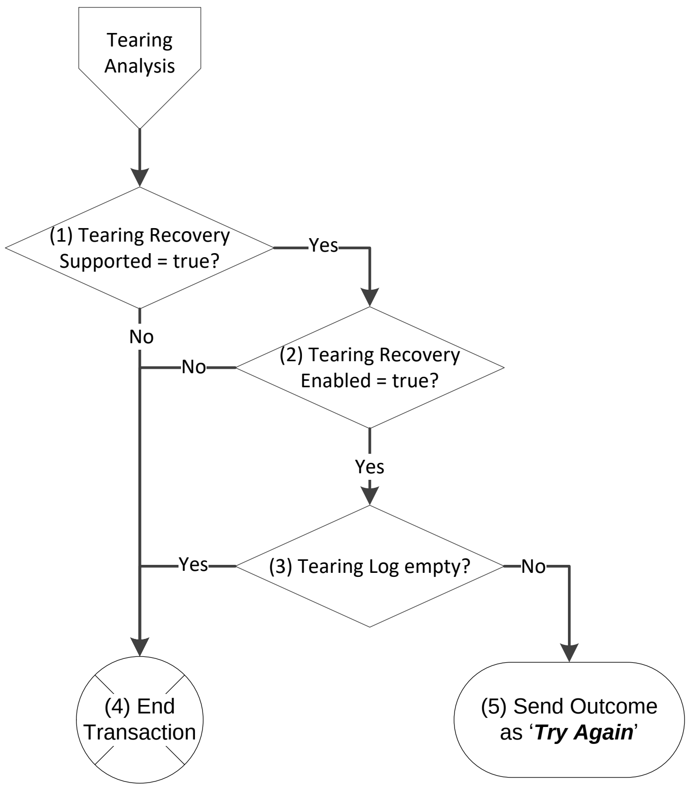

Start Read Application Data
(1) AFL returned by the card?
No
Yes
(2) Select first record referenced in AFL
(3) Send READ RECORD command
(8) Select next record referenced in AFL
(4) Response recevied?
No
(12) Tearing Analysis
Yes
Terminate
No
(5) SW12 ='9000'?
Yes
(6) Store record data
(7) More records referenced in AFL?
Yes
No
(9) Tearing Recovery enabled = true?
Yes
No
(10) Empty the Tearing Log
Yes
(11) End of Read Application Data. Kernel shall go to the next step in the transaction

#	**Description**
1	Terminal shall check if an AFL has been returned in response to the GPO / DATA GPO / RESUME GPO command. (This can be completed by setting an internal flag when verifying the command or by another mechanism.) If… | Then… Yes | Go to Step 2. No | Go to Step 9.
2	Terminal shall select the first record referenced in the AFL.
3	Terminal shall send the READ RECORD command to the card.
4	Terminal shall check that the command is sent and a response received from the card. If… | Then… Yes | Go to Step 5. No | Go to Step 12.
5	Terminal shall check if the response received from the card is ‘9000’ If… | Then… Yes | Go to Step 6. No | Terminate Transaction
6	Terminal shall store the record data in transient memory.
7	Terminal shall check if there are more record to read. If… | Then… Yes | Go to Step 8. No | Go to Step 9.
8	Terminal shall select the next record referenced in the AFL.
9	Terminal shall check if Tearing Recovery is enabled for the current transaction (Tearing Recovery Enabled = true). If… | Then… Yes | Go to Step 10. No | Go to Step 11.
10	Terminal shall empty the Tearing Log.
11	End Read Application. Terminal shall go to the next transaction processing step.
12	Terminal shall go to Figure 3-13 to process Tearing Analysis.

3.5 Tearing Analysis

Tearing Analysis is performed by the Kernel *only if a timeout or transmission error occurs during Initiate Application Processing or Read Application Data*. If Tearing Recovery is possible, the Kernel shall return the '***Try Again***' (for Tearing Recovery) Outcome to request card re-presentment, rather than terminate the transaction, as described in this section.

Figure 3-13: Tearing Analysis


Tearing Analysis
(1) Tearing Recovery Supported = true?
Yes
No
(2) Tearing Recovery Enabled = true?
No
Yes
(3) Tearing Log empty?
Yes
No
(4) End Transaction
(5) Send Outcome as '*Try Again*'

#	Description
1	Kernel shall verify it is configured to support Tearing Recovery (Tearing Recovery Supported = true). If… → Then… Yes → Go to Step 2. No → Go to Step 4.
2	Kernel shall check if Tearing Recovery is enabled for the current transaction (Tearing Recovery Enabled = true). If… → Then… Yes → Go to Step 3. No → Go to Step 4.

#	**Description**
3	Kernel shall check if its Tearing Log is empty If… Then… Yes Go to Step 4. No Go to Step 5.
4	Kernel shall terminate the transaction with '***End Application***' (for Processing Error) Outcome
5	Kernel shall provide the '***Try Again***' (for Tearing Recovery) Outcome to Entry Point.

3.6 Offline Data Authentication (ODA) for Offline Transactions

Offline Data Authentication (ODA) is performed by the Kernel after the Read Application Data step is completed, *only if the application has returned a Signed Dynamic Application Data (SDAD) data element in response to the GET PROCESSING OPTIONS (GPO), DATA GET PROCESSING OPTIONS (DATA GPO), or RESUME GET PROCESSING OPTIONS (RESUME GPO) command*. The Kernel executes the ODA process to authenticate the Contactless Card (i.e., to verify that the card being used is genuine).
Detailed information regarding the ODA process is provided in [EMV Book 2]. During the execution of the process, the Kernel shall:

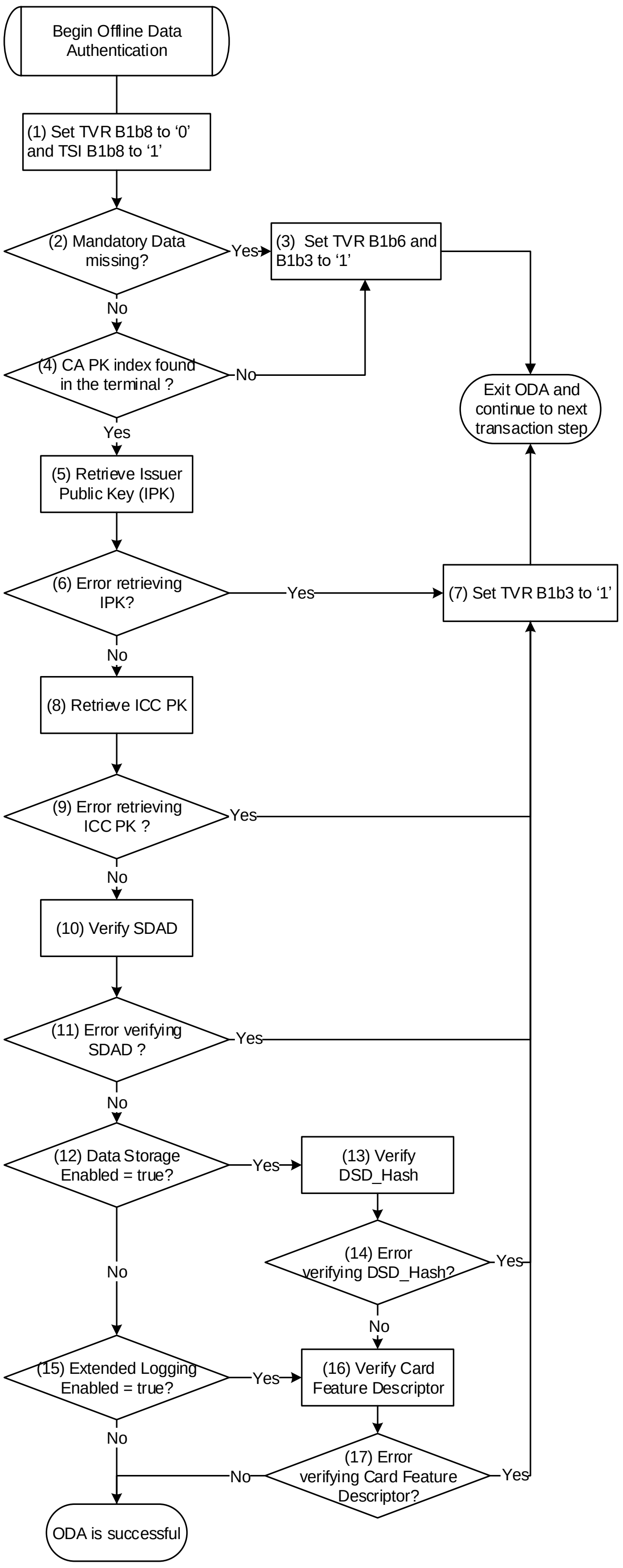
- Retrieve the CA Public Key – associated with the CA Public Key Index (tag '8F)
- Retrieve the Issuer Public Key – based on data read during the Read Application Data step
- Retrieve the ICC Public Key – based on data read during Read Application Data step
- Verify Signed Dynamic Application Data (SDAD tag '9F4B') – in the GPO / DATA GPO / RESUME GPO data returned by the card to the Kernel.

Additionally:

- If the Data Storage Enabled flag is true, the Kernel shall verify that the Data Storage Directory Hash (DSD_Hash) value present in the SDAD matches the value computed during the Read Data Storage step, and
- If either the Data Storage Enabled or Extended Logging Enabled flags are true, the Kernel shall verify that the card identity field(s) of the Card Feature Descriptor (tag 'DF3B') match the corresponding data elements read during the Read Application Data step.

It is recommended that ODA is attempted for online transactions where CDA is supported by the terminal.
Upon completion of the process, the Kernel shall set TVR B1b8 to '0' (Offline data authentication was performed). If ODA fails, then the Kernel will set TVR B1b3 to '1'(CDA failed).
Note: If ODA is not performed, the Kernel shall set TVR B1b8 to '1' (Offline data authentication was not performed).

Figure 3-14: Offline Data Authentication (Global Overview)


Begin Offline Data Authentication
(1) Set TVR B1b8 to '0' and TSI B1b8 to '1'
(2) Mandatory Data missing?
Yes
(3) Set TVR B1b6 and B1b3 to '1'
No
(4) CA PK index found in the terminal ?
No
Exit ODA and continue to next transaction step
Yes
(5) Retrieve Issuer Public Key (IPK)
(6) Error retrieving IPK?
Yes
(7) Set TVR B1b3 to '1'
No
(8) Retrieve ICC PK
(9) Error retrieving ICC PK ?
Yes
No
(10) Verify SDAD
(11) Error verifying SDAD ?
Yes
No
(12) Data Storage Enabled = true?
Yes
(13) Verify DSD_Hash
(14) Error verifying DSD_Hash?
Yes
No
No
(15) Extended Logging Enabled = true?
Yes
(16) Verify Card Feature Descriptor
No
(17) Error verifying Card Feature Descriptor?
No
Yes
ODA is successful

#	Description
1	Kernel shall set TVR B1b8 to '0' (Offline data authentication was performed) and TSI B1b8 to '1' (Offline data authentication was performed).
2	If the mandatory values are... → Then… Missing → Go to Step 3. Present → Go to Step 4.
3	Kernel shall set TVR B1b6 to '1' (ICC Data missing) and B1b3 to '1' (CDA failed). Exit ODA and continue to the next transaction step.
4	Kernel shall verify that the Certificate Authority Public Key Index (CA PKI) is present in the terminal. If CA PKI is... → Then… Missing → Go to Step 3. Present → Go to Step 5.
5	The Kernel shall attempt to retrieve the Issuer Public Key (IPK) from the card.
6	If the IPK is... → Then… Not retrievable → Go to Step 7. Retrievable → Go to Step 8.
7	Kernel shall set TVR B1b3 to '1' ('CDA failed'). Exit ODA and continue to the next transaction step.
8	The Kernel shall attempt to retrieve the ICC PK from the card.
9	If the ICC PK is... → Then… Not retrievable → Go to Step 7. Retrievable → Go to Step 10.
10	Kernel shall verify the signed SDAD as described in the Section 3.6.1.
11	If the SDAD is... → Then… Not verified → Go to Step 7. Verified → Go to Step 12.
12	If Data Storage Enabled = true... → Then… Yes → Go to Step 13. No → Go to Step 15
13	Kernel shall verify that a DSD_Hash value is present in the ICC Dynamic Data and that it matches the value computed during Read Data Storage processing.
14	If the DSD_Hash is... → Then… Not verified → Go to Step 7. Verified → Go to Step 16

#	Description	
15	If Extended Logging Enabled = true...	Then…
	Yes	Go to Step 16.
	No	ODA successful. Go to the next transaction step.
16	Kernel shall verify the Card Feature Descriptor as described in the Section 3.6.2.	
17	If the Card Feature Descriptor is...	Then…
	Not verified	Go to Step 7.
	Verified	ODA successful. Go to the next transaction step.

3.6.1 Signed Dynamic Application Data

Verifying Signed Dynamic Application Data shall only be performed if the Certificate Authority Public Key, Issuer Public Key and ICC Public Key are retrieved successfully. The aim is to authenticate the card.

To complete this authentication, the Kernel shall perform following checks:

- Verify the SDAD size: the SDAD shall have the same length as the ICC Public Key Modulus (if not, CDA is considered as failing).
- Apply the RSA algorithm on SDAD data using the ICC Public Key. Deciphered data shall be formatted as specified in [EMV Book 2].
- Check the following elements on deciphered data:
    - Recovered Data Header set to ‘6A’.
    - Recovered Data Trailer set ‘BC’.
    - Signed Data Format set to ‘05’.
- Verify that ICC Dynamic Data are present and formatted as follows:
    - ICC Dynamic Data shall be present in deciphered data starting at byte 5, and the length of ICC Dynamic Data is given by byte 4 of deciphered data with:
        - ICC Dynamic Number Length = 1 byte
        - ICC Dynamic Number = 2-8 bytes
        - Cryptogram Information Data (CID) = 1 byte
        - Transaction Certificate or ARQC = 8 bytes
        - Transaction Data Hash Code = 20 bytes
        - Data Storage Directory Hash (DSD_Hash) = 20 bytes (*conditional: present only if data storage is enabled for the transaction*)
    - Verify that the CID found in deciphered data is the same as the one returned in GET PROCESSING OPTIONS under tag ‘9F27’
    - Create the Hash Result: Kernel shall compute the hash applying the Hash Indicator Algorithm to the following data in the order presented: from the Signed Data Format to the pad pattern followed by the Unpredictable Number sent by the Kernel.

- Perform a comparison between the Hash Result recovered from deciphered data and Hash Result previously computed. If they do not match, the Kernel shall consider that CDA verification is failing, update the TVR, and skip the remainder of the CDA process.
- Create Transaction Data Hash Code: Kernel shall concatenate the following elements: PDOL values (sent by the Kernel) and TLV data returned by the card to GPO command in the order they are returned, with the exception of the Signed Dynamic Application Data.
- Perform a comparison between the Transaction Data Hash Code computed by the Kernel and Transaction Data Hash Code recovered from deciphered data. If they do not match, the Kernel shall consider that CDA verification is failing, update the TVR, and shall skip the remainder of the CDA process.

If no issues have been found, SDAD verification was successful. At the end of this step, if no issues have been found, the Kernel shall store the Application Cryptogram contained in the ICC Dynamic Data recovered in [EMV Book 2] Table 19 in tag '9F26'.

3.6.2 Card Feature Descriptor Verification

Verifying the Card Feature Descriptor (tag 'DF3B') shall only be performed *if Data Storage or Extended Logging are enabled for the transaction*. The aim is to authenticate the unique card identity encoded in the Card Feature Descriptor.

To complete this verification, the Kernel shall perform the following checks to compare the Card Feature Descriptor against the Card ID data element read during the Read Application Data step:

- Check that the Card ID (tag 'DF3E') data element is present
- Check that the Card Feature Descriptor's Card ID field (with reference to Table 4-18) matches the value of the Card ID data element (tag 'DF3E').

If no issues have been found, Card Feature Descriptor verification was successful.

3.7 Cardholder Verification

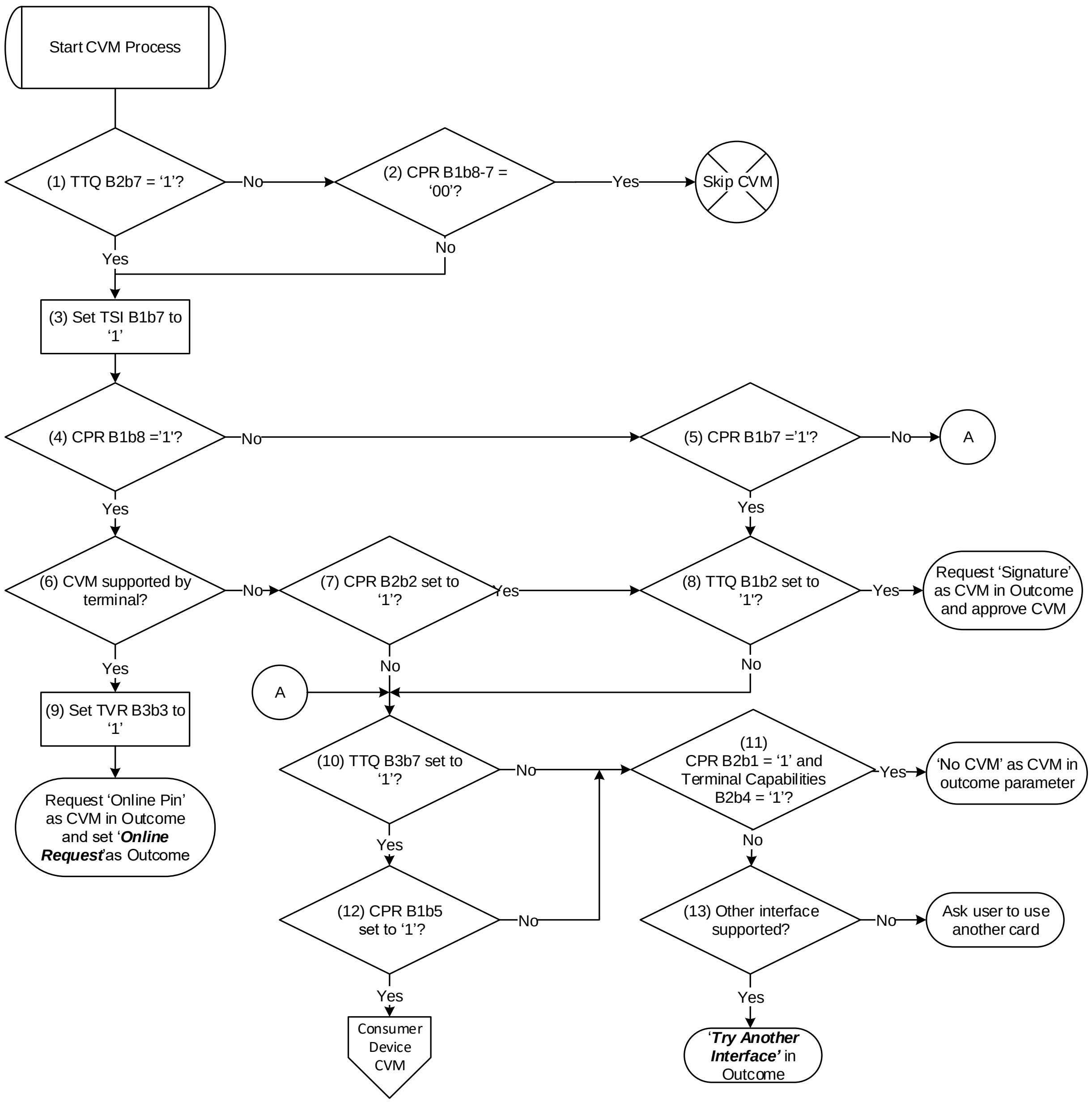
The Kernel may perform Cardholder verification based on the reader configuration and card request. The CVMs supported by the Kernel are:

- Online PIN
- Signature
- No CVM, and
- Consumer Device CVM (CD CVM) – a CVM performed on, and validated by, the consumer's payment device, independent of the reader.

The following flow diagrams describe the CVM processing steps for:

- Online PIN, Signature, and No CVM in Figure 3-15: Cardholder Verification Method Process (Online PIN, Signature, and No CVM), and
- CD CVM: Figure 3-16.

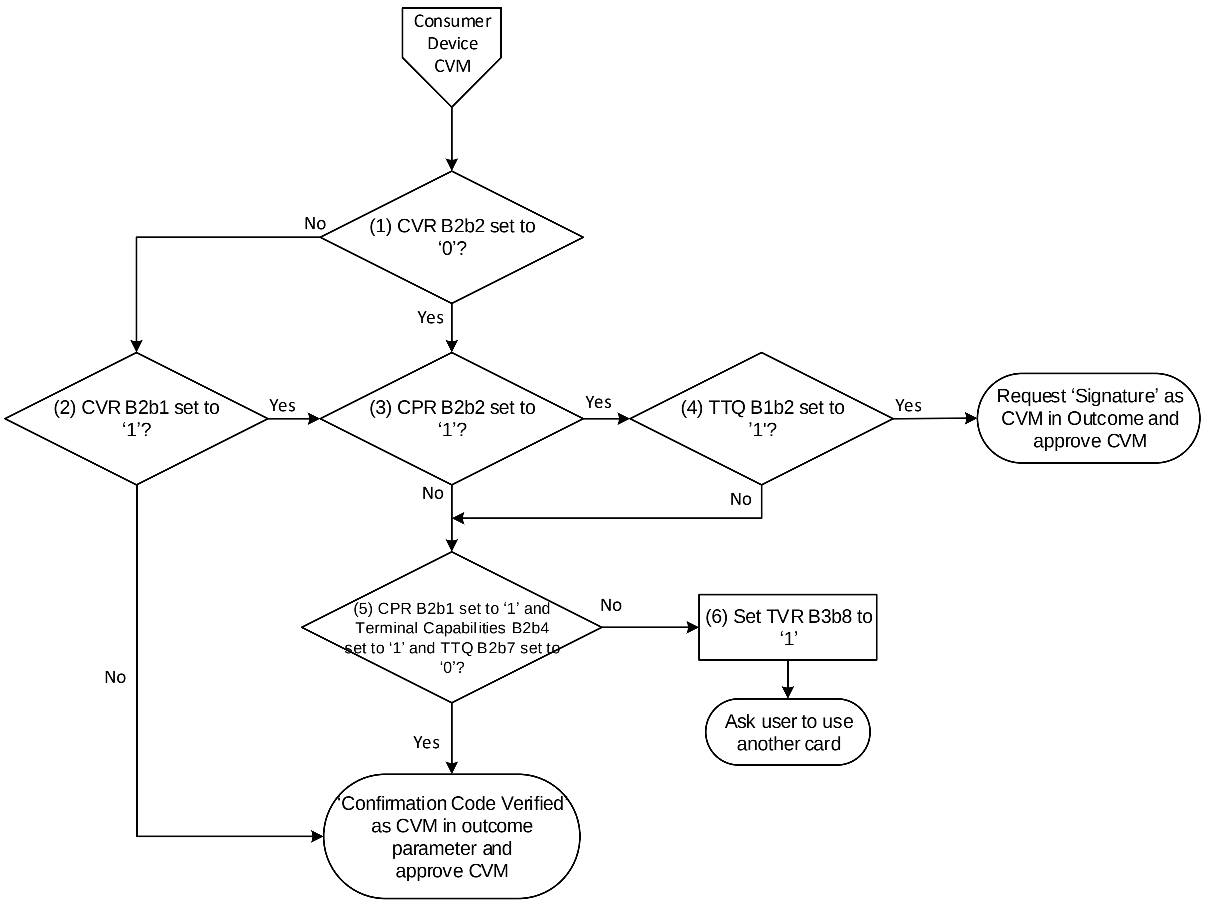
Figure 3-15: Cardholder Verification Method Process (Online PIN, Signature, and No CVM)


Start CVM Process
(1) TTQ B2b7 = '1'?
No
(2) CPR B1b8-7 = '00'?
Yes
Skip CVM
Yes
No
(3) Set TSI B1b7 to '1'
(4) CPR B1b8 ='1'?
No
(5) CPR B1b7 ='1'?
No
A
Yes
Yes
(6) CVM supported by terminal?
No
(7) CPR B2b2 set to '1'?
Yes
(8) TTQ B1b2 set to '1'?
Yes
Request 'Signature' as CVM in Outcome and approve CVM
Yes
No
No
(9) Set TVR B3b3 to '1'
A
Request 'Online Pin' as CVM in Outcome and set '*Online Request*' as Outcome
(10) TTQ B3b7 set to '1'?
No
(11) CPR B2b1 = '1' and Terminal Capabilities B2b4 = '1'?
Yes
'No CVM' as CVM in outcome parameter
Yes
No
(12) CPR B1b5 set to '1'?
No
(13) Other interface supported?
No
Ask user to use another card
Yes
Yes
Consumer Device CVM
'*Try Another Interface'* in Outcome

#	Description		
1	Kernel shall check if the Terminal has requested the Kernel to process a CVM (TTQ B2b7 = 1 CVM Required).		
		If…	Then…
		Yes	Go to Step 3.
		No	Go to Step 2.
2	Kernel shall verify if the card has not requested the Terminal to process an Online or Signature CVM (CPR B1b8-7 = ‘00’ – bit 8 Online PIN required, bit 7 Signature required).		
		If…	Then…
		Yes	The Kernel shall end the CVM process, and go to the next transaction step.
		No	Go to Step 3.
3	CVM will be performed. The Terminal shall update the Transaction Status Information (TSI) to indicate that CVM will be processed (B1b7 = 1 Cardholder verification was performed).		
4	Kernel shall check if the card requests an Online PIN verification (CPR B1b8 = 1 Online PIN required).		
		If…	Then…
		Yes	Go to Step 6.
		No	Go to Step 5.
5	Kernel shall verify if the card has requested a signature (CPR B1b7 = 1 Signature required).		
		If…	Then…
		Yes	Go to Step 8.
		No	Go to Step 10.
6	Kernel shall verify that it supports Online PIN (TTQ B1b3 =1 Online PIN supported).		
		If…	Then…
		Yes	Go to Step 9.
		No	Go to Step 7.
7	Kernel shall verify if the card supports fallback to signature (CPR B2b2 = 1 CVM Fallback to Signature allowed).		
		If…	Then…
		Yes	Go to Step 8.
		No	Go to Step 10.

#	Description	
8	Kernel shall check if it supports Signature as CVM (TTQ B1b2 =1 Signature supported).	
	If…	Then…
	Yes	The Kernel shall request ‘Obtain Signature’ in the CVM Outcome parameter and request cardholder to provide a signature.
	No	Go to Step 10.
9	When processing the Online PIN as CVM method, the Kernel shall update TVR B3b3 to ‘1’ (Online PIN entered). Kernel shall request ‘Online PIN’ as the CVM outcome parameter and set Online Request as Outcome.	
10	Online PIN and signature are not requested. The Kernel shall check if it supports Confirmation Code. (TTQ B3b7 = 1 Consumer Device CVM (CD CVM) supported).	
	If…	Then…
	Yes	Go to Step 12.
	No	Go to Step 11.
11	Online PIN, signature and confirmation code are not allowed. The Kernel shall check: • if the Card Processing Requirements (9F71) B2b1 is set to ‘1’(“CVM Fallback to No CVM Allowed”) • and if the Terminal Capabilities (9F33) B2b4 is set to ‘1’ (“No CVM Required”).	
	If…	Then…
	Yes	The Kernel shall request ‘No CVM’ in the CVM outcome parameter and approve the CVM. (End Card Verification Method process. Go to the next transaction step in the process.)
	No	Go to Step 13.
12	Kernel shall check if a Confirmation Code has been performed (CPR B1b5 = 1 Consumer Device CVM Performed).	
	If…	Then…
	Yes	Process CD CVM. (End Card Verification Method process. Go to the next transaction step.).
	No	Go to Step 11.
13	Kernel shall check if the terminal supports another interface. • If the Kernel does not support another interface, the Kernel shall ask for another card and sends ‘***End Application***’ (for Processing Error) Outcome. • ELSE the Kernel shall send ‘***Try Another Interface***’ as Outcome.	

Figure 3-16: Cardholder Verification Method Process: Consumer Device CVM (CD CVM)


Consumer Device CVM
(1) CVR B2b2 set to '0'?
No
Yes
(2) CVR B2b1 set to '1'?
Yes
(3) CPR B2b2 set to '1'?
Yes
(4) TTQ B1b2 set to '1'?
Yes
Request 'Signature' as CVM in Outcome and approve CVM
No
No
(5) CPR B2b1 set to '1' and Terminal Capabilities B2b4 set to '1' and TTQ B2b7 set to '0'?
No
(6) Set TVR B3b8 to '1'
Ask user to use another card
No
Yes
'Confirmation Code Verified' as CVM in outcome parameter and approve CVM

#	Description	If…	Then…
1	Kernel shall check if the CDCVM was not performed CVR B2b2 = 0 (Confirmation Code Verification was not performed).	Yes	Go to Step 3.
		No	Go to Step 2.
2	Kernel shall check if CDCVM verification was unsuccessful CVR B2b1 = 1 (Confirmation Code Verification performed and failed).	Yes	Go to Step 3.
		No	Approve the CVM with 'Confirmation Code Verified' in the CVM Outcome parameter. (End Card Verification Method process. Go to the next transaction step.).
3	Kernel shall check if the card allows the processing of the signature (CPR B2b2 = 1 CVM Fallback to Signature allowed).	Yes	Go to Step 4.
		No	Go to Step 5.
4	Kernel shall check if signature is supported (TTQ B1b2 =1).	Yes	Kernel shall prompt the user to sign the receipt. (End Card Verification Method process. Go to the next transaction step.)
		No	Go to Step 5.
5	Kernel shall check: • if the card allows "No CVM" (CPR B2b1 = 1 CVM Fallback to No CVM allowed) • and if the Terminal Capabilities (9F33) B2b4 is set to '1' ("No CVM Required") • and Terminal Transaction Qualifiers (9F66) B2b7 = '0' ("CVM Not Required").	Yes	The kernel shall allow 'No CVM' as the CVM outcome parameter and approve the CVM. (End Card Verification Method process. Go to the next transaction step.).
		No	Go to Step 6.
6	Kernel shall indicate that CVM has failed by setting TVR B3b8 to '1' (Cardholder verification was not successful) and shall ask cardholder to use another card. Send 'End Application' (for Processing Error) as the Outcome. (End Card Verification Method process. Go to the next transaction step.).		

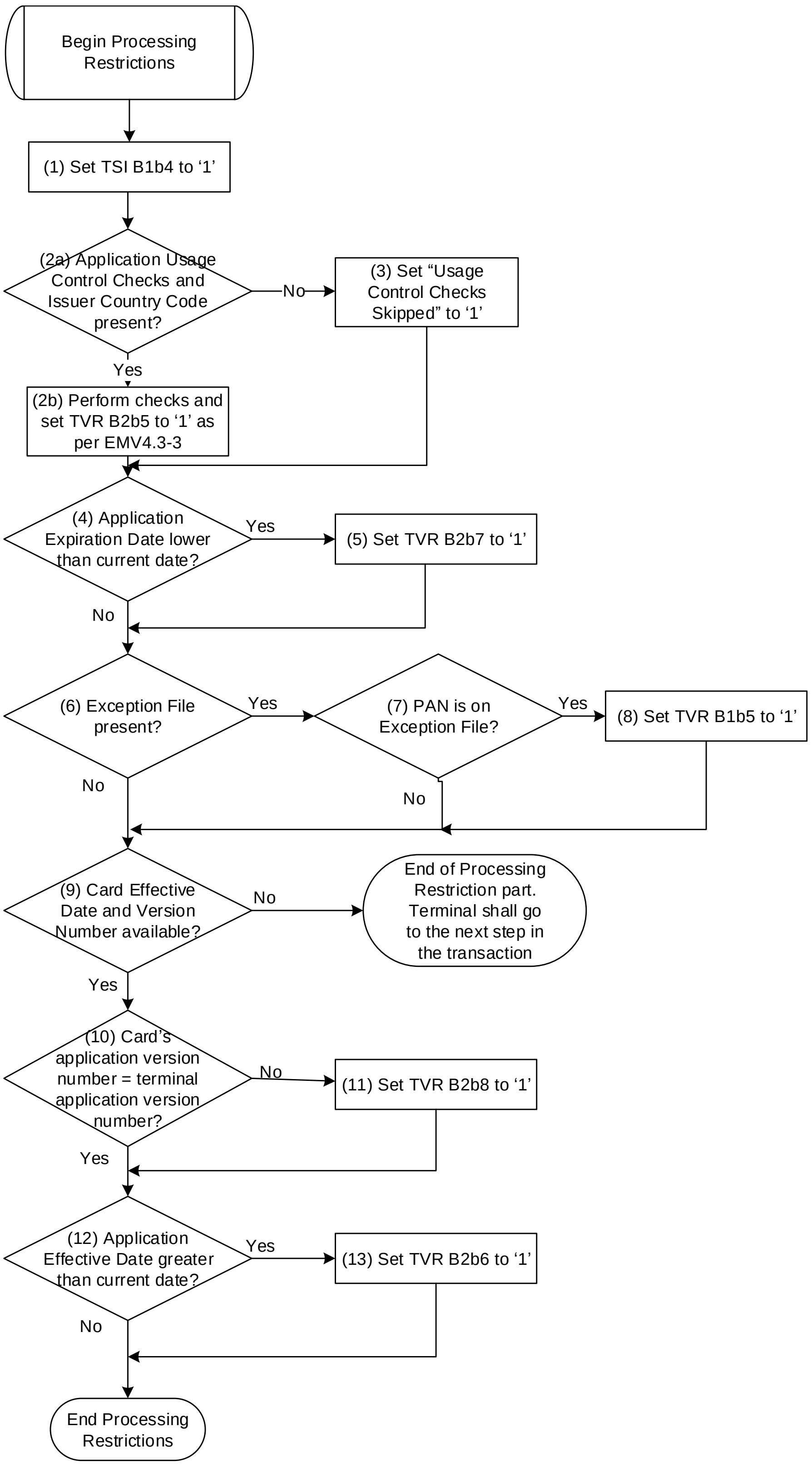
3.8 Processing Restrictions

The processing restrictions function must be performed by the Kernel using the data elements retrieved from the card, as described in [EMV Book 3]. The following checks must be performed by the Kernel (based on the data provided by the card):

- **Application Expiration Date:** The Terminal shall read the Expiration Date from tag 5F24 if present, otherwise extract it from the Track 2 Equivalent Data, and compare that value with the date of the transaction. If the date of the transaction is greater than the expiration date found in the Track 2 Equivalent Data, the Terminal shall set the TVR B2b7 to '1' (Expired application).
- **Application Effective Date:** The Terminal shall check if the application effective date is greater than the current date. If the application effective date is greater than the current date, the Terminal shall set TVR B2b6 to '1' (Application not yet effective).
- **Application Version Number:** The Terminal shall compare its application version number against the one read in the card. If they do not match, the Terminal shall set the TVR B2b8 to '1' (ICC and Terminal have different application versions).
- **Application Usage Control:** The terminal shall check for restrictions limiting the application geographically or relative to certain types of transactions. If the usage conditions are not met, the Kernel shall set the TVR B2b5 to '1' (Requested service not allowed for card product).
- **Exception File:** The Terminal shall have an internal file that references a special PAN (for example, it can be a blacklist PAN to indicate which PANs shall generate a decline transaction). If a Terminal has this type of file, it shall check that the PAN read is not present in this file. If present, the Terminal shall to set TVR B1b5 to '1' (Card appears on Terminal exception file).

Figure 3-17 describes how the Processing Restriction step shall be performed.

Figure 3-17: Processing Restriction Process


Begin Processing Restrictions
(1) Set TSI B1b4 to '1'
(2a) Application Usage Control Checks and Issuer Country Code present?
No
(3) Set "Usage Control Checks Skipped" to '1'
Yes
(2b) Perform checks and set TVR B2b5 to '1' as per EMV4.3-3
(4) Application Expiration Date lower than current date?
Yes
(5) Set TVR B2b7 to '1'
No
(6) Exception File present?
Yes
(7) PAN is on Exception File?
Yes
(8) Set TVR B1b5 to '1'
No
No
(9) Card Effective Date and Version Number available?
No
End of Processing Restriction part. Terminal shall go to the next step in the transaction
Yes
(10) Card's application version number = terminal application version number?
No
(11) Set TVR B2b8 to '1'
Yes
(12) Application Effective Date greater than current date?
Yes
(13) Set TVR B2b6 to '1'
No
End Processing Restrictions

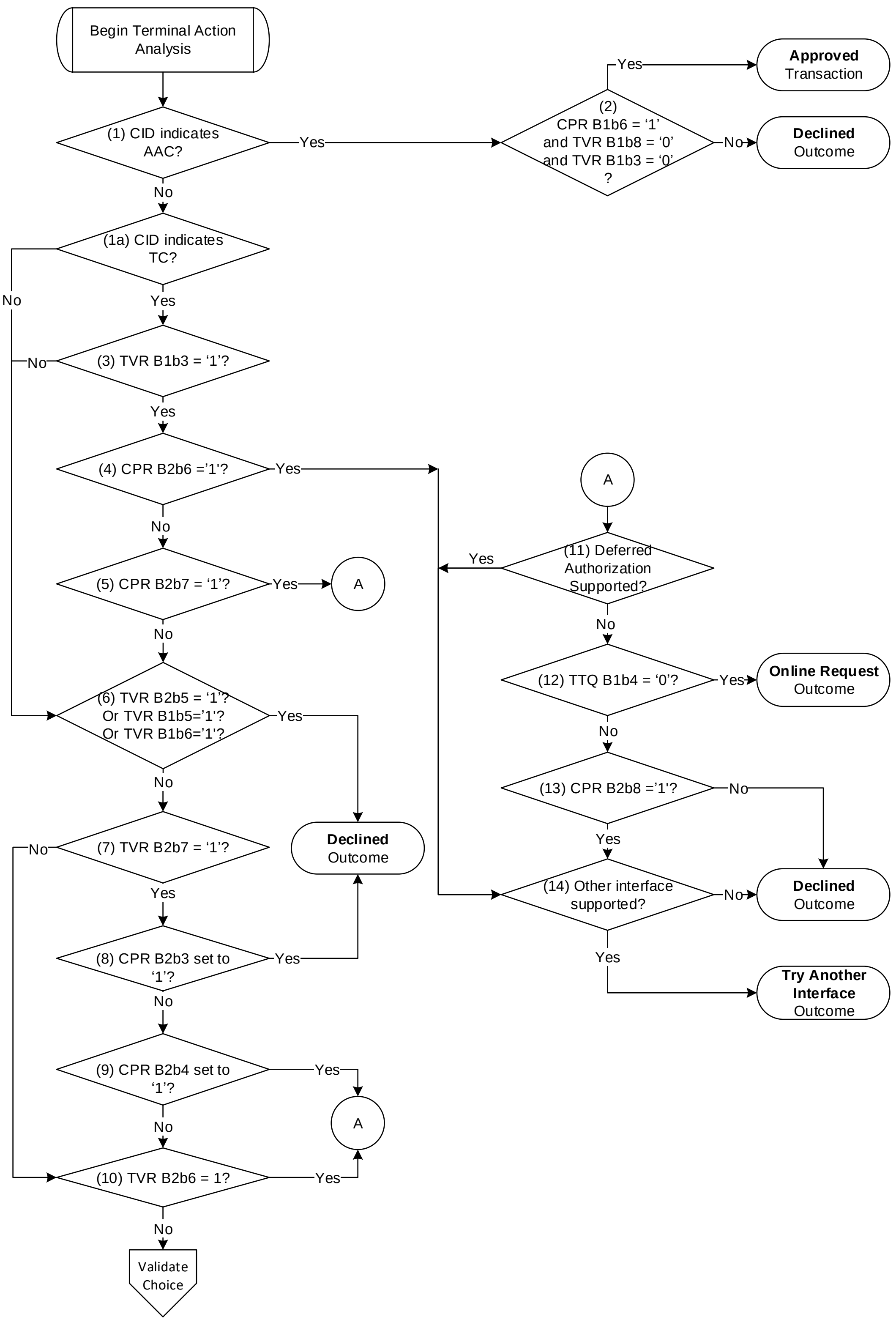
#	Description
1	Kernel shall set "Transaction Status Information" (tag '9B') B1b4 to '1' Terminal risk management was performed.
2a	Kernel shall check if the Application Usage Control (AUC) and Issuer Country Code are present**. If… → Then… Yes → Go to Step 1b. No → Go to Step 2. **For online transactions, if either the AUC or Issuer Country Code is not present, the AUC check is skipped. For offline transactions, the AUC check is mandatory, as these two fields are read from the file records.
2b	Terminal shall perform the usage control checks as described in [EMV Book 3] to determine if the service is allowed and set TVR B2b5 to '1' (Requested service not allowed for card product) if any usage restrictions are applicable to the transaction.
3	If Application Usage Control or Issuer Country Code are missing, the Terminal shall set the internal data element 'Usage Control Checks Skipped' to '1'.
4	Kernel shall determine if the card's application has expired. * If… → Then… Yes → Go to Step 4. No → Go to Step 5. 1. ***Note**: There are two ways to retrieve the Application Expiration Date, including via the: "Track 2 Equivalent Data". 2. Application Expiry date (tag '5F24'), if provided by the card. (This should match the date in the track data.)
5	If the application has expired, the Terminal shall set TVR B2b7 to '1' (Expired application).
6	Kernel shall check if it embeds an exception file that contains a list of PAN(s). If… → Then… Yes → Go to Step 6. No → Go to Step 8.
7	Kernel shall verify if the PAN returned by the card is found inside the exception file. If… → Then… Yes → Go to Step 7. No → Go to Step 8.
8	Kernel shall set TVR B1b5 to '1' (Card appears on Terminal exception file).

#	Description
9	Kernel shall determine if the card's Application Effective Date (tag '5F25') and Application Version Number (tag '9F08') are available. If… → Then… Yes → Go to Step 9. No → End the Processing Restriction check. Go to the next transaction step.
10	Kernel shall check if the card's Application Version Number (tag '9F08') matches the Terminal's Application Version Number (9F09''). If… → Then… Yes → Go to Step 11. No → Go to Step 10. **Note**: The card's Application Version Number shall be present in one record under tag '9F08'.
11	Kernel shall update TVR B2b8 to '1' (ICC and Terminal have different application versions).
12	Kernel shall check if the Application Effective Date (tag '5F25') returned by the card is older than current date. If… → Then… Yes → Go to Step 12. No → End the Processing Restriction Process. **Note**: Application Effective Date shall be present in one record under tag '5F25'
13	Kernel shall set TVR B2b6 to '1' (Application not yet effective) and exit the processing restrictions step.

3.9 Terminal Action Analysis

Terminal Action Analysis is a mandatory step performed by the Kernel. The objective of this step is to complete additional checks on the Kernel side to make a final decision concerning the current transaction.
The Kernel shall not perform this step if the decision taken by the card is to decline the transaction. The Kernel shall check if some TVR bits are set and then (if they are set) shall determine the card recommendations. The following diagrams show the Terminal Action Analysis to be performed.
The Kernel shall include Data Record Outcome Parameters according to each Outcome and as described in Annex B.11.
If Data Storage Enabled flag is true, and the chosen Outcome is not '***Try Another Interface***', then the Kernel shall include Discretionary Data Outcome Parameters as described in Annex B.12.

Figure 3-18: Terminal Action Analysis Process

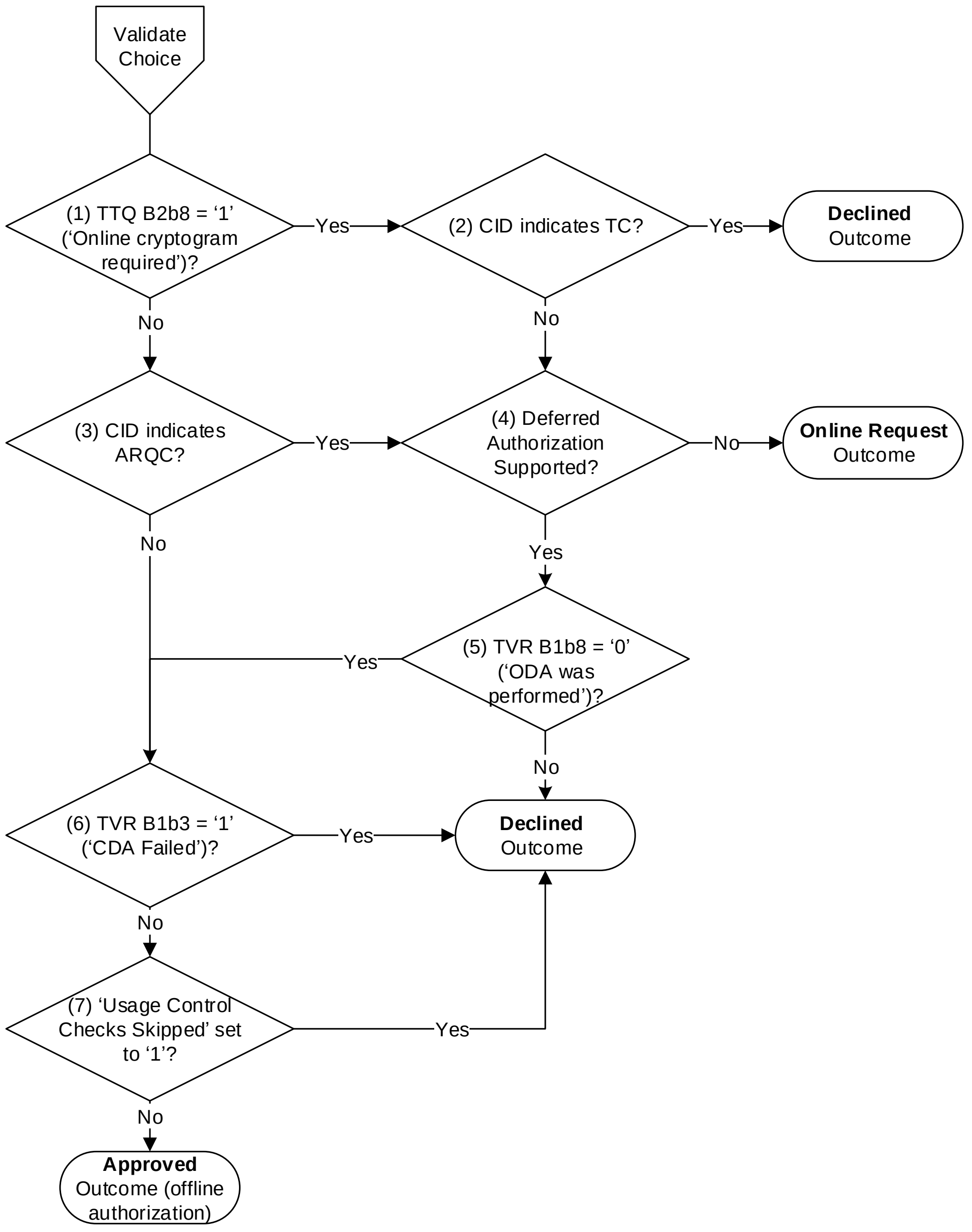


#	**Description**		
1	Kernel shall check if the card returns a decline transaction response based on the CID setting [i.e., CID B1b8-7 to '00' (00 = AAC)].	If…	Then…
		Yes	Go to Step 2.
		No	Go to Step 1a.
1a	Terminal shall check if the card wants to approve the transaction (based on the CID indicating a TC)	If…	Then…
		Yes	Go to Step 3.
		No	Go to Step 6.
2	Kernel shall verify if: • the "Card Processing Requirements" (CPR tag '9F71') B1b6 (PID Limit reached - Loyalty Transaction approved) is set to '1'*, AND • TVR B1b8 is set to '0' (ODA was performed), AND • TVR B1b3 is set to '0' (CDA was successful). **Note**: If these conditions are met then this is an approved loyalty program transaction (i.e., a free transaction for the cardholder).	If…	Then…
		Yes	The Kernel shall approve the transaction. End the Terminal Action Analysis. Go to the next transaction step.
		No	The Kernel shall decline the transaction as requested by the card. End the Terminal Action Analysis. Go to the next transaction step.
3	Kernel shall check if CDA has failed [TVR B1b3 = '1' (CDA failed)].	If…	Then…
		Yes	Go to Step 4.
		No	Go to Step 6.
	Note: When the Kernel processes the TVR, the following applies: If CDA has not been processed or CDA is successful, the Kernel shall skip the card's recommendation and go to Step 5. • If CDA is failing, go to Step 3.		
4	Kernel shall perform a CPR check to determine if the card allows a switch to another interface or decline of the transaction if CDA is failing [CPR B2b6 = '1' (Decline/switch other interface if CDA failed).	If…	Then…
		Yes	Go to Step 14.
		No	Go to Step 5.

#	Description
5	Kernel shall go online for authorization if CDA is failing [CPR B2b7 = '1' (Process online if CDA failed)]. <table><tr><td>If…</td><td>Then…</td></tr><tr><td>Yes</td><td>Go to Step 11.</td></tr><tr><td>No</td><td>Go to Step 6.</td></tr></table>
6	Kernel shall check if: • Application usage is not allowed [TVR B2b5 = '1' (Requested service not allowed for card product)], or • The card appears in the exception file [TVR B1b5 = '1' (Card appears on Terminal exception file)], or • ICC data missing bit is set [TVR B1b6 = '1' (ICC Data missing)]. If any one of the foregoing conditions is true, the Kernel shall decline the transaction with 'Declined' Outcome (End the Terminal Action Analysis). **ELSE** go to Step 7.
7	Kernel shall check if the application has expired (TVR B2b7 Expired application). • If not expired, the Kernel shall go to Step 10. • If expired, go to Step 8.
8	Kernel shall check if it should decline the transaction when application has expired: • If CPR B2b3 is set to '1' (Decline if card expired), the Kernel shall decline the transaction with '***Declined***' Outcome (End the Terminal Action Analysis). • ELSE, go to Step 9.
9	Kernel shall check if it should process the transaction online when application has expired [CPR B2b4 = '1' (Process online if card expired)]. • If CPR B2b4 is set, the Kernel shall go to Step 11. • If it is not set, go to Step 10.
10	Kernel shall check if TVR B2b6 is set to '1' (Application is not yet effective). • If yes, the Kernel shall try to process transaction online or request another interface by going to Step 11. • Else, Kernel shall go to the second part of Terminal Action Analysis.
11	Kernel shall check if it is configured for Deferred Authorizations (true if Deferred Authorization Supported flag is present and set to '1'). • If yes, the Kernel shall go to Step 14. • Else it will go to Step 12.

#	Description
12	Kernel shall check if it is configured to go online for authorization (TTQ B1b4 ='0' Offline-only Reader). • If Terminal is able to go online [TTQ B1b4 ='0' (Not Offline-only Reader)], then it shall process the online authorization with the Outcome as '***Online Request***'. (End the Terminal Action Analysis. Go to the next transaction step.) • ELSE it will go to Step 13.
13	Kernel shall check if the card allows a switch to another interface, if the Terminal is not able to go online [CPR B2b8 = '1' (Switch other interface if unable to process online)]. • If the card does not allow a switch to another interface, the Kernel shall decline the transaction with '***Declined***' Outcome. (End the Terminal Action Analysis. Go to the next transaction step). • ELSE it will go to Step 14.
14	Kernel shall verify if it supports another interface that can process the transaction. • If the Terminal has another interface, it shall send the '***Try Another Interface***' Outcome. (End the Terminal Action Analysis. Go to the next transaction step.) • ELSE it shall decline the transaction and send the Outcome as '***Declined***' (End the Terminal Action Analysis. Go to the next transaction step.)

Figure 3-19: Terminal Action Analysis Process (Validate Choice)


Validate Choice
(1) TTQ B2b8 = '1' ('Online cryptogram required')?
Yes
(2) CID indicates TC?
Yes
Declined Outcome
No
No
(3) CID indicates ARQC?
Yes
(4) Deferred Authorization Supported?
No
Online Request Outcome
No
Yes
(5) TVR B1b8 = '0' ('ODA was performed')?
Yes
No
(6) TVR B1b3 = '1' ('CDA Failed')?
Yes
Declined Outcome
No
(7) 'Usage Control Checks Skipped' set to '1'?
Yes
No
Approved Outcome (offline authorization)

#	Description
1	Kernel shall verify if an online cryptogram is required [TTQ B2b8 = '1' (Online Cryptogram required)]. If… / Then… Yes / Go to Step 2. No / Go to Step 3.

#	Description
2	Kernel shall verify if the CID indicates TC based on the CID setting [i.e., CID B1b8-7 to ‘01’ (01 = TC)]. • If the card has approved the transaction when the Kernel asked to go online (i.e., the CID indicates TC), the Kernel shall send a ‘***Declined***’ Outcome (End the Terminal Action Analysis). • Else, go to Step 4.
3	Kernel shall verify if the card requests an online authorization (i.e., CID indicates ARQC, with CID B1b8-7 = ‘10’). If… | Then… Yes | Go to Step 4. No | Go to Step 6.
4	Kernel shall verify if it is configured for Deferred Authorizations (true if Deferred Authorization Supported flag is present and set to ‘1’). • If it is not configured for Deferred Authorizations, the Kernel shall send an ‘***Online Request***’ Outcome (End the Terminal Action Analysis). • Else, go to Step 5.
5	Kernel shall verify that Offline Data Authentication (ODA) was performed (TVR B1b8 = ‘0’). • If ODA was not performed, the Kernel shall send a ‘***Declined***’ Outcome (End the Terminal Action Analysis) • Else, go to Step 6.
6	If the transaction is not sent online and the CDA check is not equal to ‘1` [(TVR B1b3 = ‘0’ (CDA did not fail)], the Kernel shall approve the transaction offline with ‘***Approved***’ Outcome, (End the Terminal Action Analysis). Else the Kernel shall decline the transaction with ‘***Declined***’ Outcome.
7	Kernel shall check if the application usage control checks were skipped (‘Usage Control Checks Skipped’ is set to ‘1’). • If ‘Usage Control Checks Skipped’ is set to ‘1’, then the Kernel shall decline the transaction • Else, the Kernel shall approve the transaction offline

3.10 Online Processing

Online processing is performed if It is supported by the reader and warranted based on the results of the prior steps. The Online Processing step consists of sending the transaction data to the Issuer, who will then decide whether to approve or decline the transaction and then send that decision back to the reader. This enables the Issuer to complete further analyses of the transaction relative to the Cardholder’s account and any other criteria specific to the Issuer. If the Reader is configured with TTQ B3b8 = ‘1’ (Issuer Update Processing supported) and the card is configured with CPR B2b5 = ‘1’ (Issuer Update Processing supported), or if the Data

Storage Enabled flag is set to true, the Kernel shall process the Outcome as '***Online Request (for Two Presentments)'***.

3.11 Completion

Completion is the last step of the transaction unless card re-presentment is required. Upon completion, the Kernel communicates an Outcome to Entry Point, and provides parameter values based on the associated Outcome and transaction mode as specified in [EMV Book A]. The final completion decision may consist of one of the following:

- **Offline declined**: The transaction has been declined. The Kernel shall notify the user and log information regarding the transaction with a '***Declined***' Outcome, or the Kernel may ask the user to present a new card or to switch to another interface by sending a '***Try Another Interface***' Outcome.
- **Offline approved**: The transaction has been approved without sending the transaction online for authorization, and the Kernel shall return the Outcome as '***Approved***' to notify the user. The Terminal shall log transaction information for later authorization (if configured to support Deferred Authorization) and / or clearing (that may occur at a later time).
- **Online approved**: The transaction has been processed online, and the Issuer approved the transaction. Information concerning the transaction may be logged; however, this is not mandatory as the Issuer has already obtained all of the information required for clearing. The Terminal notifies the user that the transaction has been approved online.
- **Online declined**: The transaction has been processed online, and the Issuer declined the transaction. Information concerning the transaction may be logged; however, this is not mandatory because no clearing is required. The Terminal notifies the user that the transaction has been online declined.
- **Switch to another interface**: The transaction cannot be processed using the contactless interface but may be processed via a magnetic stripe or contact interface. The Kernel notifies the user to use another interface by sending a '***Try Another Interface***' Outcome.
- **Select Next**: The Kernel has determined that the selected Combination is unsuitable, and the next Combination (if any) should be tried.
- **Try Again**: The Kernel requires that the device be presented again; this may be a result of an error, such as tearing, that could resolve if the transaction is attempted again.
- **End Application**: The Kernel experienced an application error, such as missing data that will not resolve if the transaction is attempted again with the same selected Contactless Card application.

Refer to Annex B for all the Transaction Outcomes by the Kernel with their Outcome parameters.

3.11.1 Post Completion

Card re-presentment may be requested after completion (irrespective of whether a transaction is approved or declined), if:

- The Issuer provides script processing commands, and/or
- The operator determines that a data storage update is required.

If Data Storage is enabled for the transaction, the POS System may invoke operator-specific processing* to review the transaction outcome and decide if any Data Container updates are required. If updates are required, they shall be formatted in a Write Data Storage Template.
* The use and function of any operator-specific business logic is out of scope of this specification.

3.12 Issuer Update Processing

Issuer Update Processing is an optional step that shall be processed only if the Kernel receives Issuer script processing commands. Issuer scripts may be processed irrespective of whether an online transaction is approved or declined. This requires both card and reader to support the second presentment of a card after an Issuer responds to an online request. If the Issuer responds with a script command in the online authorization response message, the Kernel will request the cardholder to present the card again. Once the card is presented, the reader sends the SELECT command using the same combination as used for the first presentment. If a different card is presented, then either it will not be successfully selected by the reader, or the Issuer script update will fail.
Issuer script commands are received in the authorization responses in Tags ‘71’ or ‘72’. For Kernel 6, only one template tag ‘71’ or ‘72’ is allowed in one transaction. Each individual script command within tag ‘71’ or ‘72’ is encapsulated within tag ‘86’.
The aim of Issuer Updates Processing is to transmit Issuer script commands to the card and check the answer returned by the card as follows:

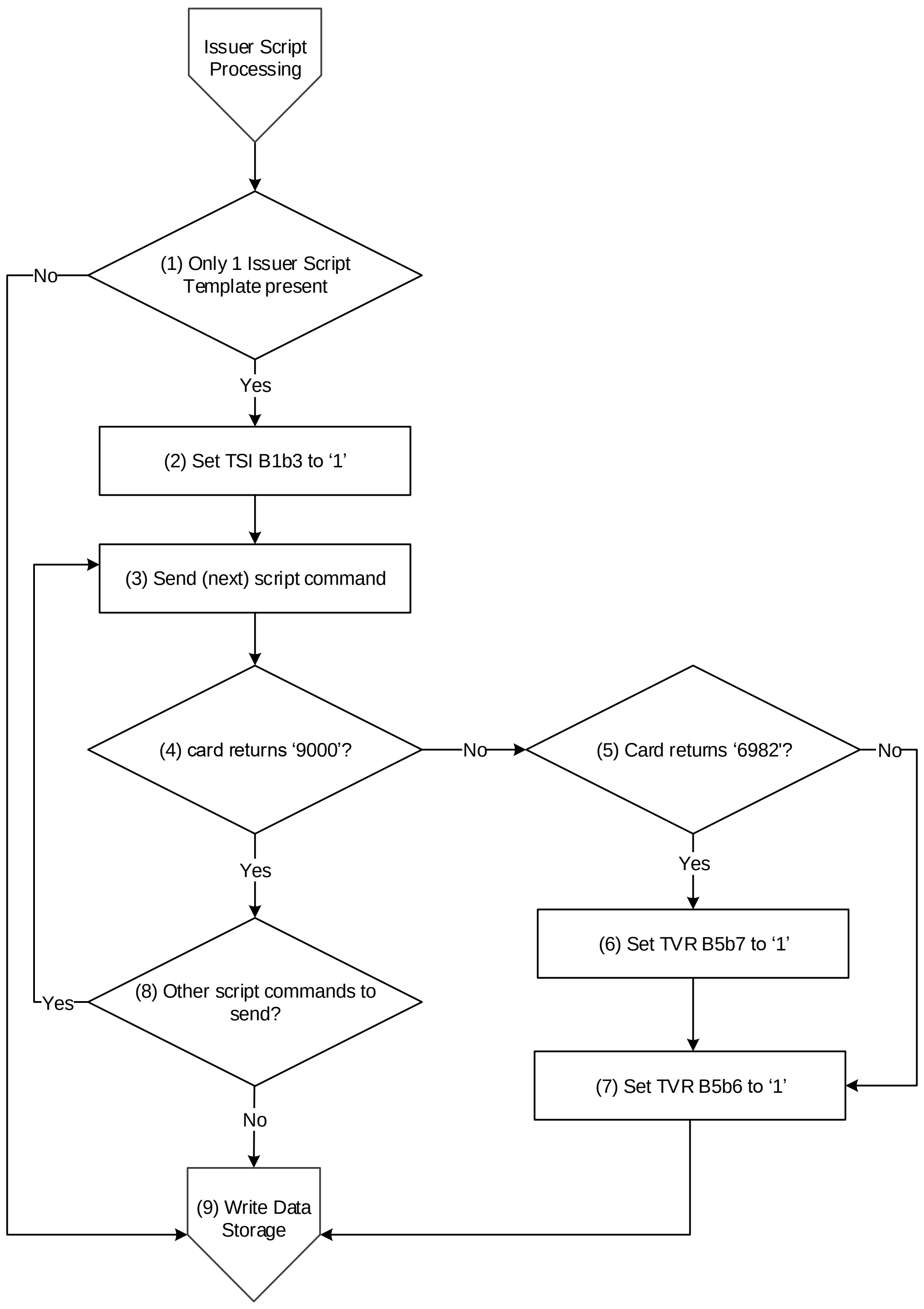
- SW = ’9000’: Kernel has successfully processed the command. The Kernel shall send the next script command (if a new script command has to be sent to the card).
- SW = ’6982’: Kernel shall update TVR B5b7 to ‘1’ (Issuer authentication failed) and TVR B5b6 to ‘1’ (Script processing failed before final GENERATE AC).
- Other SWs: Kernel shall update TVR B5b6 to ‘1’ (Script processing failed before final GENERATE AC).

Each time that the Terminal receives Issuer script commands to be transmitted to the card, the Terminal has to update:

- TSI B1b5 to ‘1’ (Issuer authentication was performed), and
- TSI B1b3 to ‘1’ (Script processing was performed).

Processing is ended when the card generates an error message or no additional scripts must be sent to the card. At that point, after optional Write Data Storage processing, the Terminal shall send the TVR and TSI to the Issuer to communicate the status of the script commands sent to the card.
The following diagram illustrates the Issuer update process.

Figure 3-20: Issuer Update Process



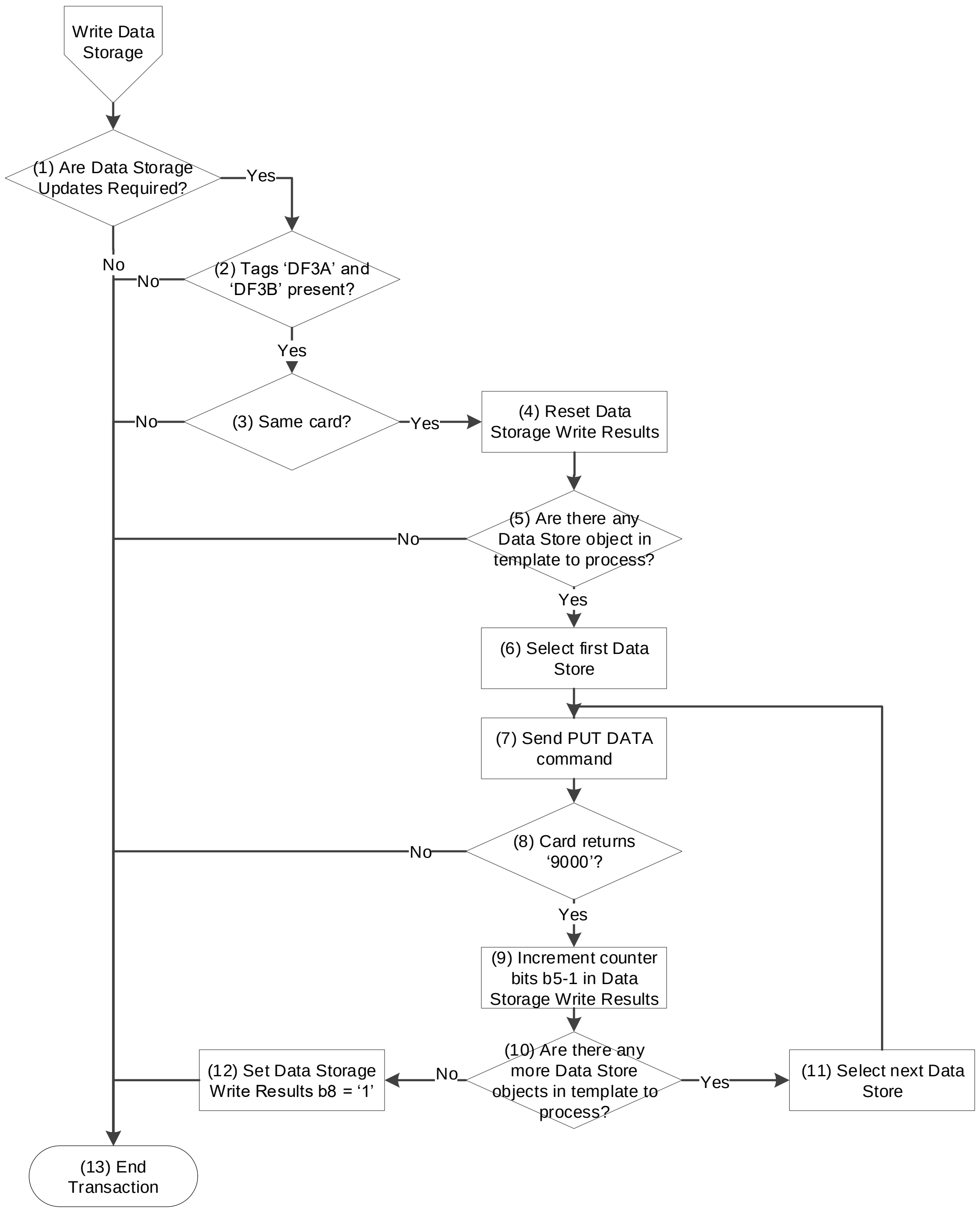
#	Description		
1	Kernel shall check if only one Issuer Script template is present.		
		If…	Then…
		Yes	Go to Step 2.
		No	Go to Step 9.
2	Terminal shall update TSI (B1b3 = 1, Script processing was performed) to indicate that Issuer scripts commands are being sent to the card.		
3	Kernel shall send the script command (next available) received from Issuer.		
4	Kernel shall verify the status word returned by the card.		
		If the card…	Then…
		Returns ‘9000’	Go to Step 8.
		Does not return ‘9000’	Go to Step 5.
5	Kernel shall verify the status word returned by the card.		
		If the card…	Then…
		Returns ‘6982’	Go to Step 6.
		Does not return ‘6982’	Go to Step 7.
6	There is an error with the MAC verification. Kernel shall update TVR B5b7 to ‘1’ (Issuer authentication failed).		
7	Kernel shall update TVR B5b6 to ‘1’ (Script processing failed before final GENERATE AC). The Terminal shall send to TVR and TSI to the Issuer. Then go to Step 9.		
8	The Kernel shall check if another script command needs to be sent to the card.		
		If …	Then…
		Yes	Go to Step 3.
		No	Terminal shall send the TVR and TSI to the Issuer, then go to Step 9.
9	Kernel shall go to Figure 3-21 to perform optional Write Data Storage.		

3.13 Write Data Storage

Write Data Storage is an optional step that shall be processed *if the Kernel receives a Write Data Storage Template from the Terminal*.
As with Issuer Update Processing, data storage updates may be processed irrespective of whether a transaction is approved or declined.
If the Terminal provides a Write Data Storage Template during the Kernel restart, the Kernel will reset Data Storage Write Results to zero and perform Write Data Storage as described in this section.
After completing this step, regardless of whether Write Data Storage is performed or not, the Kernel shall provide an ‘***End Application***’ Outcome. The POS system will indicate to the

cardholder the transaction Outcome based on the Issuer authorization response, regardless of the results of Issuer Update Processing or Write Data Storage*.
* The operator may choose to reverse the transaction if an error causes a Data Container update to fail.

Figure 3-21: Write Data Storage Process


Write Data Storage
(1) Are Data Storage Updates Required?
Yes
No
(2) Tags 'DF3A' and 'DF3B' present?
No
Yes
(3) Same card?
No
Yes
(4) Reset Data Storage Write Results
(5) Are there any Data Store object in template to process?
No
Yes
(6) Select first Data Store
(7) Send PUT DATA command
(8) Card returns '9000'?
No
Yes
(9) Increment counter bits b5-1 in Data Storage Write Results
(10) Are there any more Data Store objects in template to process?
No
Yes
(11) Select next Data Store
(12) Set Data Storage Write Results b8 = '1'
(13) End Transaction

#	Description	
1	Kernel shall check if the Data Storage Write Results is greater than zero and a Write Data Storage Template (tag ‘BF11’) is present.	
	If…	Then…
	Yes	Go to Step 2.
	No	Go to Step 13.
2	Kernel shall verify that Card Feature Version Number (tag ‘DF3A’) and Card Feature Descriptor (tag ‘DF3B’) are present in the card’s selection response.	
	If…	Then…
	Yes	Go to Step 3.
	No	Go to Step 13.
3	Kernel shall verify that Card Feature Version Number and Card Feature Descriptor match the values present in the Write Data Storage Template (to confirm that the same card is being re-presented).	
	If…	Then…
	Yes	Go to Step 4.
	No	Go to Step 13.
4	Kernel shall reset Data Storage Write Results to zero.	
5	Kernel shall check if there any Data Store data objects contained in the Write Data Storage Template (contained in nested Data Storage Update Template, tag ‘BF10’).	
	If…	Then…
	Yes	Go to Step 6.
	No	Go to Step 13.
6	Kernel shall select the first Data Store data object to be processed.	
7	Kernel shall send the Data Store data object to the card using the PUT DATA command, with P1P2 = ‘DF3F’ and CLA=’80’ (see section 4.8).	
8	Kernel shall verify that status word ‘9000’ is returned.	
	If…	Then…
	Yes	Go to Step 9.
	No	Go to Step 13.
9	Kernel shall increment the successful write counter bits 5-1 in the Data Storage Write Results.	
10	Kernel shall check if there any Data Store data objects contained in the Write Data Storage Template.	
	If…	Then…
	Yes	Go to Step 11.
	No	Go to Step 12.
11	The Terminal shall select the next Data Store object to process. Go to Step 7.	

12	The Terminal shall set the successful write bit of Data Storage Write Results data element (bit 8).
13	Kernel shall end second presentment processing with the '***End Application***' Outcome. The Kernel shall include the TVR in the Data Record Outcome Parameters (see Annex B.11). If present, the Kernel shall include the Data Storage Write Results* as a Discretionary Data Outcome Parameter (see Annex B.12). * **Note**: The operator can check the Data Storage Write Counter to verify that data storage updates were successfully applied.

[This page is intentionally left blank.]

4 Application Protocol Data Unit (APDU) Command Description

4.1 Summary

This section describes the APDU commands that the Terminal must be able to send to the card. Each command sent by the Terminal must be correctly formatted as specified in [ISO/IEC 7816-4]. Status bytes descriptions are included in [EMV Book 3].
When the card returns the answer to the command, the Terminal must always first check the SW of the answer and abort the transaction if the SW does not indicate normal processing (i.e., SW = ’9000’ or ’61XX’).

Table 4-1: List of APDU Commands Used by This Kernel

Command	**CLA**	**INS**	**P1**	**P2**	**Lc**	**Data**	**Le**
SELECT	‘00’	‘A4’	‘04’	‘00’ First or only occurrence	’05’-‘10’	AID	‘00’
				’02’ Next occurrence			
GET PROCESSING OPTIONS	‘80’	‘A8’	‘00’	‘00’	Var.	Data elements specified in PDOL Tag ‘83’ || Length || Concatenation of the BER-TLV value of the data elements specified in Processing Options Data Object List (PDOL) No PDOL: ‘8300’	‘00’

Command	**CLA**	**INS**	**P1**	**P2**	**Lc**	**Data**	**Le**
DATA GET PROCESSING OPTIONS	'80'	'D0'	Data Storage Commit Control Parameter	'00'	Var.	Data elements specified in PDOL Tag '83' || Length || Concatenation of the BER-TLV value of the data elements specified in Processing Options Data Object List (PDOL) || Data Storage Update Template tag 'BF10' || length || sequence of Data Container updates.	'00'
RESUME GET PROCESSING OPTIONS	'80'	'D1'	Control Parameter	'00'	Var.	Same data elements as provided in prior GPO or DATA GPO command.	'00'
READ RECORD	'00'	'B2'	Record Number	Reference Control Parameter	Not Present	Not Present	'00'
GET DATA	'80'	'CA'	Tag		Not Present	Not Present	'00'
							Length of TLV field
UPDATE RECORD	'84'	'DC'	Record Number	Reference Control Parameter	Var.	Record Data Object + 8 Bytes for MAC	Not Present
PUT DATA	'84'	'DA'	Tag		Var.	Tagged Data Object + 8 bytes for MAC	Not Present
			Function			Control Data + + Data Object + 8 bytes for MAC	
	'80'	'DA'	Tag		Var.	Tagged Data Object	Not Present
APPLICATION BLOCK	'84'	'1E'	'00'	'00'	'08'	MAC	Not Present
					Var.	Target AID Data + MAC	
APPLICATION UNBLOCK	'84'	'18'	'00'	'00'	'08'	MAC	Not Present
					Var.	Target AID Data + MAC	

4.2 SELECT

The SELECT command is issued to activate and select an application in the card. The SELECT command conforms to the requirements specified in [EMV Book 1].

4.3 GET PROCESSING OPTIONS

The GET PROCESSING OPTIONS (GPO) command is used to inform the Contactless Card that the processing of a new transaction is beginning. The command is sent by the kernel to the card with the information requested in the PDOL.
The GET PROCESSING OPTIONS command conforms to the requirements specified in [EMV Book 3] and the Kernel-specific requirements described in the following sections.

4.3.1 Command Format

The GPO command is coded as shown in Table 4-2.

Table 4-2: GET PROCESSING OPTIONS Command Format

Code	Value
CLA	'80'
INS	'A8'
P1	'00'
P2	'00'
Lc	Var.
Data	Tag '83' || Length || Concatenation of the BER-TLV value of the data elements specified in Processing Options Data Object List (PDOL) [|| Tag 'DF3C' || Length || custom merchant logging data].
Le	'00'

PDOL data items are coded as specified in [EMV Book 3].
If both card and Kernel support Extended Logging, then an optional Extended Logging Data element (tag 'DF3C') may be appended after the PDOL data items.

4.3.2 Data Field Returned in the Response Message

The data field returned in the GPO response message shall be coded according to EMV format 2. This format requires a constructed BER-TLV data object with template '77' as shown in the following table.

Table 4-3: GET PROCESSING OPTIONS Response Data Objects

Tag	Length	Data Object 1	Data Object 2	Data Object ....
'77'		Application Interchange Profile (AIP) (Primitive or constructed BER-TLV data object number 1)	Application File Locator (AFL) (Primitive or constructed BER-TLV data object number 2)	… (Primitive or constructed BER TLV data object number …)

The value field shall consist of several BER-TLV coded objects that shall always include the mandatory data specified in Table 4-4 (for EMV Transactions).
If any of these mandatory data elements are missing, the Terminal shall terminate the transaction.

Table 4-4: GET PROCESSING OPTIONS Response Data Field for Contactless EMV Transactions

Data Element	Tag	Length	Online Processing (ARQC[1], no CDA)	Decline Processing (AAC, no CDA)	Offline Capable Processing (TC or ARQC[1], with CDA)
Application Interchange Profile (AIP)	‘82’	2 bytes	Mandatory	Mandatory	Mandatory
Application File Locator (AFL)	‘94’	Var.	Conditional[2]	Conditional[2]	Mandatory
Application Cryptogram	‘9F26’	8 bytes	Mandatory	Mandatory	Not Present
Signed Dynamic Application Data (SDAD)	‘9F4B’	N_{IC} bytes	Not present	Not Present	Conditional[3]
Application Transaction Counter (ATC)	‘9F36’	2 bytes	Mandatory	Mandatory	Mandatory
Issuer Application Data (IAD)	‘9F10’	Var. up to 32 bytes	Mandatory	Mandatory	Mandatory
Cryptogram Information Data (CID)	‘9F27’	1 byte	Mandatory	Mandatory	Mandatory
Track 2 Equivalent Data	‘57’	Var. up to 19 bytes	Conditional (Not Present if AFL is present)	Conditional (Not Present if AFL is present)	Not present
Application Usage Control (AUC)	‘9F07’	2 bytes	Conditional (Not Present if AFL is present)	Conditional (Not Present if AFL is present)	Not present
Issuer Country Code	‘5F28’	2 bytes	Conditional (Not Present if AFL present)	Conditional (Not Present if AFL present)	Not Present

Data Element	Tag	Length	Online Processing (ARQC[1], no CDA)	Decline Processing (AAC, no CDA)	Offline Capable Processing (TC or ARQC[1], with CDA)
Application Primary Account Number (PAN) Sequence Number	'5F34'	1 byte	Conditional (Not Present if AFL is present)	Conditional (Not Present if AFL is present)	Not present
Card Processing Requirements (CPR)	'9F71'	2 bytes	Mandatory	Mandatory	Mandatory
Offline Balance	'D1'	6 bytes	Optional	Not present	Optional
Application Effective Date	'5F25'	3 bytes	Conditional (Not Present if AFL is present)	Conditional (Not Present if AFL is present)	Not present
Application Version Number	'9F08'	2 bytes	Conditional (Not Present if AFL is present)	Conditional (Not Present if AFL is present)	Not present
Payment Account Reference (PAR)	'9F24'	Var.	Optional (Not Present if AFL is present)	Optional (Not Present if AFL is present)	Not present

[1] If the card is configured to support Offline Data Authentication (ODA) for online transactions and the Reader supports ODA for Online Authorizations (TTQ B1b1 is set to '1'), then the card returns ARQCs with CDA, otherwise it returns ARQCs without CDA.

[2] Presence of the Application File Locator (AFL) for online or offline declined transactions is dependent on card configuration.

[3] Presence of the Signed Dynamic Application Data (SDAD) is dependent on card configuration. If the SDAD is not included in the response message, then it will be included instead in a record reference by the AFL (and will be read during Read Application Data processing).

4.4 DATA GET PROCESSING OPTIONS

The DATA GET PROCESSING OPTIONS (DATA GPO) command is used to inform the Contactless Card that the processing of a new data storage transaction is beginning.
The DATA GPO command acts as an extended alternative to the standard GPO command, as described in section 4.3, and is used instead of GPO when data storage updates are required during Initiate Application Processing.
The DATA GPO command takes a composite input comprising the information requested in the PDOL, plus a Data Storage Update Template containing a sequence of Data Container updates.
The command performs GPO EMV processing plus, depending on the transaction outcome, Data Storage updates.

4.4.1 Command Format

The DATA GPO command is coded as shown in Table 4-5.

Table 4-5: DATA GET PROCESSING OPTIONS Command Format

Code	Value
CLA	‘80’
INS	‘D0’
P1	Data Storage write commit control (see Table 4-6)
P2	’00’
Lc	Var.
Data	Tag ‘83’ || Length || Concatenation of the BER-TLV value of the data elements specified in Processing Options Data Object List (PDOL) || Data Storage Update Template tag ‘BF10’ || length || sequence of Data Store data objects (see Figure 4-1 and Figure 4-2)
Le	‘00’

Data Storage updates are conditional on the type of Application Cryptogram returned, as configured in Table 4-6. If the condition is met, then the card shall commit the Data Storage update; this shall occur either at the end of successful DATA GPO command processing (if no AFL returned), or on successful completion of Read Record Processing (after the final record listed in the AFL is read).

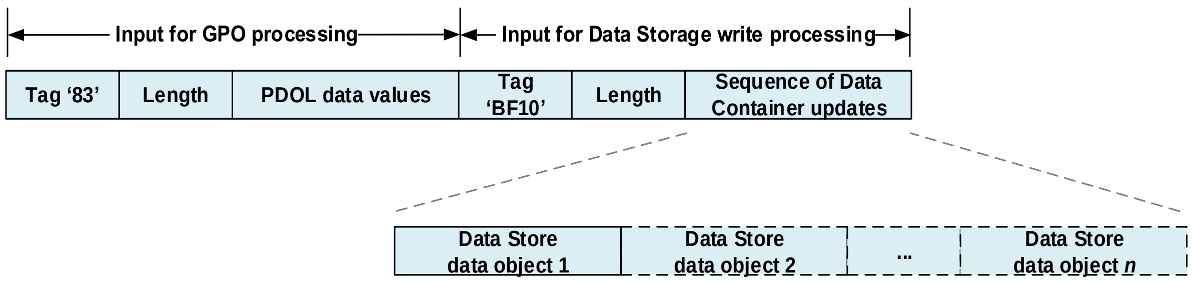
Table 4-6: DATA GET PROCESSING OPTIONS Data Storage Write Commit Control

Bit	Value
b8-5	00000 (any other value is RFU)
b4	0 = do not commit if completed Loyalty Program transaction 1 = commit update if completed Loyalty Program transaction

Bit	Value
b3	0 = do not commit data storage update if TC 1 = commit data storage update if TC
b2	0 = do not commit data storage update if ARQC 1 = commit data storage update if ARQC
b1	0 = do not commit data storage update if AAC 1 = commit data storage update if AAC

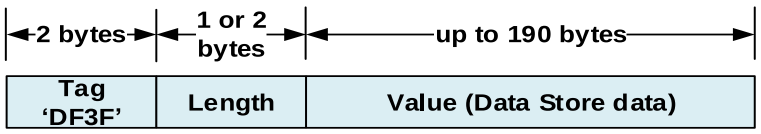
The DATA GPO input data field must be formatted as shown in Figure 4-1. Any number of Data Container updates can be included in the input, but the total length of the combined PDOL data (tag '83') and Data Storage Update Template (template 'BF10') must not exceed 255 bytes.

Figure 4-1: DATA GET PROCESSING OPTIONS – Command Data Format



The Data Storage Update Template may contain one or more Data Store data objects, as shown in Figure 4-2.

Figure 4-2: DATA GET PROCESSING OPTIONS – Data Store Data Object Encoding



An operator may update Transient and Operator Containers, sending a Data Store data object for each Data Container to be updated. Each Data Store data object is encoded as described in Annex D.17.

Note: Each Data Store data object included in the Data Storage Update Template is equivalent to using a separate PUT DATA command to write to the Data Store.

4.4.2 Data Field Returned in the Response Message

The data field returned in the DATA GPO response message shall be coded according to EMV format 2, as specified in [EMV Book 3]. This format requires a constructed BER-TLV data object with template '77' as shown in Table 4-3.

The value field shall consist of several BER-TLV coded objects that shall always include the mandatory data specified in Table 4-4. If any of the data elements marked as mandatory are missing, the Terminal shall terminate the transaction.

4.5 RESUME GET PROCESSING OPTIONS

The RESUME GET PROCESSING OPTIONS (RESUME GPO) command acts as an extended alternative to the GET PROCESSING OPTIONS (GPO) and DATA GET PROCESSING OPTIONS (DATA GPO) commands, and may be used during optional Tearing Recovery processing to attempt transaction recovery without starting a new transaction.
For RESUME GPO processing, the card will compare the given command data with that of the last EMV transaction it successfully processed, and:

- If there is a match, the card will return the same response message as the last GPO, DATA GPO or RESUME GPO command, without starting a new transaction, otherwise
- If there is no match, the card will start a new transaction and process the command as a new GPO or DATA GPO command.

4.5.1 Command Format

The RESUME GPO command is coded as shown in Table 4-7.

Table 4-7: RESUME GET PROCESSING OPTIONS Command Format

Code	Value
CLA	‘80’
INS	‘D1’
P1	= ‘00’ if resuming after a prior GPO command = ‘8x’ if resuming after a prior DATA GPO command See Table 4-8
P2	’00’
Lc	Var.
Data	If resuming after a prior GPO command, then: Tag ‘83’ || Length || Concatenation of the BER-TLV value of the data elements specified in Processing Options Data Object List (PDOL) [|| Tag ‘DF3C’ || Length || custom merchant logging data]. If resuming after a prior DATA GPO command, then: Tag ‘83’ || Length || Concatenation of the BER-TLV value of the data elements specified in Processing Options Data Object List (PDOL) || Data Storage Update Template tag ‘BF10’ || length || sequence of Data Store data objects (see Figure 4-1 and Figure 4-2)
Le	‘00’

Table 4-8: RESUME GET PROCESSING OPTIONS Parameter P1 Settings

Bit	Value
b8	0 = Resume a transaction initiated with a GPO command 1 = Resume a transaction initiated with a DATA GPO command
b7-5	0000 (any other value is RFU)
b4	0 = do not commit if completed Loyalty Program transaction 1 = commit update if completed Loyalty Program transaction
b3	0 = do not commit update if TC 1 = commit update if TC
b2	0 = do not commit update if ARQC 1 = commit update if ARQC
b1	0 = do not commit update if AAC 1 = commit update if AAC

4.5.2 Data Field Returned in the Response Message

The data field returned in the RESUME GPO response message shall be coded according to EMV format 2 as specified in [EMV Book 3]. This format requires a constructed BER-TLV data object with template '77' as shown in Table 4-3.

The value field shall consist of several BER-TLV coded objects that shall always include the mandatory data specified in Table 4-4. If any of the data elements marked as mandatory are missing, the Terminal shall terminate the transaction.

4.6 READ RECORD

The READ RECORD command shall be issued by the Kernel to retrieve data stored in one record. The data present in the record depends on how the application has been personalized. The command is used during optional Read Application Data processing, and is used if an AFL is returned by the card during Initiate Application Processing. The Kernel shall then send the command according to information in the AFL. The Terminal shall store the data read in temporary memory. Some Terminals (such as ATMs for example) are also able to read the log record.

The command may also be used during optional Read Data Storage processing to retrieve the contents of Data Containers. If Data Storage is required, the Kernel shall send the command according to information in the Card Feature Descriptor and the Data Storage Directory to retrieve each Data Container listed in the Data Container Read List.

Refer to [EMV Book 3] for additional information, including the Command format and Response message format.

For SFIs in the range 1-10, the record shall be a BER-TLV constructed data object coded as specified in [EMV Book 3].

For Data Container records, the record shall be a SIMPLE-TLV constructed data object, as specified in [ISO 7816-4], and coded as shown in Table 4-9.

Table 4-9: READ RECORD Response Message Data Field for Data Containers

Value	Length	Presence
Tag – the record number	1	M
Length – the length of the Data Container’s content	1	M
Value – the Data Container’s content	Var (0-160)	O

4.7 GET DATA

This command shall be issued by the Terminal to retrieve a primitive data object that is not encapsulated in a record within the current application. The Terminal shall store the data value in temporary memory.
Refer to [EMV Book 3] for additional details, including the Command format and Response message format.

4.8 PUT DATA

This command shall be issued by the Terminal to write new values into specific primitive data objects, during optional second presentment. The command supports two modes of operation, controlled by the command's class setting:

- When CLA = '84', the PUT DATA command is an Issuer script command that can only be sent during Issuer update processing. The full command is sent by Issuer to the Terminal. The command shall include a Secure Messaging MAC.
  - Where P1-P2 ='DF 4A', the command is a specific type of PUT DATA called a PUT DATA – PSE Data command.
  - Where P1-P2 ='DF 50' or 'DF 51', the command is a specific type of PUT DATA called a PUT DATA Inter-Application command.
  - Where P1-P2 ='DF 52', 'DF 53', 'DF 54' or 'DF 55', the command is a specific type of PUT DATA called a PUT DATA – Profile Data command.
- When CLA = '80', the PUT DATA command is used to write new Data Container values into the virtual Data Store data object and does not use Secure Messaging. The command is used by the Terminal to update Transient Containers or Operator Containers. No other data objects may be updated using this mode.

Note: Data Container updates may be initiated by operators or the card issuer.

4.8.1 Command Format

The PUT DATA command is coded as specified in Table 4-10.

Table 4-10: PUT DATA Command Format

Code	Value
CLA	'80' or '84'
INS	'DA'
P1	MSB byte of the primitive data object to be updated (set to '00' if tag is one byte long)
P2	LSB byte of the primitive data object to be updated
Lc	Var. (length of data field, including the length of the MAC when CLA = '84')

Code	**Value**
Data	New data to be set to the data object (and MAC when CLA = ‘84’) or Control Information, new data and MAC for Inter-Application PUT DATA and PUT Profile DATA variant.
Le	Not present

4.8.2 Data Field Returned in the Response Message

No data are returned by the card. The Terminal shall check that card returns ’9000’ to this command.

4.9 UPDATE RECORD

The UPDATE RECORD command is an Issuer script command that writes new values into the Elementary File (EF) Record identified by an SFI. This command shall be issued by the Terminal on Issuer request. The aim is to update one record content.
For a Contactless Card transaction, the command can only be sent during Issuer update processing (meaning during second presentment). The full command is sent by Issuer to the Terminal.

4.9.1 Command Format

The UPDATE RECORD command is coded as specified in the table below:

Table 4-11: UPDATE RECORD Command Format

Code	Value
CLA	'84'
INS	'DC'
P1	Record Number
P2	Reference Control Parameter
Lc	Var. (length of data field including the MAC)
Data	New data value and MAC (MAC is 8-bytes long)
Le	Not present

The reference control parameter shall be coded as follows

Table 4-12: Reference Control Parameter

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

4.9.2 Data Field Returned in the Response Message

No data are returned by the card. The Terminal shall check that the card returns '9000' to this command.

4.10 APPLICATION BLOCK

This is a script command provided by the Issuer to the Terminal to invalidate the application. The Terminal shall send the command as provided by the Issuer. The command shall be coded as specified in [EMV Book 2] using Secure Messaging for Integrity and Authentication Format 2.
Refer to [EMV Book 3] for additional details, including the Command format and Response message format.
A variant of the command supports inter-application blocking to invalidate other applications on the card via issuer scripting to the currently selected application.

4.11 APPLICATION UNBLOCK

This is a script command provided by the Issuer to the Terminal to unblock the application. The Terminal sends the command as provided by the Issuer. The command shall be coded as specified in [EMV Book 3] using Secure Messaging for Integrity and Authentication Format 2.
Refer to [EMV Book 3] for additional details, including the Command format and Response message format.
A variant of the command supports inter-application unblocking to unblock other applications on the card via issuer scripting to the currently selected application.

Annex A Glossary

This annex provides a glossary of terms and abbreviations used in this specification.

Term	Definition
AAC	Application Authentication Cryptogram. An Application Cryptogram generated by the card when declining a transaction.
AC	Application Cryptogram. For contactless transactions: A cryptogram generated by the card in response to a GET PROCESSING OPTIONS command.
ADF	Application Definition File. Identifies the application (AID) as described in [ISO/IEC 7816-5].
AEF	Application Elementary File. See [EMV Book 3].
AFL	Application File Locator. Indicates the location (SFI, range of records) of the AEFs related to a given application.
AID	Application Identifier. Identifies the application as described in [ISO/IEC 7816-5].
AIP	Application Interchange Profile. Indicates the capabilities of the card to support specific functions in the application.
APDU	Application Protocol Data Unit. The Data Unit used to exchange information between the Terminal via a Command APDU (C-APDU) and card via a Response APDU (R-APDU) as defined in [ISO/IEC 7816-4].
API	Application Priority Indicator. Indicates the priority of a given application or group of applications in a directory.
Application Effective Date	Date from which the application may be used.
Application Expiration Date	Date after which the application expires.
Application Label	Mnemonic associated with the AID according to [ISO/IEC 7816-5].
Application Version Number	Version number assigned by the payment system for the application.
ATC	Application Transaction Counter. Counter maintained by the application in the ICC (incrementing the ATC is managed by the ICC).
AUC	Application Usage Control. Indicates the Issuer's specified restrictions on the geographic usage and services allowed for the application.

Term	Definition
BER-TLV	Basic Encoding Rules – Tag Length Value. A set of encoding rules for a data object. As defined in [ISO/IEC 8825-1], a BER-TLV data element consists of the following three consecutive components: • Tag field (T) indicates a class, a type, and a number. • Length field (L) indicates the length of the field. • Value field (V) indicates the value of the data object. (Note that if L = '00', the value field is not present.)
CA PKI	Certificate Authority Public Key Index. Identifies the certification authority's public key in conjunction with the RID.
CDA	Combined Dynamic Data Authentication / Application Cryptogram Generation. A form of offline dynamic data authentication.
CID	Cryptogram Information Data. Indicates the type of cryptogram and the actions to be performed by the Terminal.
Combination	Apply the following: For: / The combination of: • a card — • an ADF Name • a Kernel Identifier • a reader — • an AID • a Kernel ID • the Candidate List for final selection — • an ADF Name • a Kernel ID • the Application Priority Indicator (if present) • the Extended Selection (if present)
CPR	Card Processing Requirement. Data element that indicates the card requirements for processing the transaction to the reader.
CRM-CACs	Card Risk Management-Card Action Codes. Designated by Issuers and used during the Card Action Analysis to determine the level of risk associated with a transaction.
CVM-CACs	Card Verification Method-Card Action Codes. Designated by Issuers and used during the Card Action Analysis to determine the type of Card Verification to be applied to the transaction.
Data Container	A logical unit of data storage. There are three types of Data Container: a Transient Container, an Operator Container, and an Issuer Container.
Data Element Processing	Addresses Requirements regarding the Presence of Data, Rules for Padding, and Order of Data Elements as specified in [EMV Book B].

Term	Definition
Data Storage	The optional capability for a terminal to read and write non-payment related data on a card during a transaction. The data, stored in Data Containers, will persist between transactions.
Data Storage Directory	A composite data object containing an unpredictable number (DSD_UN) generated by the card and a listing of Data Containers allocated on a card.
Data Storage Registration Authority	The body responsible for assigning Container IDs to operators.
DDF	Directory Definition File (DDF). Issuer discretionary part of the directory according to [ISO/IED 7816-5].
Deferred Authorization	Deferred Authorization provides a solution for terminals used in environments where real time online authorization is not possible. For Deferred Authorization, when an EMV transaction requires online authorization (and after successful Offline Data Authentication) the kernel will provide an Approved Outcome. The terminal or back-office may then request authorization at a later time.
DF Name	Dedicated File Name. Identifies the name of the DF as described in [ISO/IEC 7816-4].
Entry Point	Within these specifications, Entry Point is software in the POS System that is responsible for the following: • Performing pre-processing, • Discovery and selection of a contactless application that is supported by both the card and the reader, • Activation of the appropriate Kernel, and • Handling of Outcomes returned by the Kernel, including passing selected Outcomes to the reader. Under exception conditions, Entry Point may return an Outcome to the reader as a result of its own processing.
Exception File	A file listing cards and associated applications.
Extended Logging	The optional capability for a terminal to store custom data in a card's transaction log.
FCI Issuer Discretionary Data	File Control Information (FCI) Issuer Discretionary Data. Issuer discretionary part of the FCI.
FCI	File Control Information
IAD	Issuer Application Data. Contains proprietary application data for transmission to the Issuer in an online Transaction.
ICC	Integrated Circuit Card. A card with an embedded chip that communicates with a point of interaction (terminal).

Term	Definition
IDD	Issuer Discretionary Data. Issuer-specified data relating to the application that is part of the Issuer Application Data.
ISO/IEC	International Organization for Standardization/International Electrotechnical Commission
Kernel	The set of functions required to be present on every terminal implementing a specific interpreter. The kernel contains device drivers, interface routines, security and control functions, and the software for translating from the virtual machine language to the language used by the real machine. In other words, the kernel is the implementation of the virtual machine on a specific real machine.
Kernel Activation:	The activation of the selected Kernel via Entry Point to enable the beginning of Kernel processing.
Kernel Identifier	Identifier to distinguish between different kernels that may be indicated by the card.
ODA	Offline Data Authentication.
Operator (in data storage)	A transit operator, merchant, or any other business wanting to use the Data Storage feature.
PAN	Primary Account Number.
PAN Sequence Number	Primary Account Number Sequence Number. Identifies and differentiates cards with the same PAN.
PDOL	Processing Data Objects List. Contains a list of terminal resident data objects (tags and lengths) needed by the ICC in processing the GET PROCESSING OPTIONS command.
Protocol Activation	Activation of the contactless interface.
Radio Frequency	RF
RID	Registered Application Provider Identifier. Part of an AID as defined in [ISO/IEC 7816-4] and obtained as specified in [ISO/IEC 7816-5]. The RID is unique to an application provider and consists of the 5 most significant bytes of the AID.
SDAD	Signed Dynamic Application Data. Digital signature on critical application parameters for DDA or CDA.
SFI	Short File Identifier. Identifies the AEF referenced in commands related to a given ADF or DDF. The SFI is a binary data object having a value in the range 1 to 30 and with the three high order bits set to 0.

Term	Definition		
Starting Points		Start at:	Activation
	Start A	Pre-Processing	Start at Pre-Processing; activated by the reader when Autorun is 'No'. This is typical for a new transaction with a variable amount in an EMV acceptance environment.
	Start B	Protocol Activation	Activated in any of the following cases: • Activated by the reader when Autorun is 'Yes', or • Activated by the reader to handle Issuer responses after an ***Online Request*** or ***End Application*** Outcome with parameter Start = B, or • Handled internally by Entry Point for an error situation, or • Handled internally by Entry Point for a ***Try Again*** Outcome
	Start C	Combination Selection	Handled internally by Entry Point for a ***Select Next*** Outcome.
	Start D	Kernel Activation	Activated by the reader to handle Issuer responses after an Online Request Outcome with parameter Start = D.
TC	Transaction Certificate. An Application Cryptogram generated by the card when approving a transaction.		
Tearing Recovery	An optional terminal capability to recover and resume processing in the event of a torn contactless transaction.		
TTQ	Terminal Transaction Qualifiers. Indicates the requirements for online and CVM processing as a result of Entry Point processing. The scope of this tag is limited to Entry Point. Kernels may use this tag for different purposes.		

Term	Definition
Track 2 Equivalent Data	Contains the data elements of Track 2 according to [ISO/IEC 7813], excluding start sentinel, end sentinel, and Longitudinal Redundancy Check (LRC) as follows: • Primary Account Number. • Field Separator (Hex 'D'), • Expiration Date (YYMM), • Service Code, and • Discretionary Data (defined by individual payment systems). Pad with one Hex 'F; if needed to ensure whole bytes.
Triple DES (3DES)	Triple Data Encryption Standard.
TVR	Terminal Verification Results. Status of the different functions as seen from the terminal
UN	Unpredictable Number. Value to provide variability and uniqueness to the generation of a cryptogram.

Annex B Kernel 6 Transaction Outcome and Parameter Settings

An Outcome is the primary instruction from the Kernel or Entry Point on how processing should be continued. When a Kernel provides an Outcome, control is passed back to Entry Point which handles certain parameters immediately, then either processes the Outcome or forwards it to the reader as a Final Outcome.
Refer to [EMV Book A] for description of all the Outcome parameters, their values and User Interface standard Display messages.

B.1. Approved

The kernel has determined that the transaction is approved, either through offline processing or after reactivation following an online response.

Outcome	**Outcome Parameter**	**Value**
Approved	Start	N/A
	Online Response Data	N/A
	CVM	As described in Section 3.7 Cardholder Verification
	UI Request on Outcome Present	Yes Message Identifier: as applicable ‘03’ (Approved) ‘1A’ (Approved – Please Sign) Status: Card Read Successfully [Value Qualifier: Balance] [Value: Offline Balance (from Tag ‘D1’)] [Currency Code: Transaction Currency Code] **Note**: Balance is displayed only if ‘D1’ is returned by card
	UI Request on Restart Present	No
	Data Record Present	Yes The minimum data requirements for clearing records are specified in Annex B.11.
	Discretionary Data Present	Yes if Data Storage enabled, No otherwise. Discretionary Data are specified in Annex B.12
	Alternate Interface Preference	N/A
	Receipt	Yes or N/A as required

Outcome	Outcome Parameter	Value
	Field Off Request	N/A
	Removal Timeout	Zero

B.2. Online Request

The kernel requests online authorization.

Outcome	Outcome Parameter	Value
Online Request	Start	N/A
	Online Response Data	N/A
	CVM	As described in Section 3.7 Cardholder Verification
	UI Request on Outcome Present	Yes Message Identifier: '1B' (Authorizing, Please Wait) Status: Card Read Successfully [Value: Offline Balance (from Tag 'D1')] [Currency Code: Transaction Currency Code] **Note**: Balance is displayed only if 'D1' is returned by card
	UI Request on Restart Present	No
	Data Record Present	Yes The minimum data requirements for online authorization are specified in Annex B.10.
	Discretionary Data Present	Yes if Data Storage enabled, No otherwise. Discretionary Data are specified in Annex B.12
	Alternate Interface Preference	N/A
	Receipt	Yes or N/A as required
	Field Off Request	N/A
	Removal Timeout	Zero

B.3. Online Request (for Two Presentments)

The kernel has determined that the transaction must be sent online and that the card can be re-presented.

Outcome	Outcome Parameter	Value
Online Request (for Two Presentments)	Start	B
	Online Response Data	EMV data
	CVM	As described in Section 3.7 Cardholder Verification
	UI Request on Outcome Present	Yes Message Identifier: as applicable: '1B' (Authorizing, Please Wait) [Value: Offline Balance (from Tag 'D1')] [Currency Code: Transaction Currency Code] **Note**: Balance is displayed only if 'D1' is returned by card
	UI Request on Restart Present	Message Identifier: '21' (Present Card Again) Status: Ready to Read
	Data Record Present	Yes The minimum data requirements for online authorization are specified in Annex B.11.
	Discretionary Data Present	Yes if Data Storage enabled, No otherwise. Discretionary Data are specified in Annex B.12
	Alternate Interface Preference	N/A
	Receipt	Yes or N/A as required
	Field Off Request	N/A
	Removal Timeout	Zero

B.4. Declined

The kernel has determined that the transaction is declined, either through offline processing or after reactivation following an online response.

Outcome	Outcome Parameter	Value
Declined	Start	N/A
	Online Response Data	N/A
	CVM	N/A
	UI Request on Outcome Present	Yes Message Identifier: '07' (Not Authorized) Status: Card Read Successfully [Value Qualifier: Balance] [Value: Offline Balance (from Tag 'D1')] [Currency Code: Transaction Currency Code] **Note**: Balance is displayed only if 'D1' is returned by card
	UI Request on Restart Present	No
	Data Record Present	Yes The minimum data requirements for declined transactions are specified in Annex B.11.
	Discretionary Data Present	Yes if Data Storage enabled, No otherwise. Discretionary Data are specified in Annex B.12
	Alternate Interface Preference	N/A
	Receipt	Yes or N/A as required
	Field Off Request	N/A
	Removal Timeout	Zero

B.5. Try Another Interface

The kernel (or Entry Point) has determined that the transaction cannot be completed over the contactless interface and another interface such as contact chip or mag-stripe should be attempted.

Outcome	Outcome Parameter	Value
Try Another Interface	Start	N/A
	Online Response Data	N/A
	CVM	N/A
	UI Request on Outcome Present	Yes Message Identifier: ‘18’ (Please insert or swipe card) Status: Ready to Read
	UI Request on Restart Present	No
	Data Record Present	No
	Discretionary Data Present	Yes or No
	Alternate Interface Preference	Contact Chip or Mag-Stripe
	Receipt	N/A
	Field Off Request	N/A
	Removal Timeout	Zero

B.6. End Application

Outcome	Outcome Parameter	Value
End Application (for Termination of First presentment or following an Online Request with 'Two presentment' or unrecoverable error)	Start	N/A
	Online Response Data	N/A
	CVM	N/A
	UI Request on Outcome Present	No
	UI Request on Restart Present	No
	Data Record Present	Yes or No
	Discretionary Data Present	Yes if Data Storage enabled, No otherwise. Discretionary Data are specified in Annex B.12
	Alternate Interface Preference	N/A
	Receipt	N/A
	Field Off Request	N/A
	Removal Timeout	Zero

B.7. End Application (for Processing Error)

Outcome	Outcome Parameter	Value
End Application (for Processing Error: Conditions for use of contactless not satisfied)	Start	N/A
	Online Response Data	N/A
	CVM	N/A
	UI Request on Outcome Present	Yes Message Identifier:'1C' (Insert, swipe or try another card) Status: Processing Error
	UI Request on Restart Present	No
	Data Record Present	No
	Discretionary Data Present	No
	Alternate Interface Preference	N/A
	Receipt	N/A
	Field Off Request	N/A
	Removal Timeout	Zero

B.8. Try Again (Entry of a Confirmation Code)

Outcome	Outcome Parameter	Value
End Application (for Processing Error: Conditions for use of contactless not satisfied)	Start	B
	Online Response Data	N/A
	CVM	N/A
	UI Request on Outcome Present	Yes Message Identifier:‘20’ (See Phone for Instructions) Status: Processing Error Hold Time: 13
	UI Request on Restart Present	Yes Status: Ready to Read
	Data Record Present	No
	Discretionary Data Present	No
	Alternate Interface Preference	N/A
	Receipt	N/A
	Field Off Request	13
	Removal Timeout	Zero

B.9. Try Again (for Tearing Recovery)

Outcome	Outcome Parameter	Value
End Application (for Processing Error: Conditions for use of contactless not satisfied)	Start	B
	Online Response Data	N/A
	CVM	N/A
	UI Request on Outcome Present	Yes Message Identifier:‘21’ (Present Card Again) Status: Processing Error Hold Time: 13
	UI Request on Restart Present	Yes Status: Ready to Read
	Data Record Present	No
	Discretionary Data Present	No
	Alternate Interface Preference	N/A
	Receipt	N/A
	Field Off Request	13
	Removal Timeout	Zero

B.10. Select Next

The Kernel has determined that the selected combination is unsuitable and the next combination (if any) should be tried. In practice, this is handled directly by Entry Point and only the UI Request parameter settings are relevant.

Outcome	Outcome Parameter	Value
Select Next	Start	C
	Online Response Data	N/A
	CVM	N/A
	UI Request on Outcome Present	No
	UI Request on Restart Present	No
	Data Record Present	No
	Discretionary Data Present	No
	Alternate Interface Preference	N/A
	Receipt	N/A
	Field Off Request	N/A
	Removal Timeout	Zero

B.11. Data Record Outcome Parameter

This section describes the minimum data elements to be present in the Data Record Outcome parameter depending on the Transaction Outcome (Approved, Online request or Declined). Depending on the outcome, the Kernel will make appropriate data records available to enable the clearing and/or authorization of the transaction.
In Table 4-13, data elements mentioned as ‘M’ are mandatory elements that need to be present in the data record for the given Transaction Mode and Outcome and data elements mentioned as ‘C’ are conditional elements that need to be present if they are provided by the Card or POS or Kernel.

Table 4-13: Minimum Data Elements Returned in Transaction Data Record Outcome Parameter

Data Element	Tag	Source	Approved & Online request	Declined
Amount, Authorized	‘9F02’	POS	M	M
Amount, Other	‘9F03’	POS	C	C
Application Cryptogram (AC)	‘9F26’	Card	M	-
Application Identifier (AID)	‘9F06’	POS	C	-

Data Element	Tag	Source	Approved & Online request	Declined
Application Interchange Profile (AIP)	‘82’	Card	M	-
Application PAN Sequence Number	‘5F34’	Card	C	-
Application Transaction Counter (ATC)	‘9F36’	Card	M	-
Application Usage Control	‘9F07’	Card	C	-
Cardholder Name	‘5F20’	Card	C	-
Cryptogram Information Data	‘9F27’	Card	C	-
Dedicated File Name	‘84’	Card	C	-
Issuer Application Data (IAD)	‘9F10’	Card	M	-
Terminal Application Version Number	‘9F09’	POS	C	-
Terminal Capabilities	‘9F33’	POS	M	M
Terminal Country Code	‘9F1A’	POS	M	M
Terminal Type	‘9F35’	POS	M	M
Terminal Verification Results (TVR)	‘95’	Kernel 6	M	-
Track 1 Discretionary Data	‘9F1F’	Card	C	-
Track 2 Equivalent Data	‘57’	Card	M	-
Transaction Date	‘9A’	POS	M	M
Transaction Type	‘9C’	POS	M	M
Unpredictable Number	‘9F37’	POS	M	-

B.12. Discretionary Data Outcome Parameter

This section describes the Discretionary Data items that may be present in the Outcome if Data Storage is supported by the Kernel.

Table 4-14: Discretionary Data Outcome Parameter

Data Item	Tag	Source	Description
Card Feature Version Number	‘DF3A’	Card	If present on the card.
Card Feature Descriptor	‘DF3B’	Card	If present on the card.

Data Item	Tag	Source	Description
Data Storage Directory	‘DF3D’	Card	If read from the card.
Data Container Records	-	Card	A sequence of Data Container records, one for each record retrieved from the card during Read Data Storage, if any.
Data Storage Write Results	-	Kernel	If Data Store updates were provided by operator for Write Data Storage processing.

Annex C Terminal Configuration: Data Elements

This section describes the parameters to be set in the Terminal for each available combination (i.e., offline only, online only, and offline / online capable) based on the following definitions:

- **Mandatory:** Always required,
- **Conditional:** Dependent on the configuration, or
- **Optional:** May be present.

C.1. Offline Only Terminals

The following table lists data elements that must be present in Terminals that are offline capable only.

Table 4-15: Offline Only Terminals

Data Element	Tag	Presence
Amount, Authorized	‘9F02’	Mandatory
Amount, Other	‘9F03’	Conditional
Application Identifier	‘9F06’	Mandatory
Application Version Number	‘9F09’	Mandatory
Certificate Authority Public Key Checksum	Proprietary	Mandatory
Certificate Authority Public Key Exponent	Proprietary	Mandatory
Certificate Authority Public Key Index	’9F22’	Mandatory
Certificate Authority Public Key Modulus	Proprietary	Mandatory
Data Container Read List	Proprietary	Optional
Data Storage Supported flag	Proprietary	Optional
Extended Logging Supported flag	Proprietary	Optional
Extended Selection Support flag	Proprietary	Optional
Interface Device (IFD) Serial Number	‘9F1E’	Mandatory
Merchant Category Code	‘9F15’	Optional
Merchant Name and Location	‘9F4E’	Optional
Program Identifier	‘DF70’	Optional
Reader Contactless CVM limit	Proprietary	Conditional
Reader Contactless Floor limit	Proprietary	Optional

Data Element	Tag	Presence
Reader Contactless Transaction limit	Proprietary	Optional
Status Check Support flag	Proprietary	Optional
Tearing Recovery Supported flag	Proprietary	Optional
Terminal Country Code	'9F1A'	Mandatory
Terminal Floor Limit	'9F1B'	Optional
Terminal Transaction Qualifier	'9F66'	Mandatory
Transaction Currency Code	'5F2A'	Mandatory
Transaction Date	'9A'	Mandatory
Transaction Status Information	'9B'	Mandatory
Transaction Type	'9C'	Mandatory
Transaction Verification Results	'95'	Mandatory
Unpredictable Number	'9F37'	Mandatory
Value Added Tax 1	'DF71'	Optional
Value Added Tax 2	'DF72'	Optional
Zero Amount Allowed flag	Proprietary	Optional
Zero Amount Allowed Offline flag	Proprietary	Optional

C.2. Online Only Terminals

The following table lists data elements that must be present in Terminals that are online capable only.

Table 4-16: Online Only Terminals

Data Element	Tag	Presence
Amount, Authorized	'9F02'	Mandatory
Amount, Other	'9F03'	Conditional
Application Identifier	'9F06'	Mandatory
Application Version Number	'9F09'	Mandatory
Certificate Authority Public Key Checksum	Proprietary	Not present
Certificate Authority Public Key Exponent	Proprietary	Not present
Certificate Authority Public Key Index	'9F22'	Not present
Certificate Authority Public Key Modulus	Proprietary	Not present
Data Container Read List	Proprietary	Optional

Data Element	Tag	Presence
Data Storage Supported flag	Proprietary	Optional
Deferred Authorization Supported flag	Proprietary	Optional
Extended Logging Supported flag	Proprietary	Optional
Extended Selection Support flag	Proprietary	Optional
Interface Device (IFD) Serial Number	‘9F1E’	Mandatory
Merchant Category Code	‘9F15’	Optional
Merchant Name and Location	‘9F4E’	Optional
Program Identifier	‘DF70’	Optional
Reader Contactless CVM limit	Proprietary	Conditional
Reader Contactless Floor limit	Proprietary	Optional
Reader Contactless Transaction limit	Proprietary	Optional
Status Check Support flag	Proprietary	Optional
Tearing Recovery Supported flag	Proprietary	Optional
Terminal Country Code	‘9F1A’	Mandatory
Terminal Floor Limit	‘9F1B’	Optional
Terminal Transaction Qualifier	‘9F66’	Mandatory
Transaction Currency Code	‘5F2A’	Mandatory
Transaction Date	‘9A’	Mandatory
Transaction Status Information	‘9B’	Mandatory
Transaction Type	‘9C’	Mandatory
Transaction Verification Result	‘95’	Mandatory
Unpredictable Number	‘9F37’	Mandatory
Value Added Tax 1	‘DF71’	Optional
Value Added Tax 2	‘DF72’	Optional
Zero Amount Allowed flag	Proprietary	Optional

C.3. Offline / Online Terminals

Table 4-17 lists data elements that must be present in Terminals that are offline and online capable.

Table 4-17: Online and Offline Terminals

Data Element	Tag	Presence
Amount, Authorized	'9F02'	Mandatory
Amount, Other	'9F03'	Conditional
Application Identifier	'9F06'	Mandatory
Application Version Number	'9F09'	Mandatory
Certificate Authority Public Key Checksum	Proprietary	Mandatory
Certificate Authority Public Key Exponent	Proprietary	Mandatory
Certificate Authority Public Key Index	'9F22'	Mandatory
Certificate Authority Public Key Modulus	Proprietary	Mandatory
Data Container Read List	Proprietary	Optional
Data Storage Supported flag	Proprietary	Optional
Deferred Authorization Supported flag	Proprietary	Optional
Extended Logging Supported flag	Proprietary	Optional
Extended Selection Support flag	Proprietary	Optional
Interface Device (IFD) Serial Number	'9F1E'	Mandatory
Merchant Category Code	'9F15'	Optional
Merchant Name and Location	'9F4E'	Optional
Program Identifier	'DF70'	Optional
Reader Contactless CVM limit	Proprietary	Conditional
Reader Contactless Floor limit	Proprietary	Optional
Reader Contactless Transaction limit	Proprietary	Optional
Status Check Support flag	Proprietary	Optional
Tearing Recovery Supported flag	Proprietary	Optional
Terminal Country Code	'9F1A'	Mandatory
Terminal Floor Limit	'9F1B'	Optional
Terminal Transaction Qualifiers	'9F66'	Mandatory
Transaction Currency Code	'5F2A'	Mandatory
Transaction Date	'9A'	Mandatory

Data Element	Tag	Presence
Transaction Status Information	‘9B’	Mandatory
Transaction Type	‘9C’	Mandatory
Transaction Verification Results	‘95’	Mandatory
Unpredictable Number	‘9F37’	Mandatory
Value Added Tax 1	‘DF71’	Optional
Value Added Tax 2	‘DF72’	Optional
Zero Amount Allowed flag	Proprietary	Optional
Zero Amount Allowed Offline flag	Proprietary	Optional

[This page is intentionally left blank.]

Annex D Data Elements Dictionary

This section focuses on the data elements required for the Kernel 6 Card application.

General Note: Reserved for Future Use

If a bit is specified as Reserved for Future Use (RFU), it must be:

- Set as specified or to 0 if no indication is provided or ignored, unless explicitly stated otherwise, and
- Excluded from any data field whose value is dependent on multiple bytes or bits.

D.1. Application Selection Registered Proprietary Data (tag '9F0A')

Data element that is conveyed to the terminal to enable market specific proprietary terminal functionality that is based on application proprietary data.

The value field of the Application Selection Registered Proprietary Data object follows the following format:

ID1, L1, V1, ID2, L2, V2,…

Where:

- ID is a two-byte Proprietary Data Identifier. Proprietary Data Identifiers are registered by EMVCo.
- L is the length of the value field coded in 1 byte (0 to 255).
- V is the value field. Its content is proprietary and format is out of scope of D-PAS.

The Application Selection Registered Proprietary Data is a primitive data object and its value field is not BER-TLV coded. In particular:

- IDs have no structure (they are not tags according to BER-TLV coding).
- The lengths L are always 1 byte.

IDs can appear in the Application Selection Registered Proprietary Data only if:

- they have been registered by EMVCo
- and their usage by the Terminal/Kernel is according to their intended usage, as agreed by EMVCo during registration.

Format: Binary, var.

D.2. Card ID (tag 'DF 3E')

This is a proprietary data element that contains a globally unique identifier for the card. Read by the Kernel during Read Application Data processing. Must match the identifier encoded in the Card Feature Descriptor (tag 'DF3B').

Format: Binary, variable length 11 to 64 bytes.

D.3. Card Feature Descriptor (tag ‘DF3B’)

This proprietary data element contains configuration settings for the contactless card application. These settings identify the card’s optional support for Data Storage, Extended Logging and Tearing Recovery. Contained in the card application’s FCI Issuer Discretionary Data.
Format: Binary, variable length, see Table 4-18.

Table 4-18: Card Feature Descriptor Encoding

Byte	Meaning
1	Bits 8-6: 000 (RFU) Bits 5-4: 00 (Reserved for transit use). Bit 3: 0 = Contactless Tearing Recovery is not supported, 1 = Contactless Tearing Recovery is supported. Bit 2: 0 = Extended Logging is not supported, 1 = Extended Logging is supported. Bit 1: 0 = Data Storage is not supported, 1 = Data Storage supported.
2	Short File Identifier (SFI) used for Data Storage*. Format *b*, encoded as: Bits 8-4: SFI (set to 00000b if the card does not support Data Storage) Bits 3-1: 000 (RFU)
3	The total number of Data Containers personalized for Data Storage. Format *b*. Set to ’00’ if the card does not support Data Storage.
4+	Card ID – the card’s globally unique identifier. Format *b*. Must match the Card ID (tag ‘DF3E’) personalized in one of the signed AFL records**.

* Data Store SFI is in the range 21 to 30.

** The terminal must treat ODA as failed if the Card ID data element is missing or does not match the value in the Card Feature Descriptor.

D.4. Card Feature Version Number (tag ‘DF3A’)

This proprietary data element is used by the card to indicate the format of the Card Feature Descriptor (tag ‘DF3B’). Contained in the card application’s FCI Issuer Discretionary Data.
The only valid value for Card Feature Version Number is ‘02’. All other values are RFU and out of scope of this specification.
Format: Binary, 1 byte.

D.5. Card Processing Requirements (tag ‘9F71’)

This is a Contactless Kernel 6 application proprietary data element used by the card to communicate the card processing requirements for the transaction and the card capabilities to the reader.
Format: Binary, 2 bytes

Table 4-19: Card Processing Requirement (CPR) Encoding

BYTE 1: Transient Data		
Bit	**Value**	**Meaning**
b8	1	Online PIN required
b7	1	Signature required
b6	1	PID Limit reached - Loyalty Transaction approved
b5	1	Consumer Device CVM Performed
b4	0	RFU
b3	0	RFU
b2	0	RFU
b1	0	RFU
BYTE 2: Permanent Data		
Bit	**Value**	**Meaning**
b8	1	Switch other interface if unable to process online
b7	1	Process online if CDA failed
b6	1	Decline/switch to other interface if CDA failed
b5	1	Issuer Update Processing supported
b4	1	Process online if card expired
b3	1	Decline if card expired
b2	1	CVM Fallback to Signature allowed
b1	1	CVM Fallback to No CVM allowed

D.6. Cryptogram Information Data (CID, tag '9F27')

The CID indicates the type of cryptogram (TC, ARQC, or AAC) returned by the card and the actions to be performed by the reader.
Format: Binary, 1 byte

Table 4-20: Cryptogram Information Data (CID) Encoding

BYTE 1		
Bit	**Value**	**Meaning**
b8-7	XX	00 = AAC 01 = TC 10 = ARQC 11 = RFU
b6-5	00	00 = Payment System-specific cryptogram
b4	0	0 = No advice required 1 = Advice required
b3-1	000	No information given

D.7. Cryptogram Version Number (CVN)

Cryptogram version number indicates the version of the algorithm used by the card to generate an Application Cryptogram (TC, ARQC, or AAC). The value of CVN is sent to the Issuer as part of the Issuer Application Data.
Format: Binary, 1 byte

D.8. Contactless Card Verification Results (tag '9F53')

This Contactless Kernel 6 application proprietary data element is used to inform the Issuer about the transaction context, the card decision, and the exceptions conditions that occurred during the current and previous transactions.
The Contactless CVR is used with Card Risk Management-Card Action Codes (CRM-CACs) and Card Verification Method-Card Action Code (CVM-CACs) during Card Action Analysis. This data element allows the Contactless application to determine the acceptable risk for processing the:

- Transaction over the contactless interface
- Transaction offline, or
- Transaction with no cardholder verification.

The Contactless CVR is transmitted to the Issuer as part of Issuer Application Data (tag '9F10').
Format: Binary, 8 bytes

Table 4-21: Contactless Card Verification Results (CL CVR) Encoding

BYTE 1: Information for Issuer (Card Decision)		
Bit	**Value**	**Meaning**
b8	1	Online PIN Required
b7	1	Signature Required
b6-5	XX	00 = AAC 01 = TC 10 = ARQC 11 = RFU
b4	1	PID Limit reached - Transaction approved by reader
b3-1	XXX	Script Counter indicating the number of the script processed during previous contact or contactless transaction
BYTE 2: Compared with CRM-CAC and CVM-CAC		
Bit	**Value**	**Meaning**
b8	1	Online cryptogram required (if not required, then reader is offline-capable and supports CDA)
b7	1	Transaction Type required to be processed online with online PIN CVM (e.g., purchase with cash-back, prepaid top-up, etc.)
b6	1	Transaction Type required to be processed offline without any CVM (e.g., refund transaction, prepaid, ticketing, offline balance inquiry, etc.)
b5	1	Domestic Transaction (based on Contactless-ACO setting)
b4	1	International Transaction
b3	1	PIN Try Limit exceeded (Dual-Interface implementation only)
b2	1	Confirmation Code Verification performed
b1	1	Confirmation Code Verification performed and failed

BYTE 3: CVM Related Actions		
Bit	**Value**	**Meaning**
b8	1	CVM Required
b7	1	CDCVM Local validation performed
b6	1	Consecutive CVM Transaction limit 1 exceeded (CVM-Cons 1)
b5	1	Consecutive CVM Transaction limit 2 exceeded (CVM-Cons 2)
b4	1	Cumulative CVM Transaction Amount limit 1 exceeded (CVM-Cum 1)
b3	1	Cumulative CVM Transaction Amount limit 2 exceeded (CVM-Cum 2)
b2	1	CVM Single Transaction Amount limit 1 exceeded (CVM-STA 1)
b1	1	CVM Single Transaction Amount limit 2 exceeded (CVM-STA 2)
BYTE 4: CRM Related Actions		
Bit	**Value**	**Meaning**
b8	1	CDA failed during previous contactless transaction
b7	1	Last contactless transaction not completed
b6	1	'Go on-line next transaction' was set by contact or contactless application
b5	1	Issuer Authentication failed during previous contact or contactless transaction
b4	1	Script failed on previous contact or contactless transaction
b3	1	Invalid PDOL check
b2	1	PDOL forced online (during GPO)
b1	1	PDOL forced decline (during GPO)
BYTE 5: CRM Related Actions		
Bit	**Value**	**Meaning**
b8	1	Consecutive Contactless Transaction limit exceeded (CL-Cons)
b7	1	Cumulative Contactless Transaction limit exceeded (CL-Cum)
b6	1	Single Contactless Transaction Amount limit exceeded (CL-STA)
b5	1	Lower Consecutive Offline Transaction limit exceeded (LCOL)
b4	1	Upper Consecutive Offline Transaction limit exceeded (UCOL)
b3	1	Lower Cumulative Offline Transaction Amount limit exceeded (LCOA)
b2	1	Upper Cumulative Offline Transaction Amount limit exceeded (UCOA)
b1	1	Single Transaction Amount limit exceeded (STA)

BYTE 6: Information for Issuer		
Bit	**Value**	**Meaning**
b8-5	XXXX	ID of PDOL-Decline or PDOL-Online check that forced the transaction to be declined or to go online (Only valid if CL –CVR B4b1 or B4b2 is set)
b4-1	XXXX	Transaction profile identifier (0000 by default)
BYTE 7: TTQ information for Issuer		
Bit	**Value**	**Meaning**
b8	1	Contact chip supported
b7	1	Offline-only reader
b6	1	Online PIN supported
b5	1	Signature supported
b4	1	Issuer Update Processing supported
b3	0	RFU
b2	1	Data Storage Failed limit exceeded (only if Data Storage supported)
b1	1	Data Storage Directory retrieved (only if Data Storage used)
BYTE 8: RFU		
Bit	**Value**	**Meaning**
b8-1	'XX'	Program ID used to process the transaction (if supported by Terminal) when configured as '6x', '7x', or '5x' (where x = 'A' to 'F')

D.9. Data Container Read List

This Kernel 6 proprietary data item is used during optional Read Data Storage processing, as described in section 3.2. The Data Container Read List contains a list of Container IDs, one for each Data Container to be read from Data Storage if allocated on the card.
Format: Terminal vendor specific

D.10. Data Storage Directory (tag 'DF3D')

This proprietary data object provides a listing of the Data Containers allocated on a card.
The value of the Data Storage Directory is computed dynamically by the card and may be obtained during Read Data Storage processing (using the GET DATA command, as described in section 3.2).
The Data Storage Directory comprises an unpredictable number (DSD_UN) and a sequence of Directory Entries, one for each Data Container allocated on the card ($0 \leq n \leq 24$).
The card shall omit the Directory Entries of any unallocated Data Container from the Data Storage Directory data object.

A Data Storage Directory comprising a sequence of zero Directory Entries indicates that no Data Containers are allocated on the card.
Format: Binary, variable length, see Figure 4-3 and Table 4-22.

Figure 4-3: Data Storage Directory

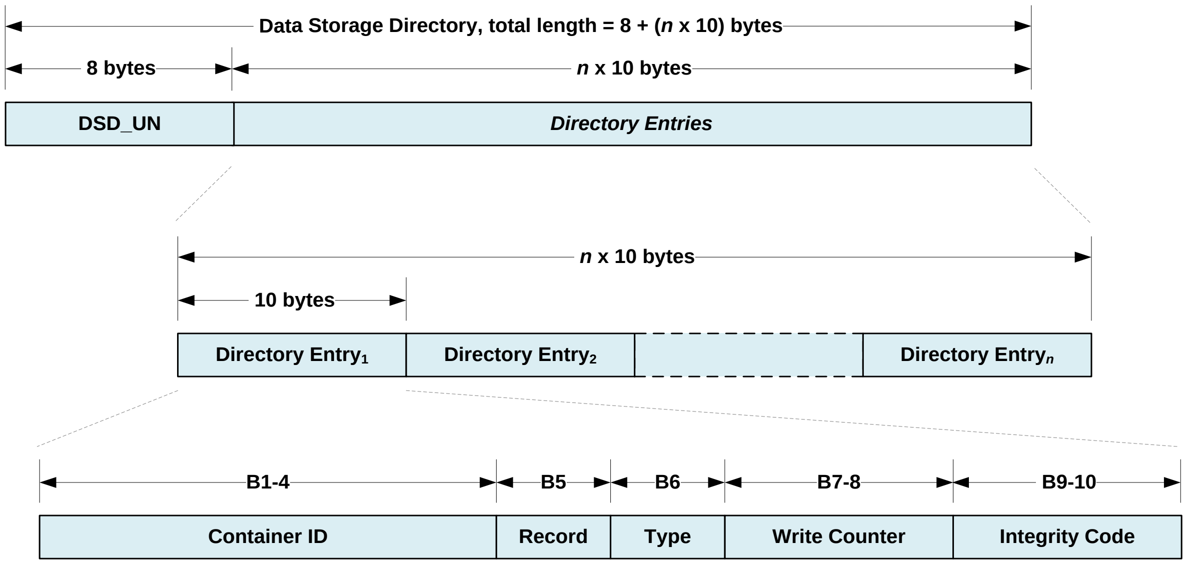


Table 4-22: Directory Entry Encoding

Byte	Meaning
1-4	Container ID – the Data Container's unique ID.
5	The record number of the associated Data Container, where:
6	The Data Container's type, where:

Byte 5:

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	0	0	0	RFU
x	x	x	x	x	x	x	x	Record number (1-24).
1	1	1	1	1	1	1	1	RFU

Byte 6:

b8	b7	b6	b5	b4	b3	b2	b1	Type
-	-	-	-	-	-	0	0	Unallocated
-	-	-	-	-	-	0	1	Issuer Container
-	-	-	-	-	-	1	0	Transient Container
-	-	-	-	-	-	1	1	Operator Container
x	x	x	x	x	x	-	-	RFU

Byte	Meaning
7-8	Write Counter – a counter incremented by the card each time the associated Data Container is updated. A container's initial Write Counter value is loaded during container allocation, and is defined by: • The Issuer for Operator Containers and Issuer Containers, or • The card for Transient Containers, and is set to 1 (specifically, it is set to zero during allocation but is immediately incremented by 1 because it has been updated).
9-10	Integrity Code – a value set by the card each time the associated Data Container is updated, where it is set to equal: `((leftmost 2 bytes of DSD_UN AND 'FFF0') OR (('00'||P1) AND '000F')` i.e., the Integrity Code equals the most significant three nibbles of DSD_UN and the least significant nibble of P1, where P1 is the command APDU's P1 value. **Note**: For data storage updates made using the DATA GPO or RESUME GPO commands, the least significant nibble of P1 contains the write commit flags. For updates made using the PUT DATA command, the least significant nibble equals 'F' (since P1/P2='DF3F').

D.11. Data Storage Directory Hash

The Data Storage Directory Hash (DSD_Hash) is a SHA-1 hash of the content of the Data Storage Directory (tag 'DF3D'), computed by the Kernel if and when the Data Storage Directory is retrieved. If computed, DSD_Hash is compared with the value present in the ICC Dynamic Data field of the card's Signed Dynamic Application Data during Offline Data Authentication.
Format: Binary, 20 bytes.

D.12. Data Storage Directory Unpredictable Number

The Data Storage Directory Unpredictable Number (DSD_UN) is an 8-byte unpredictable number generated by the card application when the Kernel retrieves the Data Storage Directory (using the GET DATA command). The computed DSD_UN is included in the Data Storage Directory response. The card application uses the value of DSD_UN to compute Update Code Hashes and Integrity Codes when updating data containers.
The operator is expected to use DSD_UN to compute the same Integrity Code value and include it in a MAC when updating Data Containers (see Annex E.2 for guidance).
Format: Binary, 8 bytes.

D.13. Data Storage Enabled

This is a Kernel 6 proprietary parameter that indicates if Data Storage is enabled for the current transaction. This transient flag defaults to false, and is set to true during Combination Selection Post Processing if Data Storage is supported by both terminal and card application.
Format: Terminal vendor specific Boolean flag

D.14. Data Storage Supported

This is a Kernel 6 proprietary terminal configuration parameter that identifies if the terminal supports use of Data Storage.
Format: Terminal vendor specific Boolean flag

D.15. Data Storage Write Results

This is a Kernel 6 proprietary parameter that counts the number Data Stores (tag 'DF3F') from an operator populated Data Storage Update Template (tag 'BF10') successfully written to a Card.
Format: Terminal vendor specific, Binary, 1 byte

Table 4-23: Data Storage Write Results Encoding

BYTE 1		
Bit	**Value**	**Meaning**
b8	1	All Data Stores successfully processed
b7	0	RFU
b6	0	RFU
b5-1	XXXXX	Write Counter indicating the number of the Data Stores successfully processed

D.16. Data Storage Update Template (tag 'BF10')

This is a Kernel 6 proprietary template that contains a sequence of Data Store ('DF3F') data objects.
Format: Binary, variable length.

D.17. Data Store (tag 'DF3F')

This is a proprietary and virtual data object used when updating Data Container content. Data Container updates can be made during Initiate Application Processing (using the DATA GPO command, as described in section 4.4), or during Write Data Storage processing (using the PUT DATA command, as described in section 4.8).

A Data Store data object comprises a Header and an Operator Data field, as shown in Figure 4-4.

The Header comprises the Container ID, Record number, container Type and, depending on the Data Container's type, two Update Code fields.

- The Record number must be '00'.
- The Type value (as defined in Table 4-22) must equal:
    - '02' when writing to a Transient Container, or
    - '03' when writing to an Operator Container.

The Operator Data comprises the Data Container's new content (empty containers are allowed).

If no Data Container is currently allocated on the card for the operator, then the operator is only allowed to put data into a Transient Container.

When updating a Transient Container, and if the given Container ID is currently unallocated, the card application allocates the next unallocated Transient Container. If there are no unallocated Transient Containers, the card application reallocates the "oldest" Transient Container (i.e. the Transient Container with the lowest Timestamp value). The container's previous content will be overwritten with the given Operator Data.

If the issuer has allocated an Operator Container to the operator, then the operator can update the container's contents. To do this, the operator must know the container's Update Code, which is used to compute an Update Code Hash. If the Update Code Hash provided by the operator matches the Update Code Hash calculated by the card (see Annex E.1), then the card replaces the container's contents with the given operator data.

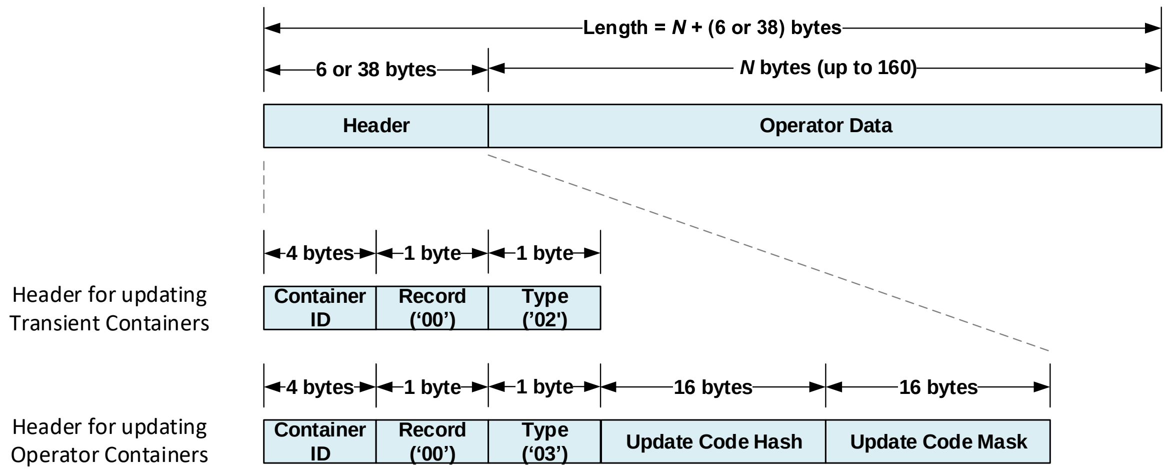
An initial Update Code value will be assigned to an Operator Container during allocation. The issuer and operator will need to agree on an appropriate value. The issuer may allocate Operator Containers either during card personalization or via issuer scripting.

As part of each successful Data Store update, the card application:

- Increments the Data Container's Write Counter by 1
- Sets the Data Container's Integrity Code value derived from DSD_UN
- Sets the Data Container's private Timestamp value to equal ATC
- For Operator Containers, sets the Update Code to equal:
  ```
  Update Code = Update Code XOR Update Code Mask
  ```
  (**Note**: An Update Code Mask of zero leaves the Update Code unchanged).

Format: Binary, variable length.

Figure 4-4: Data Store Data Object Encoding



D.18. Deferred Authorization Supported

This is a Kernel 6 proprietary terminal configuration parameter that identifies if the terminal is configured to support Deferred Authorizations.
Format: Terminal vendor specific

D.19. Extended Logging Data (tag 'DF3C')

This is a Kernel 6 proprietary data element used to include optional merchant data in the contactless card's transaction log during GET PROCESSING OPTIONS command processing.
Format: Binary, variable length (up to 32 bytes).

D.20. Extended Logging Enabled

This is a Kernel 6 proprietary parameter that indicates if Extended Logging is enabled for the current transaction. This transient flag defaults to false, and is set to true during Combination Selection Post Processing if Extended Logging is supported by both terminal and card application.
Format: Terminal vendor specific

D.21. Extended Logging Supported

This is a Kernel 6 proprietary terminal configuration parameter that identifies if the terminal supports use of Extended Logging.
Format: Terminal vendor specific

D.22. Extended Selection Support Flag

This is an optional internal data.
Format: Terminal vendor specific

D.23. Form Factor Identifier (tag '9F6E')

Element indicates the form factor of the device used. The value of the data element is proprietary and defined by Discover.
Format: Binary, 8 bytes.

D.24. Offline Balance (tag 'D1')

This is a Kernel 6 proprietary data element specifying the remaining amount of offline spending available for the application/transaction profile. The value of Offline Balance may be obtained from the application using the GET DATA command, if allowed by the card configuration: Offline balance = UCOA – COA. If the Offline balance cannot be calculated or is a negative value, then the card shall return a value of '0'.
Format: Numeric, 6 bytes.

D.25. Merchant Category Code (MCC, tag '9F15')

The MCC identifies the merchant type associated with the Terminal that is processing the transaction and is used to communicate that type to the card.
Format: Numeric, 2 bytes.

D.26. Merchant Name and Location (tag '9F4E')

The Merchant Name and Location identifies the name and location of the merchant associated with the Terminal that is processing the transaction. Data shall be coded according to [EMV Book 3].
Format: Alphanumeric (alphabetic upper case (A to Z) and numeric (0 to 9)), variable length.

D.27. Payment Account Reference (PAR, tag '9F24')

Static reference that represents the account that a PAN is associated with. This can be used to represent multiple PANs associated with an account where business processes must work across those PANs consistently (e.g. to enable merchants and acquirers to make a link between tokenized and non-tokenized PANs on the same payment account in order to enable back-office processing and value added services).
Format: Alphanumeric (alphabetic upper case (A to Z) and numeric (0 to 9)), 29 bytes.

D.28. Program Identifier (PID, tag ‘DF70’)

This is an optional field that represents the loyalty program associated with the Terminal. When defined in the Terminal, the Loyalty program Identifier must have a value of ‘6x’, ‘7x’, or ‘5x’ where x = ‘A’ to ‘F’.

The first four bits (6, 7, or 5) indicate to the card how to reach the reward, where:

- ‘6’ means: count each transaction where the amount is equal or greater than the PID Minimum Amount and is not a refund
- ‘7’ means: accumulate amount of each transaction where the amount is not equal to zero and is not a refund, and
- ‘5’ means: support both the loyalty programs identified as ‘6’ and ‘7’.

Format: Binary, 1 byte.

D.29. Reader Contactless CVM Limit

This data sets the CVM limit for a particular Combination based on the amount of the transaction. If the amount of the transactions is greater than or equal to this limit, the terminal will ask the card to perform Cardholder Verification.
Format: Numeric, 6 bytes.

D.30. Reader Contactless Floor Limit

This data sets the floor limit for a particular Combination. If the amount of the transactions is greater than this limit, the terminal will send the transaction online for processing.
Format: Numeric, 6 bytes.

D.31. Reader Contactless Transaction Limit

This data sets the reader transaction limit for a particular Combination. If the amount of the transactions is greater than or equal to this limit, the terminal will not process the transaction and inform the merchant to use a contact interface.
Format: Numeric, 6 bytes.

D.32. Resume Transaction

This internal flag indicates if a transaction is being resumed after a tear. This transient flag defaults to false, and is set to true during Application Selection processing if tearing recovery shall be performed.
Format: Terminal vendor specific

D.33. Tearing Recovery Enabled

This is a Kernel 6 proprietary parameter that indicates if Tearing Recovery is enabled for the current transaction. This transient flag defaults to false, and is set to true during Combination Selection Post Processing if Tearing Recovery is supported by both terminal and card application.
Format: Terminal vendor specific Boolean flag

D.34. Tearing Recovery Supported

This is a Kernel 6 proprietary terminal configuration parameter that identifies if the terminal supports use of Tearing Recovery.
Format: Terminal vendor specific Boolean flag

D.35. Tearing Log

This is a Kernel 6 proprietary data item used for Tearing Recovery. The Tearing Log records transaction detail sufficient for the Kernel to resume transaction processing in the event of the card being re-presented to the reader after a tear. Table 4-24 shows the minimum data items required for the Tearing Log.
The Terminal must persist the Tearing Log between Kernel restarts.
Format: Terminal vendor specific

Table 4-24: Minimum Data Items Recorded in Tearing Log

Data Item	Tag	Source	Description
Application Identifier (AID)	‘9F06’	POS	The AID selected by Entry Point
Card Feature Descriptor	‘DF3B’	Card	Contains the card’s globally unique identifier
Card Feature Version Number	‘DF3A’	Card	The version number of the card’s feature descriptor
PDOL Data	-	Kernel 6	The data element values for the associated PDOL
APDU P1	-	Kernel 6	The P1 parameter required for RESUME GPO

D.36. Terminal Transaction Qualifier (TTQ, tag ‘9F66’)

Terminal Transaction Qualifiers (TTQs) are online and CVM processing options that may be supported by the Terminal during Entry Point processing. Detailed definitions of TTQs are provided in Table 4-25.
Format: Binary, 4 bytes (as specified in EMV BOOK A)

Table 4-25: Terminal Transaction Qualifiers (TTQ) Encoding

BYTE 1: Reader Capabilities		
Bit	**Name**	**Meaning**
b8	Mag stripe mode supported	0 = Not supported, 1 = Supported
b7	0	RFU
b6	EMV mode supported	0 = Not supported, 1 = Supported
b5	EMV contact chip supported[1]	0 = Not supported, 1 = Supported
b4	Offline-only Contactless Reader	0 = Online capable, 1 = Offline Only
b3	Online PIN supported	0 = Not supported, 1 = Supported
b2	Signature supported	0 = Not supported, 1 = Supported
b1	Offline Data Authentication for Online Authorizations supported	0 = Not supported, 1 = Supported
BYTE 2: Reader CVM Requirements		
Bit	**Name**	**Meaning**
b8	Online cryptogram required[2]	0 = Not required, 1 = Required
b7	CVM required[2]	0 = Not required, 1 = Required
b6	(Contact Chip) Offline PIN supported[1]	0 = Not supported, 1 = Supported
b5	0	RFU
b4	Fast Mode Supported[3]	0 = Not supported, 1 = Supported
b3	Aggregate Mode[4]	0 = Default Profile Selected; 1 = Different Profile Selected
b2	0	RFU
b1	0	RFU
BYTE 3: Reader Additional Capabilities		
Bit	**Name**	**Meaning**
b8	Issuer Update Processing supported	0 = Not supported, 1 = Supported

b7	Consumer Device CVM supported	0 = Not supported, 1 = Supported
b6	0	RFU
b5	0	RFU
b4	Consumer Device CVM required	0 = Not supported, 1 = Supported
b3	0	RFU
b2	0	RFU
b1	0	RFU
BYTE 4		
Bit	**Name**	**Meaning**
b8	1	RFU
b7	1	RFU
b6	1	RFU
b5	1	RFU
b4	1	RFU
b3	1	RFU
b2	1	RFU
b1	1	RFU

1 This bit shall be set to ‘1’ if the Terminal has a contact interface. Otherwise, it shall be set to ‘0’ if the Terminal is a contactless only Terminal.

2 This bit is set dynamically based on the pre-processing result.

3 Fast mode support will allow “No-CDCVM” transactions. This bit can be used for specific implementations e.g. transit.

4 This bit can be used for specific implementations e. g. transit, to select a profile different from the default profile.

D.37. Terminal Verification Result (TVR, tag ‘95’)

The TVR is a bitmap that shows the result of some functions processed by the Terminal. The Terminal Verification Result is coded according to Annex C5 of [EMV Book 2].
Format: Binary, 5 bytes.

Table 4-26: Terminal Verification Result (TVR) Encoding

BYTE 1								
b8	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**Meaning**
1	-	-	-	-	-	-	-	Offline data authentication was not performed
-	1	-	-	-	-	-	-	SDA Failed[1]
-	-	1	-	-	-	-	-	ICC Data missing
-	-	-	1	-	-	-	-	Card appears on Terminal exception file
-	-	-	-	1	-	-	-	DDA failed[1]
-	-	-	-	-	1	-	-	CDA failed
-	-	-	-	-	-	1	-	SDA selected[1]
-	-	-	-	-	-	-	0	RFU
BYTE 2								
b8	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**Meaning**
1	-	-	-	-	-	-	-	ICC and Terminal have different application versions
-	1	-	-	-	-	-	-	Expired application
-	-	1	-	-	-	-	-	Application not yet effective
-	-	-	1	-	-	-	-	Requested service not allowed for card product
-	-	-	-	1	-	-	-	New card
-	-	-	-	-	0	-	-	RFU
-	-	-	-	-	-	0	-	RFU
-	-	-	-	-	-	-	0	RFU
BYTE 3								
b8	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**Meaning**
1	-	-	-	-	-	-	-	Cardholder verification was not successful
-	1	-	-	-	-	-	-	Unrecognized CVM
-	-	1	-	-	-	-	-	PIN Try Limit exceeded

-	-	-	1	-	-	-	-	PIN entry required and PIN pad not present or not working[1]
-	-	-	-	1	-	-	-	PIN entry required, PIN pad present, but PIN was not entered[1]
-	-	-	-	-	1	-	-	Online PIN entered
-	-	-	-	-	-	0	-	RFU
-	-	-	-	-	-	-	0	RFU
BYTE 4								
b8	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**Meaning**
1	-	-	-	-	-	-	-	Transaction exceeds floor limit
-	1	-	-	-	-	-	-	Lower Consecutive offline limit exceeded[1]
-	-	1	-	-	-	-	-	Upper Consecutive offline limit exceeded[1]
-	-	-	1	-	-	-	-	Transaction selected randomly for online processing
-	-	-	-	1	-	-	-	Merchant forced transaction online[1]
-	-	-	-	-	0	-	-	RFU
-	-	-	-	-	-	0	-	RFU
-	-	-	-	-	-	-	0	RFU
BYTE 5								
b8	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**Meaning**
1	-	-	-	-	-	-	-	Default TDOL used[1]
-	1	-	-	-	-	-	-	Issuer Authentication failed
-	-	1	-	-	-	-	-	Script processing failed before final GENERATE AC[1]
-	-	-	1	-	-	-	-	Script processing failed after final GENERATE AC[1]
-	-	-	-	0	-	-	-	Reserved for use by the EMV Contactless Specifications
-	-	-	-	-	0	-	-	Reserved for use by the EMV Contactless Specifications
-	-	-	-	-	-	0	-	Reserved for use by the EMV Contactless Specifications
-	-	-	-	-	-	-	0	Reserved for use by the EMV Contactless Specifications

Note [1]: These bits are only present for compatibility with contact TVR and should be set to '0' or ignored by Issuer.

D.38. Transaction Certificate (TC)

This data may be returned by the card in response to the GET PROCESSING OPTIONS command, when the card approves the transaction offline. The value is part of Signed Dynamic Application Data (SDAD). For more information, refer to [EMV Book 3 and 4].
Format: Binary, 8 bytes.

D.39. Transaction Status Information (TSI, tag '9B')

This data indicates which operation(s) have been performed during the current transaction.
Format: Binary, 2 bytes.

Table 4-27: Transaction Status Information (Byte 1)

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	X	X	X	X	X	X	X	Offline data authentication was performed
X	1	X	X	X	X	X	X	Cardholder verification was performed
X	X	1	X	X	X	X	X	Card risk management was performed1
X	X	X	1	X	X	X	X	Issuer authentication was performed
X	X	X	X	1	X	X	X	Terminal risk management was performed
X	X	X	X	X	1	X	X	Script processing was performed
X	X	X	X	X	X	0	X	RFU
X	X	X	X	X	X	X	0	RFU

Note [1]: this bit shall always be set to '1' if the card returns '9000' to GET PROCESSING OPTIONS.

Table 4-28: Transaction Status Information (Byte 2)

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	X	X	X	X	X	X	X	RFU
X	0	X	X	X	X	X	X	RFU
X	X	0	X	X	X	X	X	RFU
X	X	X	0	X	X	X	X	RFU
X	X	X	X	0	X	X	X	RFU

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
X	X	X	X	X	0	X	X	RFU
X	X	X	X	X	X	0	X	RFU
X	X	X	X	X	X	X	0	RFU

D.40. Write Data Storage Template (tag 'BF11')

This Kernel 6 proprietary data item is used during optional Write Data Storage processing, as described in section 3.13. The Write Data Storage Template is a composite data object that contains the data elements defined in Table 4-29.
Format: Terminal vendor specific

Table 4-29: Write Data Storage Template

Tag-Length	Value			Length	Presence
'BF11'	Write Data Storage Template			Var.	M
	'DF3A'	Card Feature Version Number		1	M
	'DF3B'	Card Feature Descriptor		Var.	M
	'BF10'	Data Storage Update Template		Var.	M
		'DF3F'	Data Store (1st data container update)	Var.	O
			…		
		'DF3F'	Data Store (*n*th data container update)	Var.	O

D.41. Zero Amount Allowed Flag

This is an optional internal data.

D.42. Zero Amount Allowed Offline Flag

This is an optional internal data.
Format: Terminal vendor specific

[This page is intentionally left blank.]

Annex E Data Storage Security

E.1. Operator Container Update Codes

This section provides guidance for the management of Operator Container Update Codes. The information in this section is not meant to be restrictive, in that other methods of meeting the same requirements may also be considered.
Data Storage uses Update Codes to implement write-access control for Operator Containers. The card application holds a hidden Update Code for each Operator Container, and will only allow an update to an Operator Container if the operator can provide a correct Update Code Hash value calculated with the Update Code. This works as follows:

- The operator uses the following items of card data:

Data Item	Source
Card ID	Card Feature Descriptor (tag 'DF3B')
DSD_UN	Data Storage Directory (tag 'DF3D')
Write Counter	Destination container's Directory Entry in the Data Storage Directory
Integrity Code	Destination container's Directory Entry in the Data Storage Directory
Old content	Destination container's current content

- The operator uses the card data to determine the correct Update Code. The operator may use any appropriate strategy to manage its Update Codes. For example, a value embedded in the container's content could be used to derive a card-specific value from a master value.
- The operator prepares an Update Code Mask. This shall have a value of zero (eight bytes of '00') unless the operator wants to change the Update Code, in which case:
  ```
  Update Code Mask = Update Code XOR New Update Code
  ```
- The operator uses the Update Code to compute an Update Code Hash as a hash over the concatenated Container ID, card data items, and the container's new content, where:
  ```
  Update Code Hash = leftmost 8 bytes of SHA-1(Container ID ||
  Write Counter || Integrity Code || DSD_UN || Update Code Mask
  || Operator Data || Update Code)
  ```
- The Operator Container update is formatted as a Data Store object and sent to the card during Initiate Application Processing or Write Data Storage.
- The card application computes its own Update Code Hash value, using its own values for Write Counter, Integrity Code, DSD_UN and Update Code (held privately), and the given Container ID, Operator Data and Update Code Mask.
- The card application compares its computed Update Code Hash with the given Update Code Hash. If the values do not match, then the update request is rejected, otherwise:
  - The card application writes the new content into the container's file record,
  - The card application updates the container's Write Counter, Integrity Code, and
  - The card application sets the container's hidden Update Code to equal:

```
Update Code = Update Code XOR Update Code Mask
```

(**Note**: An Update Code Mask of zero leaves the Update Code unchanged).

An initial Update Code value will be assigned to an Operator Container during allocation. The issuer and operator will need to agree on an appropriate value. The issuer may allocate Operator Containers either during card personalization or via Issuer Scripting.

E.2. Data Container Integrity

This section provides guidance for the verification of Data Container integrity. The information in this section is not meant to be restrictive, in that other methods of meeting the same requirements may also be considered.

It is recommended that operators include a Message Authentication Code (MAC) within the body of their container contents to allow for data integrity verification.

It is recommended that the MAC include the container's Write Counter and Integrity Code, as well as the Operator's own data content. This will guard against data skimming (copying data from one card to another) and data replay (overwriting containers with old content).

The approach illustrated in Figure 4-5 and Figure 4-6 will support this.

Note: When writing to an unallocated Transient Container (i.e. there is no Directory Entry for the container in the card's Data Storage Directory), the initial Write Counter value will equal the maximum Write Counter value of any of the card's Directory Entries (or zero, if there are none).

Figure 4-5: Preparing Data Container MAC

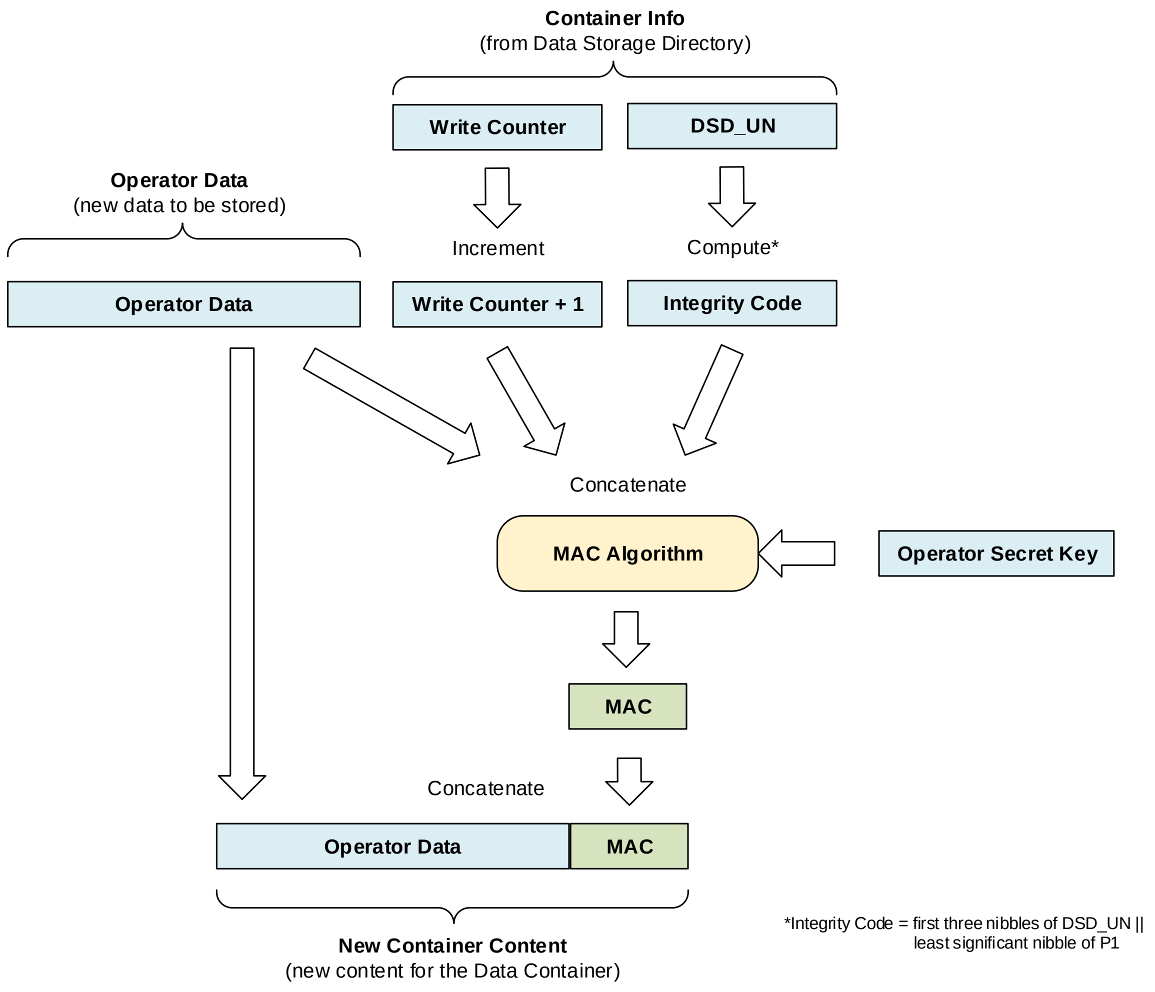
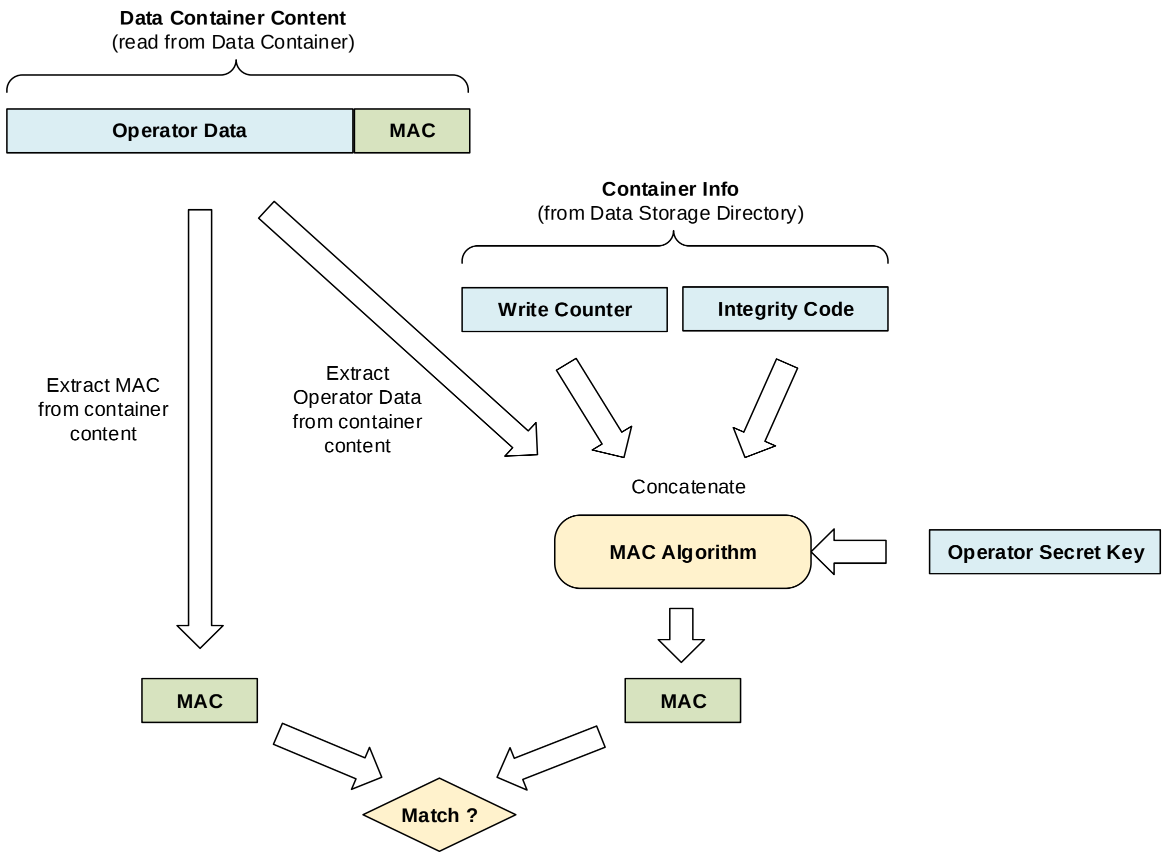


Figure 4-6: Checking Data Container MAC



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