EMV®
Contactless Specifications for Payment Systems

Book C-8

Kernel 8 Specification

Version 1.1
June 2023

Legal Notice

The EMV® Specifications are provided “AS IS” without warranties of any kind, and EMVCo neither assumes nor accepts any liability for any errors or omissions contained in these Specifications. EMVCO DISCLAIMS ALL REPRESENTATIONS AND WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AS TO THESE SPECIFICATIONS.

EMVCo makes no representations or warranties with respect to intellectual property rights of any third parties in or in relation to the Specifications. EMVCo undertakes no responsibility to determine whether any implementation of the EMV Specifications may violate, infringe, or otherwise exercise the patent, copyright, trademark, trade secret, know-how, or other intellectual property rights of third parties, and thus any person who implements any part of the EMV Specifications should consult an intellectual property attorney before any such implementation.

Without limiting the foregoing, the Specifications may provide for the use of public key encryption and other technology, which may be the subject matter of patents in several countries. Any party seeking to implement these Specifications is solely responsible for determining whether its activities require a license to any such technology, including for patents on public key encryption technology. EMVCo shall not be liable under any theory for any party’s infringement of any intellectual property rights in connection with the EMV Specifications.

Revision Log – Version 1.1

This section outlines notable updates that have been made to this specification since the publication of the *EMV Contactless Specifications for Payment Systems, Book C-8 Kernel 8 Specification v1.0*.

- Information about cryptographic algorithms and implementations has been moved to *EMV Contactless Specifications for Payment Systems, Book E – Security and Key Management*.
  - The following sections have been moved from Book C-8 to Book E:
    - Chapter 8: Security Algorithms
    - Annex B: ECC Certificate Format
    - Annex C: Algorithm Suite Indicators
    - Annex D: Curve P-256
  - Security related functions that have been moved to Book E include:
    - Secure channel use for Privacy protection.
    - Unpredictable number generation.
    - Checks on the Certification Authority Public Key type.
    - Definitions of Certificate Algorithm Suite Indicators and Secure Channel Algorithm Suite Indicators.
    - Generation of ECC public and private key pairs and recovery of the blinded public key.
    - Derivation of Session Keys for Confidentiality and Integrity.
- Changes to data management handling include:
  - Updates to how the TLV Database is instantiated and populated.
  - Improved definitions of some data types and how they are handled in the TLV Database.
  - ParseAndStoreCardResponse service updates to align with changes to data types and TLV Database operation.
  - Variables scope changes from local to global – Start Time, Response Message Data Field.
  - Improved DOL handling rules.
  - Clean up of how proprietary tags are handled and general data management and handling.
  - Constructing Extended SDA Tag List Related Data clarified to improve length handling.

List handling has been updated.

List encoding has been clarified.

Tag Length Checking has been clarified.

ACT Signal (i.e. the transaction data items signal) now has a minimum set of data.

Removal of redundant data

Removed unused subfields of the User interface Request Data.

TVR Unrecognised CVM bit removed as it was not used.

Other changes include:

Clarifications of some terminology and definitions.

Token Requestor ID added to the data dictionary.

The IAD MAC is now stored in the TLV Database once calculated.

General corrections, typographical error corrections, and formatting adjustments.

Contents

Tables

Figures

1 Using This Manual

1.1 Purpose

This document, *EMV® Contactless Specifications for Payment Systems, Book C-8 – Kernel 8 Specification*, should be read in conjunction with:

- *EMV Contactless Specifications for Payment Systems, Book A – Architecture and General Requirements*, hereafter referred to as [EMV Book A]
- *EMV Contactless Specifications for Payment Systems, Book B – Entry Point Specification*, hereafter referred to as [EMV Book B]
- *EMV Contactless Specifications for Payment Systems, Book E – Security and Key Management*, hereafter referred to as [EMV Book E]

This document defines the behaviour of the Kernel used in combination with cards having a Kernel Identifier indicating Kernel 8, as defined in [EMV Book B].

Note: While this kernel is compatible with any Entry Point version, payment systems may specify a minimum Entry Point version that they require. For example, if the Kernel Identifier–Terminal (tag '96') data object is to be used, then the minimum version of Entry Point is Version 2.10 with Specification Bulletin 268.

1.2 Audience

This specification is intended for use by manufacturers of contactless readers and terminals. It may also be of interest to manufacturers of contactless cards and to financial institution staff responsible for implementing financial applications in contactless cards.

1.3 Related Information

The following references are used in this document. It is noted that the latest version applies unless a publication date is explicitly stated.

Table 1.1—Related Information

Reference	Document Title
[EMV Book 1]	Integrated Circuit Card Specifications for Payment Systems – Book 1, Application Independent ICC to Terminal Interface Requirements

Reference	Document Title
[EMV Book 2]	Integrated Circuit Card Specifications for Payment Systems – Book 2, Security and Key Management
[EMV Book 3]	Integrated Circuit Card Specifications for Payment Systems – Book 3, Application Specification
[EMV Book 4]	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
[EMV Book E]	EMV® Contactless Specifications for Payment Systems, Book E – Security and Key Management
[EMV CL L1]	EMV® Level 1 Specifications for Payment Systems, EMV Contactless Interface Specification
[EMV Token]	EMV® Payment Tokenisation Specification, Technical Framework
[ISO 639-1]	Codes for the representation of names of languages – Part 1: Alpha-2 Code
[ISO 3166-1]	Codes for the representation of names of countries and their subdivisions – Part 1: Country codes
[ISO 4217]	Codes for the representation of currencies and funds
[ISO/IEC 7813]	Information technology — Identification cards — Financial transaction cards
[ISO/IEC 7816-4]	Identification cards — Integrated circuit(s) cards with contacts — Part 4: Organization, security and commands for interchange
[ISO/IEC 7816-5]	Registration of application providers
[ISO 8583:1987]	Financial transaction card originated messages – Interchange message specifications
[ISO 8583:1993]	Financial transaction card originated messages – Interchange message specifications

Reference	Document Title
[ISO/IEC 8825-1]	Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)
[ISO/IEC 8859]	Information technology – 8-bit single-byte coded graphic character sets
[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 14888-3]	Information technology — Security techniques — Digital signatures with appendix — Part 3: Discrete logarithm based mechanisms
[ISO/IEC 18031:2005]	Information technology – Security techniques – Random bit generation
[NIST SP800-22A]	A statistical test suite for random and pseudorandom number generators for cryptographic algorithms

1.4 Terminology

The following terms are used in this document, carrying specialised meanings as indicated.

Table 1.2—Terminology

Term	Description
Card	Card, as used in these specifications, is a consumer device supporting contactless transactions.
Combination	Combination is the combination of an AID and a Kernel ID.
Configuration Option	A Configuration Option allows activation or deactivation of the Kernel software behind this option. The Configuration Option may change the execution path of the software but it does not change the software itself. A Configuration Option is set in the Kernel database per AID and Transaction Type.
Implementation Option	An Implementation Option allows the vendor to select whether the functionality behind the option will be implemented in a particular installation.
Kernel	The Kernel contains the interface routines, security and control functions, and logic to manage a set of commands and responses to retrieve all the necessary data from the Card to complete a transaction. The Kernel processing covers the interaction with the Card between the selection of the card application (excluded) and the processing of the transaction's outcome (excluded).
NULL	The state of a data object of a given type with no value.
POS System	The POS System is the collective term given to the payment infrastructure present at the merchant. It is made up of the Terminal and Reader.
Process	A Process is a logical component within a Reader that has one or more Queues to receive Signals. The processing of Signals, in combination with the carried data, may then generate other Signals to be sent. Processing can continue until all the Queues of a Process are empty, or until the Process terminates.

Term	Description
Queue	A Queue is a buffer that stores events to be processed. The events are stored in the order received.
Reader	The Reader is the device that supports the Kernel(s) and provides the contactless interface used by the Card. In this specification, the Reader is considered as a separate logical entity, although it can be an integral part of the POS System.
Signal	A Signal is an asynchronous event that is placed in a Queue. A Signal can convey data as parameters, and the data provided in this way is used in the processing of the Signal.
String	A sequence of bytes.
Terminal	The Terminal is the device that connects to the authorisation and/or clearing network and that together with the Reader makes up the POS System. The Terminal and the Reader may exist in a single integrated device. However, in this specification, they are considered separate logical entities.

1.5 Notations

1.5.1 State Machine

This document specifies the Kernel processing as a state machine that is triggered by Signals that cause state transitions. The application states of the Kernel are written in a specific format to distinguish them from the rest of the text:

`State`

Example:

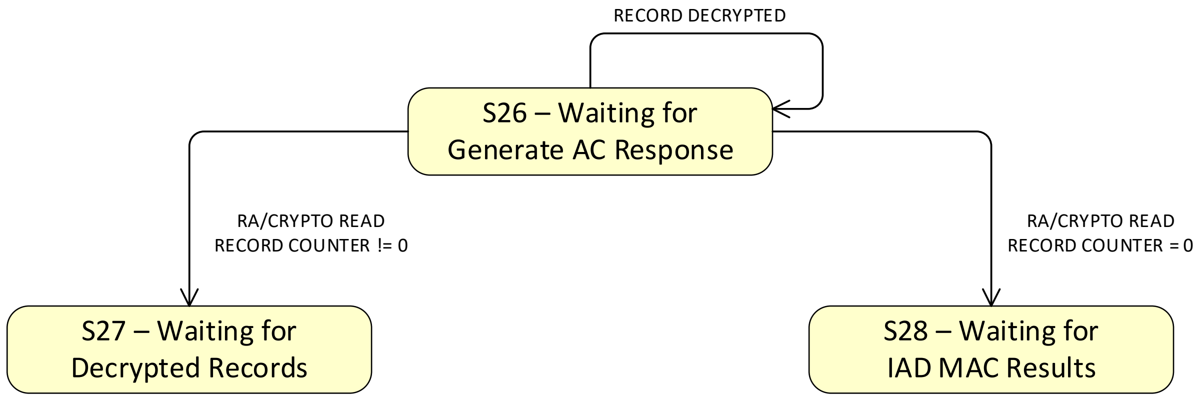
`S22 – Waiting for Read Record Response`

The state machine of the Kernel is represented using a state diagram.

Example:

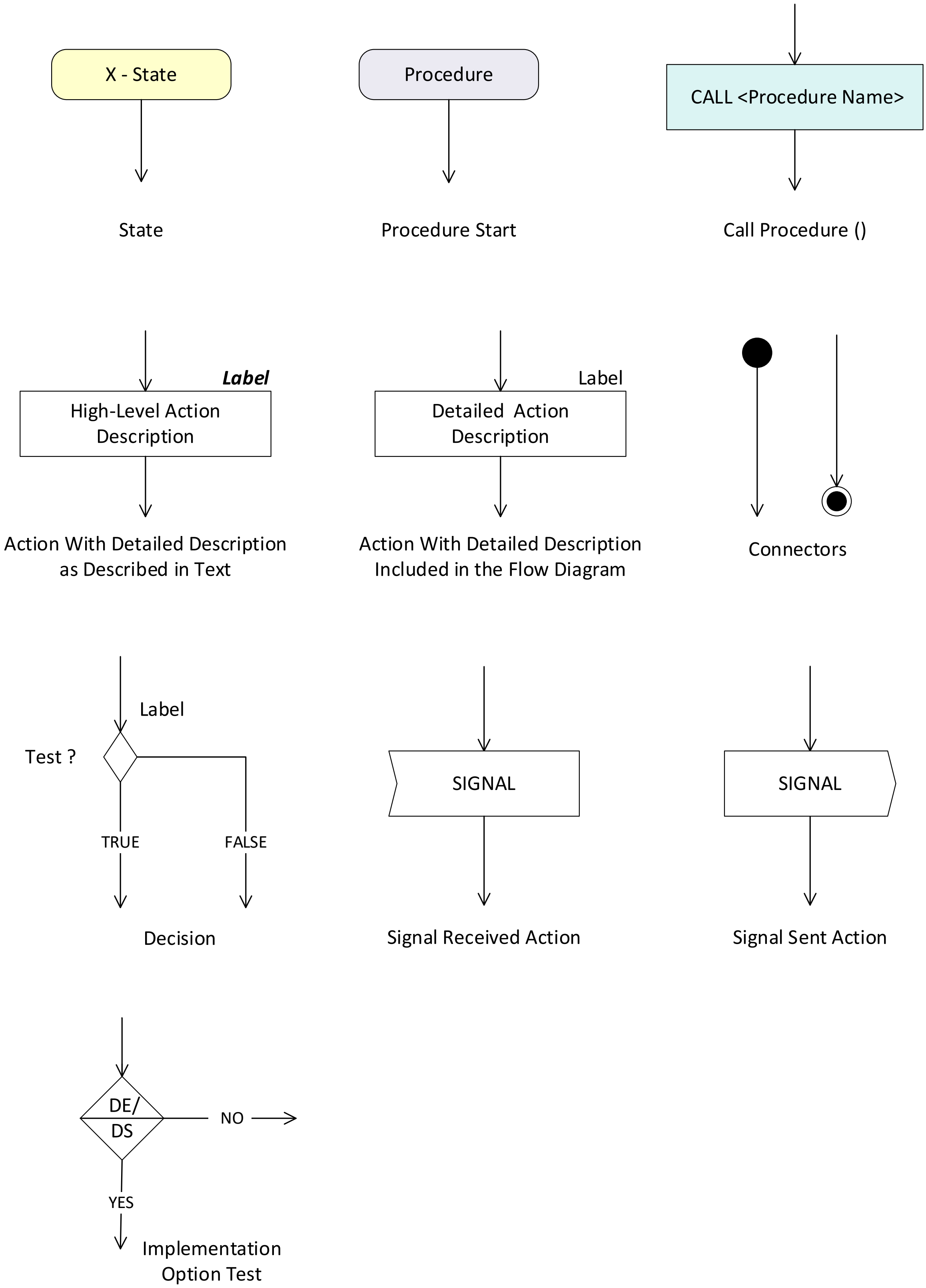
Figure 1.1 depicts the transitions for state `S26 – Waiting for Generate AC Response`. Upon receiving the Signal RECORD DECRYPTED, the state machine remains in state `S26 – Waiting for Generate AC Response`. The state machine leaves state `S26 – Waiting for Generate AC Response` upon receiving an RA Signal. In this case, the state machine goes to `S28 – Waiting for IAD MAC Results` if Crypto Read Record Counter = 0 and to `S27 – Waiting for Decrypted Records` if Crypto Read Record Counter ≠ 0.

Figure 1.1—Example of State Diagram Notation



This document uses a combination of flow diagrams and textual description to describe the state transitions in the state machine of the Kernel. Figure 1.2 depicts the symbols used in the flow diagrams.

Figure 1.2—Symbols Used in Flow Diagrams



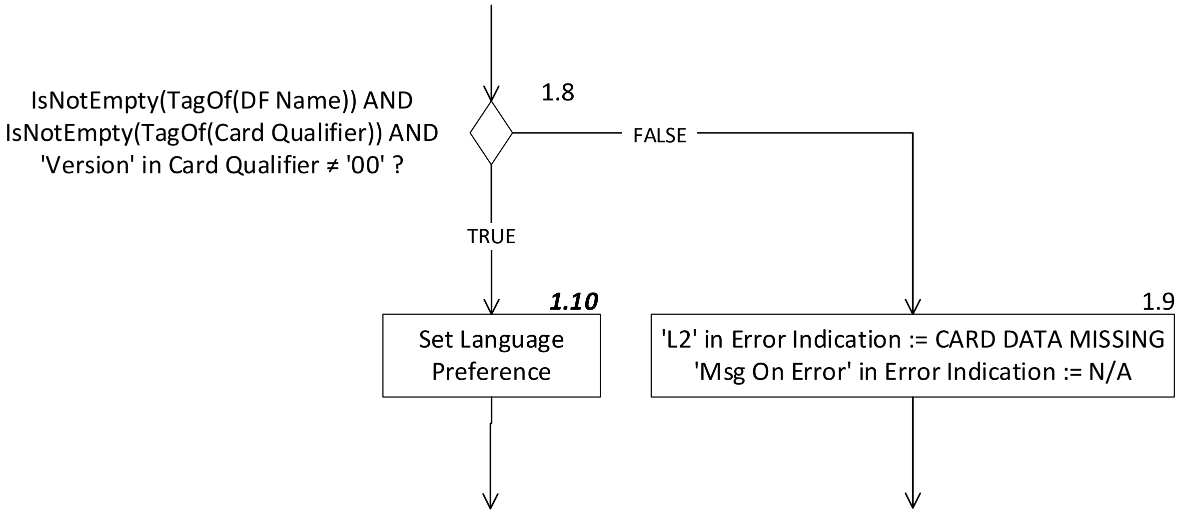
On the Kernel behaviour, the combination of the flow diagrams and the corresponding textual descriptions constitutes the requirements:

- Each diagram in this specification has a unique label.
- Each symbol in a diagram has a unique identifier that is the concatenation of the diagram label with the symbol number.
- The textual description corresponding to a symbol may be included in the diagram or it may be a detailed textual description included in the text below the diagram.

The flow diagrams are read from top to bottom and define the order of execution of the processing steps. The textual description specifies the behaviour of the individual steps but bears no information on the order of execution.

An example of a flow diagram is given in Figure 1.3 in combination with the detailed textual description below.

Figure 1.3—Example of Flow Diagram



1.10

If the Language Preference is returned from the Card, then copy it to 'Language Preference' in User Interface Request Data 1 and User Interface Request Data 2:

IF [IsNotEmpty(TagOf(Language Preference))]

THEN

'Language Preference' in User Interface Request Data 1 := Language Preference

'Language Preference' in User Interface Request Data 2 := Language Preference

If the length of Language Preference is less than 8 bytes, then pad 'Language Preference' in User Interface Request Data 1 and User Interface Request Data 2 with trailing hexadecimal zeroes to 8 bytes.

ENDIF

Only symbol 1.10 has a detailed textual description. The textual descriptions of symbols 1.9 and 1.8 are included in the flow diagram.

The requirements are related to the behaviour of the Kernel, while leaving flexibility in the actual implementation. The implementation must behave in a way that is indistinguishable from the behaviour specified in this document, in that it creates the output as predicted by this specification for a given input. There is no requirement that the implementation realises the behaviour through a state machine as described in this document.

1.5.2 Data Object Notation

Data objects used for this specification are capitalised:

Data Object Name

Example:

Application File Locator

To refer to a sub-element of a data object (i.e. a specific bit, set of bits, or byte of a multi-byte data object), the following notational convention is used:

Named sub-element

If the sub-element is defined in the data dictionary (Annex A), with each possible value of the sub-element having a name, then the following conventions apply:

- The reference to the sub-element is 'Name of Sub-element' in Data Object Name.
- The reference to the value is VALUE OF SUB-ELEMENT.

Examples:

'CVM Limit exceeded' in Terminal Risk Management Data refers to bit 8 of byte 2 in Terminal Risk Management Data.

'CVM' in Outcome Parameter Set := ONLINE PIN means the same as bits 8 to 5 of byte 4 of Outcome Parameter Set are set to 0010b.

Sub-element identified by index

Alternatively, an index may be used to identify a sub-element of a data object. In this case the following notational conventions apply:

- To refer to a specific byte of a multi-byte data object, a byte index is used within brackets (i.e. [ ]). The first byte (leftmost or most significant) of a data object has index 1.

  For example, Terminal Verification Results[2] represents byte 2 of Terminal Verification Results.

- To refer to a specific bit of a single byte multi-bit data object, a bit index is used within brackets [ ]. The first bit (rightmost or least significant) of a data object has index 1.

  For example, Cryptogram Information Data[7] represents bit 7 of the Cryptogram Information Data.

- To refer to a specific bit of a multi-byte data object, a byte index and a bit index are used within brackets (i.e. [ ][ ]).

  For example, Terminal Verification Results[2][4] represents bit 4 of byte 2 of the Terminal Verification Results.

- Ranges of bytes are expressed using the x:y notational convention:

  For example, Terminal Verification Results[1:4] represents bytes 1, 2, 3, and 4 of the Terminal Verification Results.

- Ranges of bits are expressed using the y:x notational convention:

  For example, Cryptogram Information Data[5:1] represents bits 5, 4, 3, 2, and 1 of the Cryptogram Information Data.

  For example, Application File Locator [1][8:4] represents bits 4 to 8 of byte 1 of AFL.

1.5.3 Other Notational Conventions

Notations for processing data and managing memory are described in Table 1.3.

Table 1.3—Other Notational Conventions

Notation	Meaning	Example
'0' to '9' and 'A' to 'F'	Hexadecimal notation. Values expressed in hexadecimal form are enclosed in straight single quotes.	27509 decimal is expressed in hexadecimal as '6B75'.
1001b	Binary notation. Values expressed in binary form are followed by the letter b.	'08' hexadecimal is expressed in binary as 00001000b.
340	Decimal notation. Values expressed in decimal form are not enclosed in single quotes.	'0C' hexadecimal is expressed in decimal as 12.
C-APDU	C-APDUs are written in all caps to distinguish them from the text	GET PROCESSING OPTIONS
SET	A specific bit in a data object is set to the value 1b	SET 'Kernel 8 processing and TVR format' in Terminal Verification Results
CLEAR	A specific bit in a data object is set to the value 0b	CLEAR 'Cardholder verification was not successful' in Terminal Verification Results
:=	A specific value is assigned to a data object or to a sub-element of a data object	'Status' in Outcome Parameter Set := END APPLICATION
=	This notation is used for both the logical and bitwise equivalence operations. Its meaning is therefore context-specific.	Logical equivalence: L = GetLength(T) V = GetValue(T)
OR	This notation is used for both the logical and bitwise OR operations. Its meaning is therefore context-specific.	Logical OR: IF [IsNotPresent(T) OR IsEmpty(T)]

Notation	Meaning	Example
AND	This notation is used for both the logical and bitwise AND operation. Its meaning is therefore context-specific.	Logical AND: IF [IsNotEmptyList(Data To Send) AND IsEmptyList(Tags To Read Yet)] Bitwise AND: TVR := (TVR AND Kernel Reserved TVR Mask)
NOT	This notation is used for both the logical and bitwise negation operation. Therefore, its meaning is context-specific.	Logical NOT: IF [NOT ParseAndStoreCardResponse (TLV)]
||	Two values are concatenated.	A := 'AB34' B := A || 'FFFF' means that B is assigned the value 'AB34FFFF'

IF
THEN
ELSE
ENDIF
	This textual description is used to specify decision logic, using the following syntax:
	IF T
	THEN
	Logic 1
	ELSE
	Logic 2
	ENDIF
	where T is a statement resulting in true or false.
		IF [IsNotEmptyList(Data Envelopes To Write Yet)]
		THEN
		P2 := '80'
		ELSE
		P2 := '00'
		ENDIF

Notation	Meaning	Example

FOR
EXIT loop
	This textual description is used to specify repetition control logic, using the following syntax:
	FOR every x in list
	{
	IF T
	THEN
	Action
	EXIT loop
	ENDIF
	}
		FOR every T in Extended SDA Tag List
		{
		IF[IsNotPresent(T) OR IsEmpty(T)]
		THEN
		Data objects referenced in Extended SDA Tag List present := FALSE
		EXIT loop
		ENDIF
		}

CALL	Functionality of a certain complexity and appearing more than once is implemented by means of a procedure. In the textual description, a procedure is referenced by means of the key word CALL.	CALL Process Restrictions ( )
A mod n	The reduction of the integer A modulo the integer n, that is, the unique integer r, $0 \leq r < n$, for which there exists an integer d such that $A = dn + r$	54 mod 16 = 6
A div n	The integer division of A by n, that is, the unique integer d for which there exists an integer r, $0 \leq r < n$, such that $A = dn + r$	54 div 16 = 3
$X \oplus Y$	The bit-wise exclusive-OR of the data blocks X and Y.	11001100b $\oplus$ 10101010b = 01100110b

1.6 Version

Kernel and Card both maintain a version number, coded as 'Version' in Kernel Qualifier and in Card Qualifier respectively. The lowest version number of the two determines the functionality that can be used for the transaction.

Two versions are defined, VERSION 1 and VERSION 2, with details on the corresponding functionality provided in Table 1.4.

A Kernel implementing this version of the specification must set 'Version' to VERSION 2 in the Kernel Qualifier and must support Cards that provide 'Version' with value VERSION 1 or VERSION 2 in the Card Qualifier.

Table 1.4—Versions and Corresponding Functionality

Version	Functionality
VERSION 1	The generation of the Issuer Application Data MAC does not use the Issuer Application Data as input. The Issuer Application Data MAC is inserted in the Issuer Application Data before using the Issuer Application Data as input for the generation of the Enhanced Data Authentication MAC.
VERSION 2	The generation of the Issuer Application Data MAC uses the Issuer Application Data as input. The Issuer Application Data MAC is used as input for the generation of the Enhanced Data Authentication MAC. The Issuer Application Data MAC is (optionally) included in the Issuer Application Data by the Kernel before including the Issuer Application Data in the Data Record.

[EMV Book E] section 3.2 describes the generation of the Issuer Application Data MAC (IAD MAC) for VERSION 2; for VERSION 1, the Kernel must provide IAD MAC input data as specified in this document and compute IAD MAC with the same process as VERSION 2.

2 General Architecture

2.1 Introduction

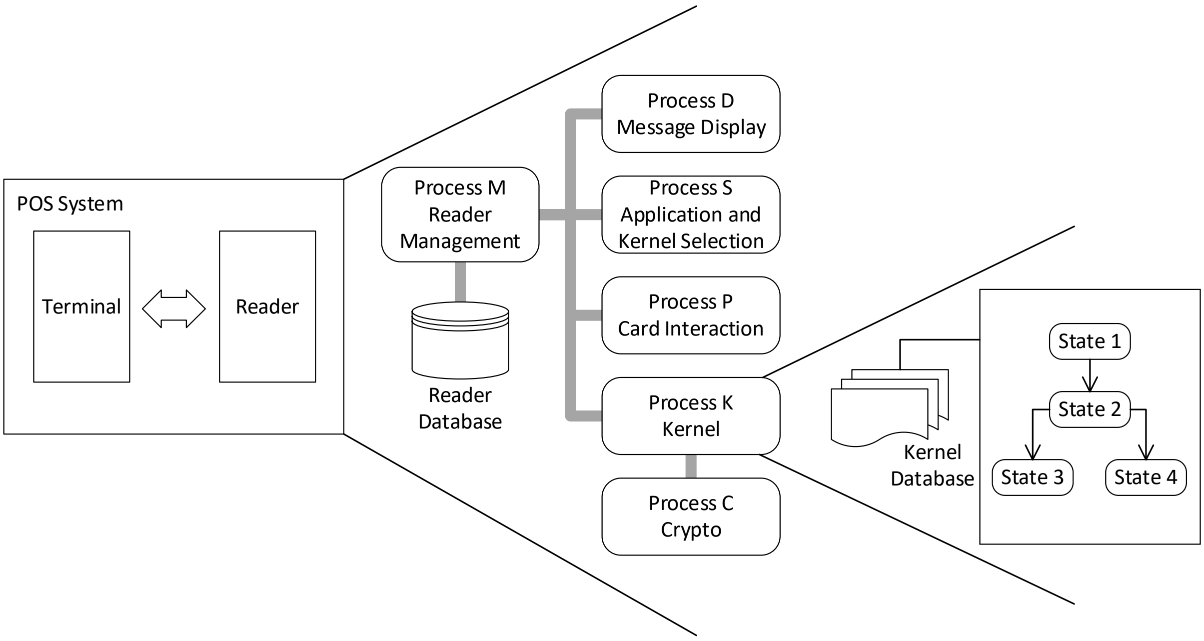
As described in [EMV Book A], the general architecture of a POS System consists of a Terminal and a Reader, where the terms Terminal and Reader refer to a separation in responsibility and functionality between two logical entities.

Figure 2.1 shows how the Reader functionality is allocated to different processes: Process M(ain), Process D(isplay), Process S(elect), Process P(CD), Process K(ernel) and Process C(rypto).

Communication between different processes happens through Signals stored on Queues. A Signal is an asynchronous event that is placed on a Queue waiting to be processed. A Signal can convey data as parameters, and the data provided in this way is used in the processing of the Signal.

Zooming in further on Process K, Figure 2.1 illustrates the two components of the Kernel: the Kernel software, modelled as a state machine, and the Kernel database, consisting of a number of separate datasets.

Figure 2.1—General Architecture



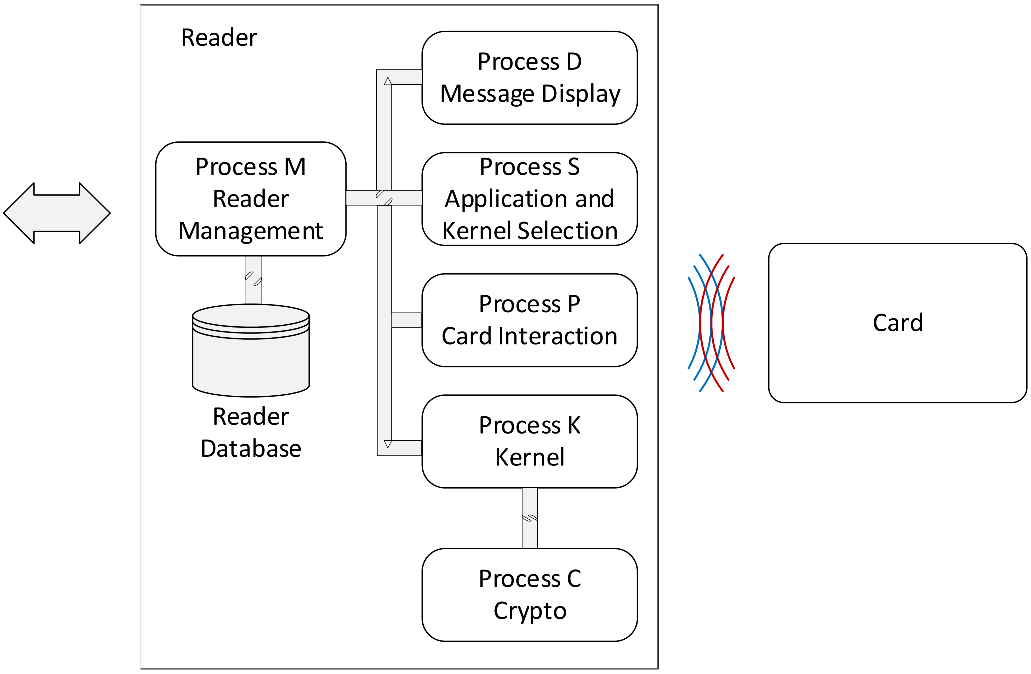
There is no requirement to create devices that use the architecture and the partitioning as laid out in this document, as equally there is no requirement in [EMV Book A] on the partitioning.

The only requirements in this document apply to Process K and Process C, and these requirements define the externally-observable behaviour, independent of the internal organisation of the Reader.

2.2 Reader Processes

In Figure 2.2, the Reader is modelled as a set of Processes and each Process runs independently of the other Processes. The role of the Reader database is explained in section 2.3.

Figure 2.2—Reader Logical Architecture



The different processes are listed in Table 2.1.

Table 2.1—Reader Processes

Process	Responsibility
Process P(CD)	Management of the contactless interface
Process D(isplay)	Management of the user interface
Process S(election)	Selection of the Card application and Kernel
Process K(ernel)	Interaction with the Card once the application has been selected, covering the transaction flow specific to Kernel 8
Process M(ain)	Overall control and sequencing of the different processes. The configuration and activation of the Kernel and the processing of its outcome are also part of this role.
Process C(rypto)	Processing of cryptographic material

2.2.1 Process P

Process P implements the functionality described in [EMV CL L1] and [ISO/IEC 7816-4] and manages the access to the Card.

Process K communicates with Process P through the CMD, RA and L1RSP Signals as shown in Table 2.2.

Table 2.2—Process P Signals

Signal In	Signal Out	Comment
CMD	RA(R-APDU)	If there is no L1 error, the RA Signal contains the R-APDU sent back in response to a C-APDU.
	L1RSP(code)	If there is an L1 error, L1RSP is returned with code as one of the following: • Error – Timeout: An L1 timeout has occurred • Error – Protocol: An L1 protocol error has occurred • Error – Transmission: Any other error

Process P sends the C-APDU included in the CMD Signal to the Card and responds with either:

- An RA Signal containing the R-APDU and status bytes returned by the Card, or
- An L1RSP Signal including an L1 event such as a timeout, transmission error, or protocol error.

2.2.2 Process D

Process D manages the User Interface Requests as defined in [EMV Book A] and displays a message and/or a status.

A MSG Signal is used as a carrier of User Interface Request Data (UIRD). Process D may receive MSG Signals from any other Process. The MSG Signal is not acknowledged.

For more information on UIRD, refer to [EMV Book A] section 7.1. For the definition of the UIRDs used in this specification, see Annex A.

For displaying messages and/or indicating status, Process D needs the following configuration data:

- Default language
- The currency symbol to display for each currency code and the number of minor units for that currency code
- A number of messages in the default language and potentially other languages

The status identifiers and message identifiers are defined in [EMV Book A] section 9.2 and section 9.4 respectively.

2.2.3 Process S

Process S manages the application and Kernel selection. It runs constantly, and it is waiting for an ACT Signal from Process M. After processing the ACT Signal, it returns the selected Combination of application and Kernel (AID and Kernel ID) and the File Control Information Template and status bytes returned by the Card in response to the final SELECT command in an OUT Signal.

Refer to [EMV Book B] for the specification of an implementation of Process S that supports a multi-kernel architecture.

2.2.4 Process K

The Reader may support multiple Kernels but only one Kernel will execute at a time. The Kernel that is activated depends on the information returned by Process S, which may in turn depend on data retrieved from the Card.

For each transaction, Process K is configured with a Kernel-specific dataset. The values in the dataset depend on the AID and the Transaction Type. More information on the initialisation of the Kernel-specific dataset is provided in section 2.3.

Once the Kernel is selected and configured, it executes as Process K. Process K manages the interaction with the Card application beyond application selection, using the services of Process P as an intermediary. Upon completion, Process K sends its results to Process M in an OUT Signal and then terminates. For the remainder of the document, it is assumed that Kernel 8 is selected.

From the viewpoint of the Reader and depending on the Configuration Options, Kernel 8 can provide two services:

- Through its interaction with the Card, it creates a transaction record for authorisation and/or clearing.
- It can interact with the Terminal directly through the Data Exchange mechanism.

Seen from the Terminal and depending on the Configuration Options, Kernel 8 allows reading and writing data from and to the Card.

The different services are listed in Table 2.3, with the corresponding Signal to call the service indicated in the right column. Only Process M and the Terminal request these services from Process K.

Table 2.3—Services from Process K

Service	Corresponding Signal
Return an authorisation or clearing record.	ACT(Data) “Data” includes File Control Information Template (received from the Card in the response to the SELECT command) and other data objects with ACT in the update condition, such as: • Account Type • Amount, Authorised (Numeric) • Amount, Other (Numeric) • Data Envelopes To Write • Proceed To First Write Flag • Tags To Read • Transaction Currency Code • Transaction Currency Exponent • Transaction Date • Transaction Time • Transaction Type
Request data from the Kernel database or from the Card. Write data to the Kernel database or to the Card.	DET(Data)

Process K responds to the incoming service request with an outgoing Signal as described in Table 2.4.

Table 2.4—Responses from Process K

<table>
<tr><th>Signal In</th><th>Signal Out</th><th>Comment</th></tr>
<tr><td>ACT</td><td>OUT</td><td>The OUT Signal includes:
Outcome Parameter Set
Data Record – if any
Discretionary Data
User Interface Request Data 1 – if any
User Interface Request Data 2 – if any</td></tr>
<tr><td>DET</td><td>DEK or n/a</td><td rowspan="2">The DEK Signal can be used to request additional data to be provided in a subsequent DET Signal, as well as to provide data that was requested via a configuration setting or a previous DET Signal.
The DEK Signal contains:
The Data Needed data object, which is the list of tags of data items that the Kernel needs from the Terminal
The Data To Send data object, which is the list of data values that the Terminal has requested</td></tr>
<tr><td>n/a</td><td>DEK</td></tr>
</table>

Within a transaction, the Terminal can only send one or more DET Signals after it received a DEK Signal indicating that Process K is active. The DEK Signal is sent only if the Kernel has data for the Terminal or needs data from the Terminal.

The DEK and DET Signals are exchanged as part of the Data Exchange mechanism and are only used if DE/DS implementation option, as defined in section 3.1, is implemented.

Process K implements a Timer that runs as a daemon parallel to Process K. If not stopped, the Timer places a TIMEOUT Signal on the queue of Process K at a pre-set time interval after the Timer is started.

2.2.5 Process M

Process M is responsible for coordinating the other processes to perform a transaction. The overall process is illustrated as follows:

- Process M receives the ACT Signal from the Terminal.
- Process M starts Process P to start polling for Cards as described in [EMV CL L1].
- Process M requests Process D to display the READY message.

- When Process P indicates to Process M that a Card is detected, Process M activates Process S. When Process S completes successfully, it responds with an OUT Signal with the selected Combination {AID – Kernel ID}, the File Control Information Template of the selected Card application, and the status bytes returned by the Card.
- Based on this information, Process M then configures Process K for the specific Transaction Type and AID, using a Kernel-specific dataset, and sends it an ACT Signal containing transactional data (see Table 2.3). When Process K has completed the transaction, it returns an OUT Signal to Process M, including the Outcome Parameter Set, Discretionary Data, Data Record (if any), and optionally one or two UIRDs.
- Process M analyses the 'Status' in Outcome Parameter Set and executes the instructions encoded in the other fields of the Outcome Parameter Set. As required, Process M instructs Process P to perform the removal sequence. Alternatively, it may instruct Process S to select the next application on the Card.
- Process M passes a subset of the Outcome Parameter Set, the Data Record, and the Discretionary Data to the Terminal.
- If the transaction is processed online, the Reader receives a MSG Signal from the Terminal to indicate whether the transaction was approved or declined.

2.2.6 Process C

Process C is the cryptographic process and only Process K communicates with Process C.

Communication between Process K and Process C is described in terms of Signals. The specification presumes that Signals between Process C and Process K are managed on a dedicated Queue that is exclusively reserved for their interactions.

Process C is designed to run in parallel with Process K, offloading the processing of cryptographic data to simplify Process K and speed up the processing transaction flow.

It is an implementation decision as to how it is implemented; if a reader has very fast cryptographic processing and the implementer prefers not to implement parallel processing, then its functions may be executed sequentially with respect to Process K but it is presented in this specification as a parallel process.

Process C handles the following tasks:

- Secure channel and certificate algorithm suite negotiation
- Processing of the key related data in the GET PROCESSING OPTIONS response
- Decryption of privacy protected (encrypted) record or READ DATA data
- Local authentication including validation of the Card and issuer certificates and the blinding factor
- Generation of the Issuer Application Data MAC and validation of the Enhanced Data Authentication MAC

- Validation of MACs for READ DATA and WRITE DATA
- Encryption of data for WRITE DATA for transmission to the Card

It is invoked by Process K by means of Signals. The Signals that Process K sends to Process C are listed in Table 2.5. Process C will process Signals in the order they are sent to it by Process K. In response to these Signals, Process C will return a message for each message sent, subject to the exceptions shown as described in Table 2.5.

Table 2.5—Process C Signals

In response to	Process C will send	
	Normal response	**Exception**
INITIALISE PROCESS C	INIT PROCESS C OK	INIT PROCESS C FAIL
GPO PRIVACY	None	DECRYPTION FAILED
RECORD RECEIVED	None[1] or RECORD DECRYPTED	DECRYPTION FAILED
PROCESS ISSUER CERTIFICATE	ISSUER CERTIFICATE OK	ISSUER CERTIFICATE FAIL
PROCESS ICC CERTIFICATE	ICC CERTIFICATE OK	ICC CERTIFICATE FAIL
PROCESS EDA MAC	EDA MAC OK	EDA MAC FAIL
PROCESS BLINDING FACTOR	BLINDING FACTOR OK	BLINDING FACTOR FAIL
VALIDATE MESSAGE	MESSAGE OK	MESSAGE FAIL
ENCRYPT MESSAGE	ENCRYPTED MESSAGE	ENCRYPTION FAILED
PROCESS IAD MAC	IAD MAC OK	IAD MAC FAIL

(1) If a RECORD RECEIVED Signal contains a plaintext record, Process C will not return a message. Process K sends plaintext records to Process C as it is Process C that builds the Static Data To Be Authenticated.

2.3 The Reader Database

The Reader maintains a database that is divided into different datasets held in persistent memory. An overview of the different persistent datasets is given in Figure 2.3, with additional details in Table 2.6.

Figure 2.3—Reader Database – Persistent Datasets

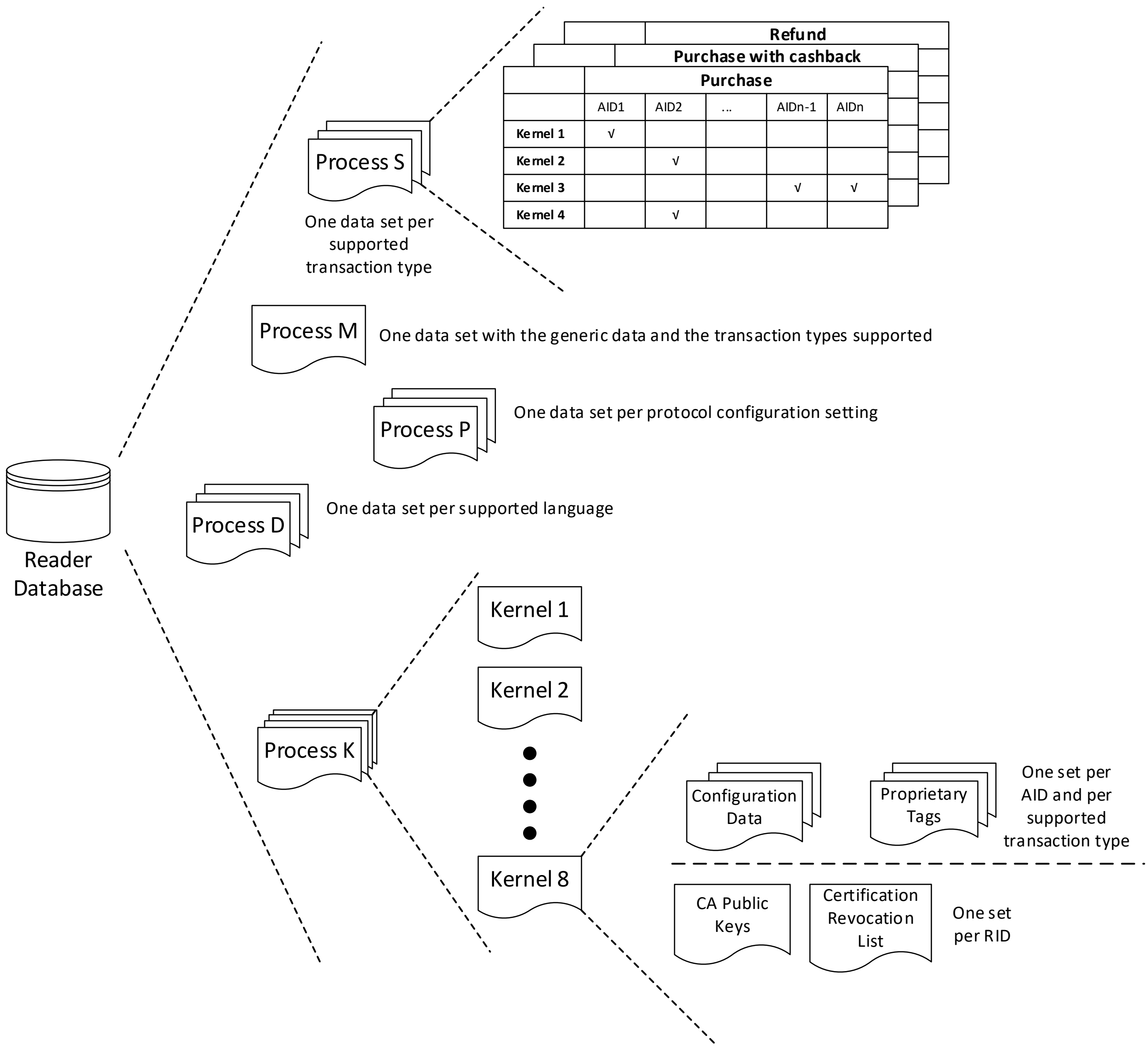


Table 2.6—Reader Database

Process	Data Sets
Process M	One dataset, including generic data and the different transaction types supported. Examples of generic data are Interface Device Serial Number, Terminal Country Code, Transaction Currency Code, and Transaction Currency Exponent. Examples of transaction types are purchase, purchase with cashback, and refund.
Process P	One or more datasets, one for each protocol configuration setting. Each dataset contains (part of) the configuration settings as defined in [EMV CL L1] Annex A.
Process D	Multiple datasets for Process D, one for each supported language. Each dataset contains the messages behind the message identifiers.
Process S	Multiple datasets for Process S, one dataset per Transaction Type. Each dataset contains a list of Combinations {AID – Kernel ID}.
Process K	Multiple Kernel-specific datasets for Process K. Each Kernel-specific dataset includes different subsets. For Kernel 8, see Table 2.7.

Table 2.7—Persistent Dataset for Kernel 8

Subset	Purpose
Configuration Data	Contains the TLV-coded persistent data objects relevant to a specific AID (used by Process S to identify the application during application selection) and Transaction Type. The values of the TLV-coded data objects do not vary per transaction.
Proprietary Tags	Data objects with proprietary tags (i.e. data objects with tags not listed in Table A.37). For each proprietary tag, access conditions, format, and length must be defined.
CA Public Key Database	Information linked to the Certification Authority Public Keys, including the index, modulus, and exponent. Certification Authority Public Keys can be shared between AIDs that have the same RID and sharing can be done across Kernels. The Reader must be able to store the information for at least six RSA keys per RID and ten ECC keys per RID.
Certification Revocation List	A list of Issuer Public Key Certificates that payment systems have revoked for each RID supported by the Kernel. As for the CA Public Key Database, entries in the Certification Revocation List may be shared between Kernels where Kernels support the same RID.

The Reader organises and manages the Reader database in an implementation specific manner. The only requirement is that the Kernel must have access to the datasets defined in Table 2.7 at start-up. The Reader provides the Configuration Data and Proprietary Tags per AID and Transaction Type. It is, however, not a requirement that the Reader stores a complete set of Configuration Data for each combination AID - Transaction Type. Generic configuration data objects may be stored only once, while other configuration data objects may be stored per RID.

3 Overview of Kernel Functionality

3.1 Implementation Options

Kernel 8 supports the Implementation Option listed in Table 3.1.

Table 3.1—Kernel Implementation Options

Implementation Option	Description
Data Exchange & Data Storage	This Implementation Option gives the implementer the choice to build Kernel 8 with or without support for Data Exchange and Data Storage. This Implementation Option is further on referenced as DE/DS.

This Implementation Option gives rise to the following two possible Kernel implementations:

- DE/DS implemented
- DE/DS not implemented

3.2 Kernel Data Sources

When the Kernel process starts, the Kernel database, as introduced in section 2.3, is already initiated with the persistent dataset of Kernel 8 for a specific AID (or RID) and Transaction Type. This includes the Configuration Data, Proprietary Tags, CA Public Key Database, and the Certification Revocation List (see Table 2.7).

For the purpose of data management during transaction processing, the Kernel also instantiates an empty key-value based data store, nominally referred to as the TLV Database. Each entry in the database has a unique identifier as the key and a value. The TLV Database is stored in volatile memory and cleared after the Kernel exits. At the start of the transaction, the Kernel copies the Configuration Data and Proprietary Tags (if any) into the TLV Database.

In addition to the Kernel database, the Kernel receives transaction data items in the ACT Signal. These data items originate from the Terminal and from the OUT Signal of Process S. These data items with their transient values are stored in the TLV Database as well.

During transaction processing, the Kernel may receive events from Process M, the Card, and the Terminal. This input, together with the Kernel's progression through the transaction processing, causes further updates to the TLV Database.

While performing a transaction, the Kernel ensures that updates to the TLV Database are done only by the authorised 'source' (origin) of the data item. For this purpose, data items are put in different categories and the category determines the Signal – and therefore source – that can update data objects within a category. The different categories and corresponding Signals are illustrated in Table 3.2.

Table 3.2—Kernel TLV Database Categories

Data Category	Signal
Terminal sourced data object – configuration data	n/a
Transaction data object (terminal and card sourced data objects – see Table 2.3)	DET [1], ACT
Kernel defined value or internal data object	n/a Value can only be changed as part of Kernel processing
Card sourced data object	RA [2]

[1] Only implemented for the DE/DS Implementation Option

[2] The File Control Information Template is received in an ACT Signal but is treated as an RA as that is how it was delivered to Process S.

3.3 Transaction Flow

Kernel 8 executes the following steps:

- The Kernel sends an ephemeral ECC public key to the Card in the GET PROCESSING OPTIONS command data. Using its private key and a blinding factor, the Card generates a shared secret and from that a set of session keys. In the GET PROCESSING OPTIONS response, the Card sends its blinded public key and the encrypted blinding factor so that the Kernel can compute the same shared secret and session keys. The blinding factor will permit the Kernel to authenticate the Card in conjunction with the Card certificates.
- It reads the data records of the Card (using READ RECORD commands). Any record data returned by the Card that uniquely identifies it (for example containing the Application PAN) is encrypted by the Card using the Session Key for Confidentiality with the AES-CTR encryption operation as described in [EMV Book E] section 8.6.2.
- It performs Terminal Risk Management and Terminal Action Analysis.
- It requests an Application Cryptogram from the Card by issuing a GENERATE AC command. The Kernel offers to the Card in the GENERATE AC command data a list of the CVMs that it supports for the transaction and the Card picks one from that list. The Card informs the Kernel of its choice in the response to the GENERATE AC command.
- It generates the Issuer Application Data MAC (AES-CMAC+, an AES-CMAC based MAC function calculated over static card data and transaction related data using the Session Key for Integrity as described in [EMV Book E] section 8.6.5) and validates the Enhanced Data Authentication MAC (an AES-CMAC calculated over the Issuer Application Data MAC and Application Cryptogram using the same session key).
- The Card may use its instance of the Issuer Application Data MAC as input to the Application Cryptogram generation. In that case the Kernel instance of the Issuer Application Data MAC has to be transferred to the issuer. This may be done by including the Issuer Application Data MAC in the Issuer Application Data. The Card indicates in 'Copy IAD MAC in IAD' in Application Interchange Profile if and how the Kernel must copy the Kernel instance of the Issuer Application Data MAC in the Issuer Application Data.
- If both Card and Kernel are configured to support local authentication, then the Kernel validates the Card and issuer certificates and the blinding factor.

3.4 Data Exchange

3.4.1 Introduction

Terminal and Kernel can communicate through the Data Exchange mechanism if the DE/DS Implementation Option is implemented.

The Kernel can send tagged data to and request data from the Terminal through the DEK Signal.

The Terminal can control the Kernel through the DET Signal by virtue of its ability to:

- Update the current transaction database of the Kernel
- Request tagged data from the Kernel or from the Card
- Manage the transaction flow pace by means of the Proceed To First Write Flag.

3.4.2 Sending Data to the Terminal

As part of its configuration or through an ACT or DET Signal, the Kernel has a data object (Tags To Read) containing the tags of the data objects to be sent to the Terminal. If a tag refers to Card data, this data is retrieved through READ RECORD commands – as part of reading the records listed in the Application File Locator – or through the READ DATA commands. The Kernel has a pre-defined list of data objects with pre-defined tags ('9F8111' to '9F811A') that are read using READ DATA. All other data objects are read using READ RECORD commands. No files or records other than those being listed in the Application File Locator are read.

When the Kernel has completed the (currently outstanding) requests from the Terminal, it sends the data to the Terminal via a DEK Signal.

The information in the DEK Signal may trigger the Terminal to send another list of data to read by means of Tags To Read in a DET Signal. This list is then appended to the original list and may result in another set of READ DATA commands if the request includes pre-defined tags ('9F8111' to '9F811A').

The Kernel uses a buffer, called Tags To Read Yet, to accumulate the different read requests included in Tags To Read.

Data To Send is another buffer, accumulating the multiple data that the Kernel has for the Terminal. It is populated with TLV data retrieved in response to Tags To Read Yet processing.

The process continues until all records have been read and there are no more data objects in the list that need to be read using a READ DATA command.

3.4.3 Requesting Data from the Terminal

The Kernel uses Data Needed to request the following data objects from the Terminal:

- Tags To Read
- Data Envelopes To Write
- Proceed To First Write Flag

If Tags To Read or Data Envelopes To Write is an empty list at start-up of the Kernel (i.e. it is configured with an empty list in Configuration Data or an empty list is received in the ACT Signal), then the Kernel adds the tag of Tags To Read or Data Envelopes To Write to Data Needed and sends Data Needed with a DEK Signal to the Terminal.

In return the Terminal may send a DET Signal with the requested data objects.

The Terminal may send multiple DET Signals during the transaction. Each DET Signal may also contain a Data Envelopes To Write data object. The Kernel manages these DET Signals through the Data Envelopes To Write Yet buffer. This buffer is used to accumulate the TLV data objects included in Data Envelopes To Write lists.

All data are read from the Card previous to any data writing process. To ensure that the reading process is completed and that the Terminal has received all required data, the Kernel checks whether it can move to the writing stage by means of the Proceed To First Write Flag data object:

- When Proceed To First Write Flag is absent, the Kernel can move to the writing phase of the transaction.
- When Proceed To First Write Flag has length zero, the Kernel requests a value for the Proceed To First Write Flag from the Terminal. It waits until the Terminal provides this value before moving to the writing phase.
- When Proceed To First Write Flag has value zero, the Kernel waits until the Terminal provides a value other than zero before moving to the writing phase.
- When Proceed To First Write Flag has a value other than zero, the Kernel can move to the writing phase of the transaction.

3.5 Data Storage

Data storage is an extension of the regular transaction flow such that the Card can be used as a scratch pad or mini data store with simple write and read functionality.

Data storage relies on the Data Exchange mechanism as described in section 3.4 and is only supported if the DE/DS Implementation Option is implemented.

Data storage uses dedicated commands (READ DATA and WRITE DATA) for explicit reading and writing of data. It introduces a range of tags ('9F8111' to '9F811A') for the reading and writing of non-payment data, in such a way that they can be included in Tags To Read and Data Envelopes To Write (see section 3.4). The whole range is freely readable using the READ DATA command. Writing is performed using a WRITE DATA command.

The length of the data is variable with a maximum of 243 bytes.

The Data Envelopes To Write list is TLV coded, containing Tag, Length, and Value of the data to write. The list may be part of the Configuration Data, or it may be communicated to the Kernel during the transaction using the Data Exchange mechanism, via a DET Signal. If Data Envelopes To Write is an empty list at start-up of the Kernel, then a DEK Signal is sent to the Terminal to request the Terminal to communicate the Data Envelopes To Write to the Kernel via a DET Signal.

After the GENERATE AC command, the Kernel may send one or more WRITE DATA commands to the Card. Each command contains one data object from the Data Envelopes To Write list. The WRITE DATA commands are sent in the order as they appear in the Data Envelopes To Write list.

Data storage protects data being sent to the Card and received from the Card with the same privacy mechanism as the one used to protect the Card data returned to the Kernel in response to READ RECORD commands.

When a READ DATA command is used, the data is protected by a MAC computed with one of the session keys so that the Kernel may have confidence that it was received unaltered. When data is written by the Kernel using WRITE DATA, the Card returns a MAC computed over the recovered plaintext to provide confidence that it was received by the Card unaltered.

3.6 Relay Resistance Protocol

A relay attack takes place when a fraudulent terminal is used to mislead an unsuspecting cardholder into transacting, where the actual transaction is relayed via a fraudulent Card (or simulator) to the authentic terminal of an unsuspecting merchant. It may also happen when a fraudulent reader is used without the cardholder being aware of the transaction.

RRP operates as follows:

1. A bit in Application Interchange Profile is used to inform the Kernel that the Card supports RRP. A bit in Kernel Configuration is used to enable RRP in the Kernel.
2. The Kernel invokes RRP if both Card and Kernel support the procedure. In this case, it sends a timed C-APDU (EXCHANGE RELAY RESISTANCE DATA) to the Card with an unpredictable number (Terminal Relay Resistance Entropy). The Card responds with an unpredictable number (Device Relay Resistance Entropy) and timing estimates (Min Time For Processing Relay Resistance APDU, Max Time For Processing Relay Resistance APDU, and Device Estimated Transmission Time For Relay Resistance R-APDU).
3. If the timings being determined by the Kernel exceed the maximum limit computed, the Kernel will perform a second trial (in case there was a communication error or in case other processing on the Card interrupted the EXCHANGE RELAY RESISTANCE DATA command processing). The Kernel will execute up to two retries.
4. Terminal Verification Results are used to permit the Kernel to be configured through the Terminal Action Codes to decline or send transactions online in the event that timings are outside the limits computed.
5. RRP relies on local authentication for offline transactions and on the Issuer Application Data MAC combined with online card authentication for online transactions.
6. The Terminal Relay Resistance Entropy is the same as the Unpredictable Number. In the event of retries, new values for the Unpredictable Number are computed.
7. The Kernel considers a transaction to be valid if the processing time determined from the measured time is within the window stated by the Card (i.e. Max Time For Processing Relay Resistance APDU and Min Time For Processing Relay Resistance APDU). In addition some tolerance is given by the reader in the form of a grace period below and above the window defined by the Card (Minimum Relay Resistance Grace Period and Maximum Relay Resistance Grace Period).

   The Reader has an accuracy threshold (Relay Resistance Accuracy Threshold) that indicates whether the measured time is greater than a reader permitted limit. Another accuracy threshold (Relay Resistance Transmission Time Mismatch Threshold) considers the mismatch of the Card communication time.

3.7 Privacy Protection

Privacy Protection is a mechanism that ensures that eavesdropping attacks on the communications between Card and Reader cannot tell the identity of the Card that is used to perform the transaction. The mechanism ensures that it is impossible to tell from the payment application data if two transactions performed at the same terminal were performed by the same or different Cards or payment applications.

The privacy protection is based on a secure channel between the Card and the Reader as described in [EMV Book E] Chapter 2.

3.8 CVM Processing

The Kernel offers to the Card in the GENERATE AC command data a list of the CVMs that it is willing to see used for the transaction and the Card picks one from that list. The Card informs the Kernel of its choice in the response to the GENERATE AC command.

The CVMs that can be used are:

- No CVM
- Signature
- CDCVM
- Online PIN

The Kernel uses a CVM limit to determine which CVMs are offered to the Card. If the transaction is above the CVM limit then 'No CVM' should not be offered.

The Reader CVMs are coded in CVM Capability – CVM Required and CVM Capability – No CVM Required which are part of Configuration Data. It is the Terminal Risk Management Data that is used to communicate the available CVMs from the Kernel to the Card.

Terminal Risk Management Data also carries three other CVM related bits. The first of these, 'CVM Limit exceeded' is used as an explicit indication to the Card that the transaction amount exceeds the limit above which a CVM is required by the merchant. The other two bits ('CDCVM bypass requested' and 'SCA exempt') are related to the business rules relevant to this installation. For example, if CDCVM bypass is requested by a transit terminal or whether the transaction is not subject to SCA regulations, thus maximising frictionless and user-friendly payment experiences.

Based on the above, the Card then chooses the CVM to be used for this transaction. The Card's decision is indicated in the Cardholder Verification Decision data object included in the GENERATE AC response. This data object reflects the Card's choice of CVM. The card may indicate CV FAILED if there is no CVM available from the Kernel that the Card is willing to use for this transaction.

Because the choice of CVM by the Card also affects the value of the Terminal Verification Results, the Card may change certain bits in the Terminal Verification Results. If for example the Card chooses Online PIN, the Terminal Verification Results need to be set to indicate that Online PIN will be entered. This means that the Card needs to use a TVR value different from the value it received in the GENERATE AC command data. Therefore, the Card may optionally include in the GENERATE AC response data a Card TVR corresponding to the one it used for generating the Application Cryptogram, and the Kernel needs to reflect this change in the data sent to the acquirer.

A mask value is applied to avoid that the Card could change bits in the Terminal Verification Results that reflect the Kernel's decision making. This mask is referred to as the Kernel Reserved TVR Mask and the bits set to 1b cannot be altered by the Card. The Kernel sets the Terminal Verification Results bits that the Card can change to the value of the corresponding bits in the Card TVR.

3.9 Kernel Configuration Options

The different Configuration Options supported by Kernel 8 are listed in Table 3.3, as well as the method to activate a particular option.

Table 3.3—Kernel Configuration Options

Configuration Option	Description	Activation
RRP	The Kernel supports RRP	Through the setting of 'Relay resistance protocol enabled' in Kernel Configuration
Report local authentication failed in TVR	The Kernel sets the 'Local authentication failed' bit in the Terminal Verification Results if Card certificate, issuer certificate or blinding factor validation fails. This bit is set after the GENERATE AC command and used to determine the Kernel Decision. If the 'Report local authentication failed in TVR' in Kernel Configuration is not set, then the 'Local authentication failed' bit in the Terminal Verification Results will be cleared before storing the Terminal Verification Results in the Data Record.	Through the setting of 'Report local authentication failed in TVR' in Kernel Configuration
RSA certificates	The Kernel activates validation of RSA certificates returned by the Card for local authentication.	Through the setting of 'RSA certificates enabled' in Kernel Configuration
Local authentication	The Kernel activates validation of Card certificates, issuer certificates and the blinding factor.	Through the setting of 'Local authentication' in Security Capability

All the above Configuration Options for the Kernel are set at the level of the AID and the Transaction Type and are part of the Configuration Data in the persistent dataset of Kernel 8.

4 Data Organisation

4.1 TLV Database

4.1.1 Principles

The Kernel maintains a TLV Database to store known data objects. A data object is known if its tag is listed in the data dictionary (Table A.37). Other data objects with proprietary tags not listed in Table A.37 become known if they are included in Proprietary Tags.

This TLV Database is instantiated by the Kernel as described in section 3.2. It is updated during the processing of the transaction from a number of sources: at start-up from the Reader, with data from the Card, with data from the Terminal, and with data that results from the Kernel's own processing.

A data object in the TLV Database has T as the key and notionally L as the length and V as the value.

A data object is uniquely identified by its tag in the TLV Database (that is, no two data objects of the same tag can exist in the same TLV Database); it is considered to be present if its tag appears in the TLV Database.

Constructed TLV-coded data objects received from Card and Terminal are not stored in their entirety in the TLV Database. Only the primitive TLV-coded data objects present in the value field of the constructed TLV-coded data object are stored in the TLV Database.

In this document, data objects in the TLV Database are referred to by their name as defined in the data dictionary; in this context, the name is synonymous with the tag as the key for the data object. For example, for the following data object:

Name:	Terminal Verification Results
Tag:	'95'
Length:	5
Value:	'FFFFFFFFFF'

The name of the TLV-coded data object is used to access the data object and its value, and the following notation updates the value of the data object in the TLV Database to zero:

Terminal Verification Results := '0000000000'

4.1.2 Access Conditions

Data objects in the TLV Database are assigned access conditions as described in Table 4.1.

Table 4.1—Access Conditions

Access Condition	Description
ACT/DET	These data objects are data objects related to the transaction, being sent to the Kernel by the Terminal with the ACT and DET Signals. They may also be present in the TLV Database when the Kernel is instantiated.
RA	These data objects are data objects related to the transaction, being sent to the Kernel by the Card with the RA Signal.
K	All data objects in the TLV Database can be updated by the Kernel. Every data object has the K (Kernel) access condition assigned.

All data objects can be requested by the Card (via a DOL) and by the Terminal (via Tags To Read) if DE/DS is implemented.

4.1.3 Services

The Kernel uses the following services from the TLV Database:

Boolean IsPresent(T)

Returns TRUE if the TLV Database includes a data object with tag T.

Boolean IsNotPresent(T)

Returns TRUE if the TLV Database does not include a data object with tag T.

Boolean IsNotEmpty(T)

Returns TRUE if all of the following are true:

- IsPresent(T)
- GetLength(T) > 0

Boolean IsEmpty(T)

Returns TRUE if all of the following are true:

- IsPresent(T)
- GetLength(T) = 0

Initialise(T)

Initialises the data object with tag T with a zero length. After initialisation, IsPresent(T) returns TRUE.

DataObject GetTLV(T)

Retrieves the TLV-coded data object with tag T from the TLV Database:

IF [IsNotPresent(T)]
THEN
    Return NULL
ELSE IF [IsEmpty(T)]
THEN
    Return T || '00'
ELSE
    Return T || L || V
        Note: L is coded on the minimum number of bytes (that is, on 1 byte if < 128, on 2 bytes if in the range 128 to 255, and so on). See the BER-TLV coding rules in section 4.7.1.
END IF

Length GetLength(T)

Retrieves from the TLV Database the length of the value in bytes of the data object with tag T. Returns NULL if IsNotPresent(T) returns TRUE.

Value GetValue(T)

Retrieves from the TLV Database the value in bytes of the data object with tag T. Returns NULL if IsEmpty(T) OR IsNotPresent(T) returns TRUE.

The Kernel also uses the following general services to evaluate and manipulate data objects:

Boolean IsKnown(T)

Returns TRUE if tag T is defined in Table A.37 or if tag T is included in Proprietary Tags.

T TagOf(DataObjectName)

Returns the tag of the data object with name DataObjectName.

Boolean ParseAndStoreCardResponse(TLV String)

```
IF [TLV String is a single TLV data object]
THEN
    Parse TLV String according to the Basic Encoding Rules in [ISO/IEC 8825-1] and
    section 4.7.1.
    Iterate the parsing process if TLV String represents a nested constructed data object.
    The result of this parsing step is a list of primitive TLVs, TLV List.
    IF [parsing error occurs]
    THEN
        Return FALSE
    ENDIF
ELSE
    Return FALSE
ENDIF
```

```
FOR every primitive TLV in TLV List
{
    IF [IsKnown(T)]
    THEN
        IF  [(IsNotPresent(T) OR IsEmpty(T) OR (V = GetValue(T)))
            AND update conditions of T include RA Signal
            AND L is within the range specified by Length field of the data object
            with tag T in the data dictionary]
        THEN
            Store LV in the TLV Database for tag T
        ELSE
            Return FALSE
        ENDIF
    ENDIF
}
Return TRUE
```

UpdateWithDetData(Terminal Sent Data) [3]

Copies all incoming data (Terminal Sent Data) to the TLV Database if update conditions allow.

Individual data objects contained within lists in Terminal Sent Data are not stored in the database.

```
FOR every TLV in Terminal Sent Data
{
    IF      [IsKnown(T)
            AND update conditions of T include DET Signal]
    THEN
            Store LV in the TLV Database for tag T
    ENDIF
}
IF  [Terminal Sent Data includes Tags To Read]
THEN
    AddListToList(Tags To Read, Tags To Read Yet)
ENDIF
IF  [Terminal Sent Data includes Data Envelopes To Write]
THEN
    AddListToList(Data Envelopes To Write, Data Envelopes To Write Yet)
ENDIF
```

[3] Only implemented for the DE/DS Implementation Option

4.1.4 DOL Handling

Data objects moved from the Kernel to the Card are identified by a DOL sent to the Kernel by the Card. Such a list is not TLV-coded but is a single constructed field built by concatenating several data objects together. A DOL is a concatenated list of entries, with each entry representing a single data object to be included in the constructed field. The format of each entry is a one-, two-, or three-byte tag identifying the data object, followed by a one-byte length which represents the number of bytes the field shall occupy in the command data. DOLs must contain only tags representing primitive data objects.

The Kernel applies the following rules to perform the concatenation:

- If the tag of any data object identified in the DOL is unknown or represents a constructed data object, the Kernel concatenates a value of hexadecimal zeroes with the length specified in the DOL entry.
- If the length specified in the DOL entry is less than the length of the data object, the leftmost bytes of the value of the data object are truncated if the data object has numeric (n) format, or the rightmost bytes for any other format.
- If the length specified in the DOL entry is greater than the length of the data object, the following padding applies:
  - Leading hexadecimal zeroes if the data object has numeric format
  - Trailing hexadecimal 'FF's if the data object has compressed numeric format
  - Trailing hexadecimal zeroes for any other format

The list of values is concatenated in the sequence in which the corresponding data objects appear in the DOL.

4.2 Working Variables

The Kernel makes use of a number of working variables that are not stored in the TLV Database. They are managed by the Kernel in an implementation specific way.

Working variables can be:

Local

The lifetime of local working variables is limited to the state transition process or procedure for which they are defined. These data objects do not appear in the data dictionary.

Global

The lifetime of global working variables is the same as the lifetime of the Kernel process. Global working variables are listed in the data dictionary without a tag.

These data objects are managed by the Kernel itself.

Global working variables can only be read and written by internal processing of the Kernel.

4.3 List Handling

A List acts as a container for a collection of ListItems. Each list has a unique name.
A ListItem is a single element in a List. For instance, a ListItem is a tag in a list of tags and a TLV-coded data object in a list of data objects.

The following lists of tags are supported:

- Discretionary Data Tag List
- Tag Mapping List

If DE/DS is implemented, then also the following lists of tags are supported:

- Tags To Read
- Tags To Read Yet
- Data Needed
- Read Data Tags To Validate Yet

The following lists of TLV-coded data objects are supported:

- Data Record
- Discretionary Data

If DE/DS is implemented, then also the following lists of TLV-coded data objects are supported:

- Data Envelopes To Write

- Data Envelopes To Write Yet
- Data To Send

The following list of strings is supported:

- Active AFL

The following methods are used to manipulate lists.

Initialise(List)

Initialises a List. This creates the List structure if it does not exist, and initialises its contents to be empty, i.e. the List contains no ListItems. This method can be called at any time during the operation of the Kernel in order to clear and reset a list.

After initialising a list that is stored in the TLV Database, IsPresent(TagOf(this empty list)) returns TRUE.

AddToList(ListItem, List)

If ListItem is not included in List, then adds ListItem to the end of List.

Updates ListItem if it is already included in the List.

RemoveFromList(ListItem, List)

Removes ListItem from the List if ListItem is present in List. Ignores otherwise.

AddListToList(List1, List2)

Adds the ListItems in List1 that are not yet included in List2 to the end of List2.

Updates ListItems that are already included in List2.

ListItem GetAndRemoveFromList(List)

Removes and returns the first ListItem from List. Returns NULL if List is empty.

T GetNextReadDataTagFromList(List)

Removes and returns the first tag from a list of tags that is categorised as being available from the Card using a READ DATA command.

Returns NULL if no tag is found.

Boolean IsEmptyList(List)

Returns TRUE if List contains no ListItems.

Boolean IsNotEmptyList(List)

Returns TRUE if List contains ListItems.

Boolean IncludedInList(ListItem, List)

Returns TRUE if List contains ListItem.

CreateDataRecord ()

```
Initialise(Data Record)
FOR every T in the first column of Table A.11
{
    IF      [IsPresent(T)]
    THEN
            TLV := GetTLV(T)
            IF      [T is included as one of the to be mapped tags in Tag Mapping List]
            THEN
                    Replace T in TLV with T' where T' is the tag following T in Tag
                    Mapping List.
                    AddToList(T'LV, Data Record)
            ELSE
                    AddToList(TLV, Data Record)
            ENDIF
    ENDIF
}
```

CreateDiscretionaryData ()

```
Initialise(Discretionary Data)
FOR every T in Discretionary Data Tag List
{
    IF      [IsPresent(T)]
    THEN
            TLV := GetTLV(T)
            IF      [T is included as one of the to be mapped tags in Tag Mapping List]
            THEN
                    Replace T in TLV with T' where T' is the tag following T in Tag
                    Mapping List.
                    AddToList(T'LV, Discretionary Data)
            ELSE
                    AddToList(TLV, Discretionary Data)
            ENDIF
    ENDIF
}
```

4.4 Configuration Data

The Configuration Data is passed to the Kernel at the start-up of the Kernel. The Kernel copies the configuration data objects in the TLV Database at the start of the transaction processing. Some configuration data objects must be present for the Kernel to process transactions correctly. If any mandatory data object is not present in the Configuration Data, then the transaction is aborted. Refer to section A.3 for the list of configuration data objects.

4.5 CA Public Key Database

The Kernel has access to a CA Public Key Database containing the Certification Authority Public Keys applicable for the RID of the selected AID. This CA Public Key Database is made available to Process C and is read-only.

The Certification Authority Public Key Index uniquely identifies the Certification Authority Public Key in the CA Public Key Database.

Table 4.2 lists the set of data objects that must be available in the CA Public Key Database for each RSA Certification Authority Public Key.

Table 4.2—RSA Certification Authority Public Key Related Data

Field Name	Length	Description	Format
Registered Application Provider Identifier (RID)	5	Identifies the payment system to which the Certification Authority Public Key is associated	b
Certification Authority Public Key Index	1	Identifies the Certification Authority Public Key in conjunction with the RID	b
Certification Authority Hash Algorithm Indicator	1	Identifies the hash algorithm used to produce the Hash Result in the digital signature scheme	b
Certification Authority Public Key Algorithm Indicator	1	Identifies the digital signature algorithm to be used with the Certification Authority Public Key. Always Hex value '01' for this version of the specification.	b
Certification Authority Public Key Modulus	var. (max 248)	Value of the modulus part of the Certification Authority Public Key	b
Certification Authority Public Key Exponent	1 or 3	Value of the exponent part of the Certification Authority Public Key, equal to 3 or $2^{16} + 1$	b
Certification Authority Public Key Check Sum (Only necessary if used to verify the integrity of the Certification Authority Public Key)	20 or 32	A check value calculated on the concatenation of all parts of the Certification Authority Public Key (RID, Certification Authority Public Key Index, Certification Authority Public Key Modulus, Certification Authority Public Key Exponent) using SHA-1. Alternatively, SHA-256 may be used.	b

Also ECC Certification Authority Public Keys must be stored in the CA Public Key Database. A separate table as shown in Table 4.3 is preferable to store the set of data objects that must be available in the CA Public Key Database for each ECC Certification Authority Public Key. This table includes (*x*, *y*) coordinates to avoid that the Kernel has to recompute the *y*-coordinate for each transaction.

Table 4.3—ECC Certification Authority Public Key Related Data

Field Name	Length	Description	Format
Registered Application Provider Identifier (RID)	5	Identifies the payment system to which the Certification Authority Public Key is associated.	b
Certification Authority Public Key Index	1	Identifies the Certification Authority Public Key in conjunction with the RID.	b
Certification Authority Public Key Algorithm Suite Indicator	1	Indicates the algorithms to be used with the Certification Authority Public Key. Always Hex value '10' for this version of the specification.	b
Certification Authority Public Key	$2*N_{FIELD}$	Representation of Certification Authority Public Key – (*x*, *y*) coordinates of the Certification Authority Public Key point – on the curve identified by the Certification Authority Public Key Algorithm Suite Indicator.	b
Certification Authority Public Key Check Sum (Only necessary if used to verify the integrity of the Certification Authority Public Key)	20 or 32	A check value calculated on the concatenation of the above data elements using SHA-1, SHA-256, or the hash algorithm associated with the Certification Authority Public Key Algorithm Suite	b

The Registered Application Provider Identifier is always included in the Certification Authority Public Key Check Sum computation. Whether it is stored within the CA Public Key Database for each Certification Authority Public Key or not is an implementation choice.

4.6 Certification Revocation List

The Kernel has access to a CRL applicable for the RID of the selected AID. This CRL is made available to the Kernel and is read-only.

Table 4.4 lists the set of data objects that must be available in the CRL for each revoked certificate. If when verifying an Issuer Public Key Certificate, the concatenation of RID, Certification Authority Public Key Index (Card) and Certificate Serial Number appears as an entry in the CRL, then the Issuer Public Key Certificate is revoked and certificate verification fails.

Table 4.4—Certification Revocation List Related Data

Field Name	Length	Description	Format
Registered Application Provider Identifier (RID)	5	Identifies the payment system to which the Certification Authority Public Key is associated.	b
Certification Authority Public Key Index	1	Identifies the Certification Authority Public Key in conjunction with the RID	b
Certificate Serial Number	3	Number unique to this certificate assigned by the certification authority	b
Additional Data	var.	Optional Terminal proprietary data, such as the date the certificate was added to the revocation list	b

4.7 Data Object Format

4.7.1 Format

All data objects known to the Kernel (other than local working variables) are listed in the data dictionary. All the length indications in the data dictionary are given in number of bytes. Data object formats are binary (b), numeric (n), compressed numeric (cn), alphanumeric (an), or alphanumeric special (ans).

Data objects that have the numeric (n) format are BCD encoded, right justified with leading hexadecimal zeros. Data objects that have the compressed numeric (cn) format are BCD encoded, left justified, and padded with trailing 'F's.

Note:
The length indicator in the numeric and compressed numeric format notations (e.g. n 4) specifies the number of digits and not the number of bytes.

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

Data objects that have the binary (b) format consist of either unsigned binary numbers or bit combinations that are defined in the specification.

When moving data from one entity to another (for example Card to Reader) or when concatenating data, the data must always be passed in decreasing order, regardless of how it is stored internally. The leftmost byte (byte 1) is the most significant byte.

Data objects are TLV-coded in the following cases:

- Data objects sent from the Card to the Kernel (RA Signal)
- Data objects sent to the Kernel at instantiation (e.g. configuration data objects) or with the ACT and DET Signals
- Data objects sent to the Terminal included in Data To Send
- Data objects included in the MSG and OUT Signals
- Data objects included in the Extended SDA Tag List Related Data

The tag field of TLV-coded data objects used in this specification is coded on one, two, or three bytes.

One- and two-byte tag fields used in the context of this specification are coded according to the rules specified in [EMV Book 3] section B1.

Three-byte tag fields used in the context of this specification are coded as follows:

- The use of primitive TLV-coded data objects with a tag field in the range '9F8101' to '9FEF7F' [4] is reserved for this specification.
- The use of primitive TLV-coded data objects with a tag field in the range '9FF001' to '9FFF7F' [4] is reserved for the payment systems.
- The use of primitive TLV-coded data objects with a tag field in the range 'DFF001' to 'DFFF7F' [4] is left to the discretion of the issuer.
- The use of constructed TLV-coded data objects with a tag field in the range 'BF8101' to 'BFEF7F' [4] is reserved for this specification.
- The use of constructed TLV-coded data objects with a tag field in the range 'BFF001' to 'BFFF7F' [4] is reserved for the payment systems.

A TLV coded data object returned by the Card with a tag field with length greater than 3 (i.e. a tag with Byte 3, b8 = 1b) is considered by the Kernel as a format error and processed as described in section 4.7.2.

4.7.2 Format Checking

It is the responsibility of the issuer to ensure that data in the Card is of the correct format. No format checking other than that specifically defined is mandated for the Kernel.

However, if during normal processing it is recognised that data read from the Card or provided by the Terminal is incorrectly formatted, the Kernel must perform the processing described in this section.

If a format error is detected in data received from the Card, the Kernel must update the Error Indication data object as follows:

'L2' in Error Indication := CARD DATA ERROR

If a format error is detected in data received from the Terminal, the Kernel must update the Error Indication data object as follows:

'L2' in Error Indication := TERMINAL DATA ERROR

The Kernel must then process the exception according to the state in which it occurs, as described here.

[4] Where the last byte has a value between '01' and '7F'.

States 1 and 2

The Kernel must prepare the User Interface Request Data 1, the Discretionary Data, and the Outcome Parameter Set and send an OUT Signal as shown here:

> 'Message Identifier' in User Interface Request Data 1 := ERROR – OTHER CARD
>
> 'Status' in User Interface Request Data 1 := NOT READY
>
> 'Hold Time' in User Interface Request Data 1 := Message Hold Time
>
> 'Status' in Outcome Parameter Set := END APPLICATION
>
> CreateDiscretionaryData ()
>
> SET 'UI Request on Outcome Present' in Outcome Parameter Set
>
> Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
> GetTLV(TagOf(Discretionary Data)),
> GetTLV(TagOf(User Interface Request Data 1))) Signal

The Kernel must then exit.

States 20, 21, 22, 23, 24, and 25

The Kernel must process the error as described under connector S202122232425 – E in Figure 6.13.

States 26 and 27

The Kernel must process the error as described under connector S2627 – C in Figure 6.16.

State 28

The Kernel must process the error as described under connector S28 – C in Figure 6.17.

States 29 and 30

The Kernel must process the error as described under connector S2930 – H in Figure 6.20.

4.8 Reserved for Future Use (RFU)

A bit specified as Reserved for Future Use (RFU) must be set as specified, or to 0b if no indication is given. An entity receiving a bit specified as RFU must ignore such a bit and must not change its behaviour, unless explicitly stated otherwise.

A data field having a value coded on multiple bits or bytes must not be set to a value specified as RFU. An entity receiving a data field having a value specified as RFU behaves as defined by a requirement that specifically addresses the situation.

5 C-APDU Commands

This chapter defines the commands and responses supported by Kernel 8.

5.1 Introduction

The INS byte of the C-APDU is structured according to [ISO/IEC 7816-4]. The coding of INS and its relationship to CLA are shown in Table 5.1.

Table 5.1—Coding of the Instruction Byte

CLA	INS	Meaning
'80'	'EA'	EXCHANGE RELAY RESISTANCE DATA
'80'	'AE'	GENERATE AC
'80'	'A8'	GET PROCESSING OPTIONS
'84'	'32'	READ DATA [5]
'00'	'B2'	READ RECORD
'84'	'34'	WRITE DATA [5]

The status bytes returned by the Card are coded as specified in [EMV Book 3] section 6.3.5. In addition to the status bytes specific to each command, the Card may return the status bytes shown in Table 5.2.

Table 5.2—Generic Status Bytes

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

[5] Only implemented for the DE/DS Implementation Option

5.2 Exchange Relay Resistance Data

5.2.1 Definition and Scope

The EXCHANGE RELAY RESISTANCE DATA command exchanges relay resistance related data with the Card.

5.2.2 Command Message

The EXCHANGE RELAY RESISTANCE DATA command message is coded according to Table 5.3.

Table 5.3—Exchange Relay Resistance Data Command Message

Code	Value
CLA	'80'
INS	'EA'
P1	'00'
P2	'00'
Lc	'04'
Data	Terminal Relay Resistance Entropy
Le	'00'

5.2.3 Data Field Returned in the Response Message

The expected data object returned in the response message is a primitive data object with tag '80' and length '0A'. The value field consists of the concatenation without delimiters (tag and length) of the value fields of the data objects specified in Table 5.4.

Table 5.4—Exchange Relay Resistance Data Response Message Data Field

Tag	Length	Byte	Value	Presence
'80'	'0A'	1-4	Device Relay Resistance Entropy	M
		5-6	Min Time For Processing Relay Resistance APDU	M
		7-8	Max Time For Processing Relay Resistance APDU	M
		9-10	Device Estimated Transmission Time For Relay Resistance R-APDU	M

5.2.4 Status Bytes

The status bytes that may be sent in response to the EXCHANGE RELAY RESISTANCE DATA command are listed in Table 5.5.

Table 5.5—Status Bytes for Exchange Relay Resistance Data Command

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

5.3 Generate AC

5.3.1 Definition and Scope

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

5.3.2 Command Message

The GENERATE AC command message is coded according to Table 5.6.

Table 5.6—Generate AC Command Message

Code	Value
CLA	'80'
INS	'AE'
P1	Reference Control Parameter (see A.1.108)
P2	See Table 5.7
Lc	var.
Data	CDOL1 Related Data
Le	'00'

The coding of P2 is shown in Table 5.7.

Table 5.7—P2 of Generate AC Command

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0								No more commands to follow
1								More commands to follow
	x	x	x	x	x	x	x	Each bit RFU

The data field of the command message contains CDOL1 Related Data coded according to CDOL1 following the rules defined in section 4.1.4.

5.3.3 Data Field Returned in the Response Message

The expected data object returned in the response message is a constructed data object with tag '77' (Response Message Template Format 2) as shown in Table 5.8. Data objects in Response Message Template Format 2 may appear in any order.

The Kernel aborts the transaction if any mandatory primitive data object listed in Table 5.8 is not present in the TLV Database after parsing the GENERATE AC response message data field.

Table 5.8—GENERATE AC Response Message Data Field

Tag	Value		Presence
'77'	Response Message Template Format 2		M
	'9F27'	Cryptogram Information Data	M
	'9F36'	Application Transaction Counter	M
	'9F8102'	Cardholder Verification Decision	M
	'9F8106'	Authenticated Application Data	O
	'9F8104'	Card TVR	O
	'9F8108'	Restart Indicator	O
	'9F26'	Application Cryptogram	M
	'9F10'	Issuer Application Data	M
	'9F8105'	Enhanced Data Authentication MAC	M

5.3.4 Status Bytes

The status bytes that may be sent in response to the GENERATE AC command are listed in Table 5.9.

Table 5.9—Status Bytes for Generate AC Command

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

5.4 Get Processing Options

5.4.1 Definition and Scope

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

5.4.2 Command Message

The GET PROCESSING OPTIONS command message is coded according to Table 5.10.

Table 5.10—Get Processing Options Command Message

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

The data field of the command message consists of PDOL Related Data. PDOL Related Data is the Command Template with tag '83' and with a value field coded according to the PDOL provided by the Card in the response to the SELECT command. If the PDOL is not provided by the Card, the length field of the Command Template is set to zero. Otherwise the length field is the total length of the value fields of the data objects transmitted to the Card. The value fields are concatenated according to the rules defined in section 4.1.4.

5.4.3 Data Field Returned in the Response Message

The expected data object returned in the response message is a constructed data object with tag '77' (Response Message Template Format 2) as shown in Table 5.11. Data objects in Response Message Template Format 2 may appear in any order.

The Kernel aborts the transaction if any mandatory primitive data object listed in Table 5.11 is not present in the TLV Database after parsing the GET PROCESSING OPTIONS response message data field.

Table 5.11—Get Processing Options Response Message Data Field

Tag	Value		Presence
'77'	Response Message Template Format 2		M
	'82'	Application Interchange Profile	M
	'94'	Application File Locator	M
	'9F8103'	Card Key Data	M
	'8C'	CDOL1	O
	'9F36'	Application Transaction Counter	O

5.4.4 Status Bytes

The status bytes that may be sent in response to the GET PROCESSING OPTIONS command are listed in Table 5.12.

Table 5.12—Status Bytes for Get Processing Options Command

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

5.5 Read Data

5.5.1 Definition and Scope

The READ DATA command is used to retrieve one of the Data Envelope x data objects from the Card. It must be noted that the READ DATA command is only implemented for the DE/DS Implementation Option.

5.5.2 Command Message

The READ DATA command message is coded according to Table 5.13.

Table 5.13—Read Data Command Message

Code	Value
CLA	'84'
INS	'32'
P1	'00'
P2	'00'
Lc	'03'
Data	Tag of Data Envelope x ('9F8111' – '9F811A')
Le	'00'

5.5.3 Data Field Returned in the Response Message

The expected data field of the response message contains two fields (no TLV-coding): the encryption of the TLV-coded Data Envelope x referred to in the data field of the command message followed by an 8-byte MAC computed over the encrypted Data Envelope x.

5.5.4 Status Bytes

The status bytes that may be sent in response to the READ DATA command are listed in Table 5.14.

Table 5.14—Status Bytes for Read Data Command

SW1	SW2	Meaning
'69'	'85'	Conditions of use not satisfied
'6A'	'86'	Incorrect parameters P1-P2
'6A'	'88'	Referenced data (data object) not found
'90'	'00'	Normal processing

5.6 Read Record

5.6.1 Definition and Scope

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

5.6.2 Command Message

The READ RECORD command message is coded according to Table 5.15.

Table 5.15—Read Record Command Message

Code	Value
CLA	'00'
INS	'B2'
P1	Record number
P2	See Table 5.16
Lc	Not present
Data	Not present
Le	'00'

Table 5.16 specifies the coding of P2 of the READ RECORD command.

Table 5.16—P2 of Read Record Command

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

5.6.3 Data Field Returned in the Response Message

The expected data field of the response message contains the record requested by the command. For SFIs in the range 1-10, a record that is not encrypted is delivered as a constructed TLV data object with tag '70' as shown in Table 5.17.

Table 5.17—Read Record Response Message Data Field

'70'	Length	Record Body

A record may also be delivered using a tag of 'DA' as shown in Table 5.18. This signifies that the data has been encrypted using the session key and must be decrypted by the Kernel, yielding TLV data that is formatted as if it were in a '70' template.

Table 5.18—Encrypted Read Record Response Message Data Field

'DA'	Length	Encrypted Record Body

5.6.4 Status Bytes

The status bytes that may be sent in response to the READ RECORD command are listed in Table 5.19.

Table 5.19—Status Bytes for Read Record Command

SW1	SW2	Meaning
'69'	'85'	Conditions of use not satisfied
'6A'	'82'	Wrong parameters P1 P2; file not found
'6A'	'83'	Wrong parameters P1 P2; record not found
'6A'	'86'	Incorrect parameters P1 P2
'90'	'00'	Normal processing

5.7 Write Data

5.7.1 Definition and Scope

The WRITE DATA command is used to store a value in Data Envelope x. It must be noted that WRITE DATA is only implemented for the DE/DS Implementation Option.

5.7.2 Command Message

The WRITE DATA command message is coded according to Table 5.20.

Table 5.20—Write Data Command Message

Code	Value
CLA	'84'
INS	'34'
P1	'00'
P2	See Table 5.21
Lc	var.
Data	Encrypted form of TLV-coded Data Envelope x
Le	'00'

The coding of P2 is shown in Table 5.21.

Table 5.21—P2 of Write Data Command

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0								No more commands to follow
1								More commands to follow
	x	x	x	x	x	x	x	Each bit RFU

5.7.3 Data Field Returned in the Response Message

The expected data field in the response message of the WRITE DATA command contains an 8-byte MAC computed over the plaintext TLV-coded Data Envelope x which is included in encrypted form in the data field of the command message.

5.7.4 Status Bytes

The status bytes that may be sent in response to the WRITE DATA command are listed in Table 5.22.

Table 5.22—Status Bytes for Write Data Command

SW1	SW2	Meaning
'67'	'00'	Wrong length
'6A'	'86'	Incorrect parameters P1-P2
'6A'	'88'	Referenced data (data object) not found
'90'	'00'	Normal processing

6 Kernel State Machine

This section describes the transaction processing of the Kernel after it has been initiated by Process M.

6.1 Principles

The transaction processing is specified as a state machine that is triggered by external Signals that cause state transitions.

These principles are used in order to present the application concepts. For the actual implementation, it is not mandatory to follow the same principles. However, the implementation must behave in a way that is indistinguishable from the behaviour specified in this chapter.

The Kernel receives incoming Signals on two different Queues. The first Queue is used by Process M (for the ACT Signal), by Process P (for the RA and L1RSP Signals) and by the Terminal (for the DET Signal) [6]. The second Queue is used to communicate with Process C only.

The Kernel can be in a state for which a state transition is triggered by more than one Signal on both Queues. In this case the Signal on the first Queue must be processed first and this Signal will trigger the state transition. If for example in Figure 6.8 there is an RA Signal on the first Queue and a RECORD DECRYPTED Signal on the second Queue, then the RA Signal must be handled first.

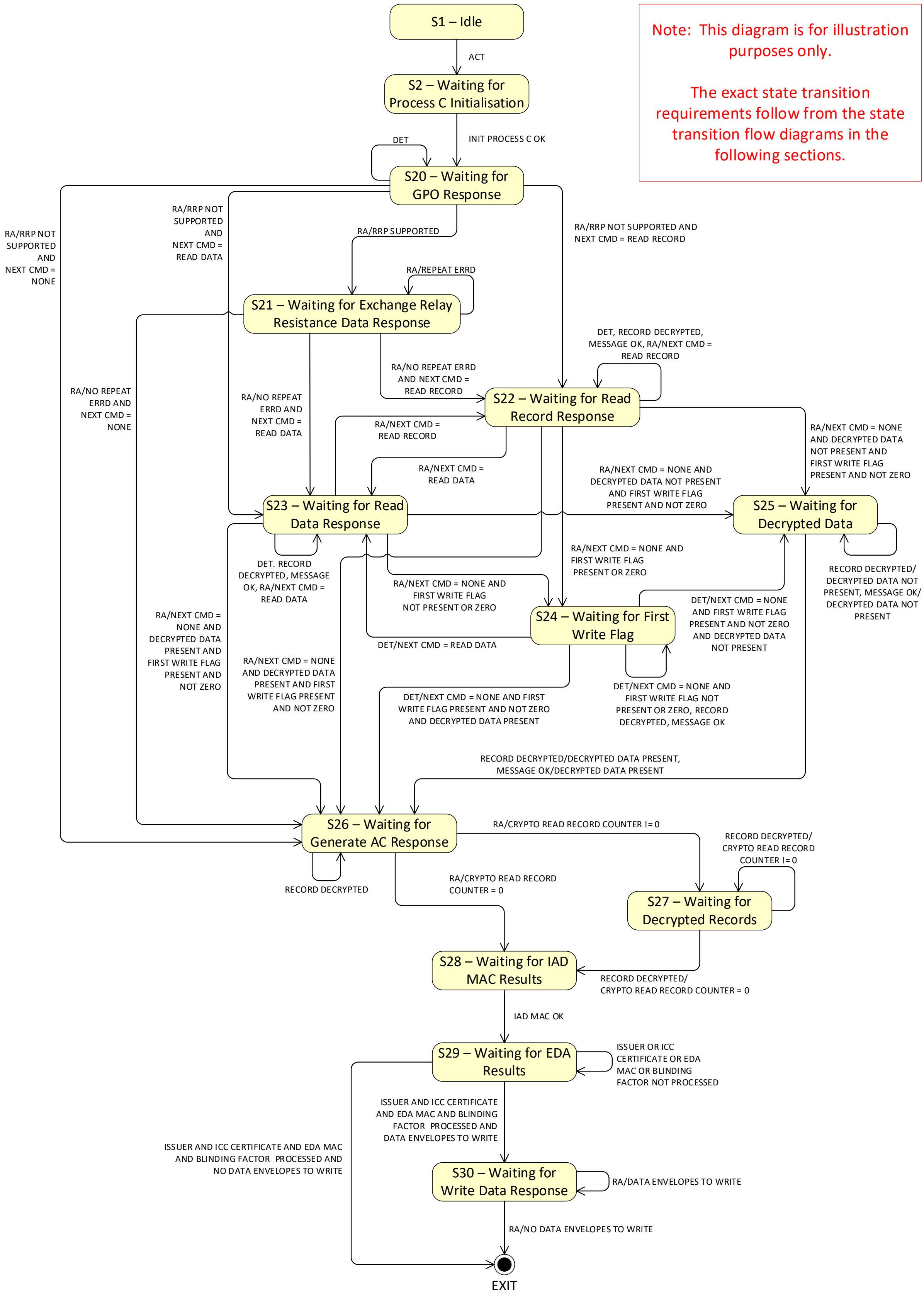
The Kernel can be in a state for which a state transition is triggered by Signals that are not at the top of a Queue. In this case the state transition is triggered by the first Signal on the Queue that causes a state transition. Signals on the Queue that are not handled in the current state must not be lost and must stay on the Queue for processing in subsequent states.

6.2 Kernel State Diagram

A high level state diagram is presented for information in Figure 6.1. The diagram is for illustration purposes only and represents the full state diagram including the DE/DS Implementation Option. The exact state transition requirements follow from the state transition flow diagrams in the following sections.

[6] Only implemented for the DE/DS Implementation Option

Figure 6.1—Kernel State Diagram



6.3 State Transitions

This section specifies the flow diagrams of the state transitions between the states of the Kernel state diagram caused by external Signals.

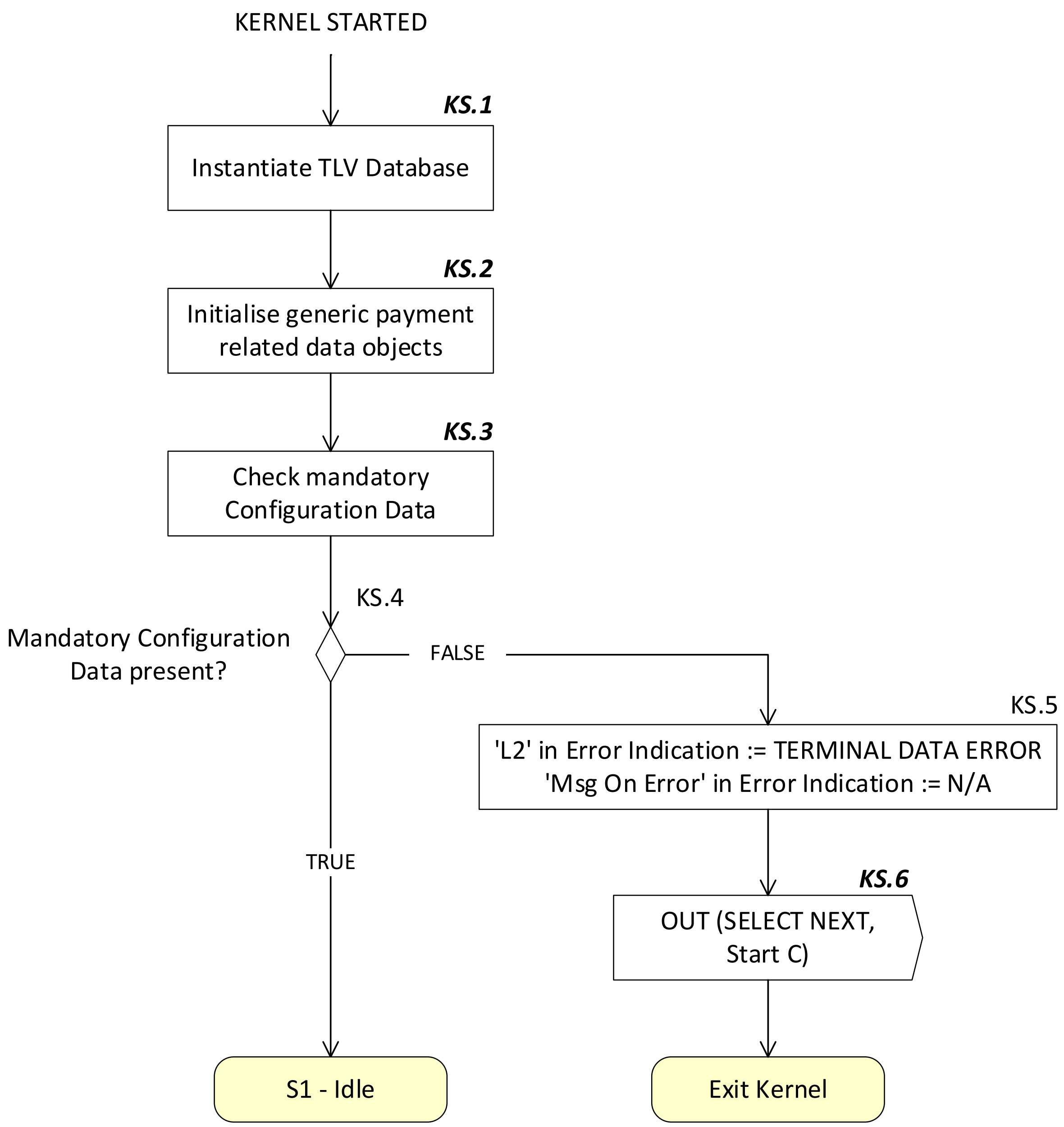
6.3.1 Kernel Started

Figure 6.2 shows the flow diagram of start-up of the Kernel. Symbols in this diagram are labelled KS.X.

Table 6.0—State KS.1 Local Variables

Name	Length	Format	Description
Mandatory Configuration Data Present	1	b	Boolean used to store the result of verification of the presence of mandatory configuration data
T	var.	b	Tag of TLV-coded string

Figure 6.2—Kernel Started Flow Diagram



KS.1

Instantiate an empty TLV Database to be able to store the tags identified in Table A.37 and Proprietary Tags if present.

For each configuration data object present in the Configuration Data, create an entry and store the value from the Configuration Data in the TLV Database.

If Proprietary Tags are present, then for each data object present in the Proprietary Tags that is not empty, create an entry in the TLV Database and populate the entry with the proprietary data object value.

KS.2

Initialise Outcome Parameter Set as follows:

Outcome Parameter Set := '0000 … 00'
'Status' in Outcome Parameter Set := N/A
'Start' in Outcome Parameter Set := N/A
'CVM' in Outcome Parameter Set := N/A
CLEAR 'UI Request on Outcome Present' in Outcome Parameter Set
CLEAR 'UI Request on Restart Present' in Outcome Parameter Set
CLEAR 'Data Record Present' in Outcome Parameter Set
SET 'Discretionary Data Present' in Outcome Parameter Set
'Receipt' in Outcome Parameter Set := N/A
'Alternate Interface Preference' in Outcome Parameter Set := N/A
'Field Off Request' in Outcome Parameter Set := N/A
'Removal Timeout' in Outcome Parameter Set := 0
'Online Response Data' in Outcome Parameter Set := N/A

Initialise User Interface Request Data 1 as follows:

User Interface Request Data 1 := '0000 … 00'
'Message Identifier' in User Interface Request Data 1 := N/A
'Status' in User Interface Request Data 1 := N/A
'Hold Time' in User Interface Request Data 1 := Message Hold Time
'Language Preference' in User Interface Request Data 1 := '0000000000000000'

Initialise User Interface Request Data 2 as follows:

User Interface Request Data 2 := '0000 … 00'
'Message Identifier' in User Interface Request Data 2 := N/A
'Status' in User Interface Request Data 2 := N/A
'Hold Time' in User Interface Request Data 2 := '000000'
'Language Preference' in User Interface Request Data 2 := '0000000000000000'

Initialise Error Indication as follows:

Error Indication := '0000 … 00'
'L1' in Error Indication := OK
'L2' in Error Indication := OK
'L3' in Error Indication := OK
'SW12' in Error Indication := '0000'
'Msg On Error' in Error Indication := ERROR – OTHER CARD

KS.3

```
Mandatory Configuration Data Present := TRUE
FOR every mandatory configuration data object T in Table A.38
{
    IF      [IsNotPresent(T) OR IsEmpty(T)
            OR GetLength(T) is not within the range specified by Length field of the data
            object with tag T in the data dictionary (Annex A)]
    THEN
        Mandatory Configuration Data Present := FALSE
        EXIT loop
    ENDIF
}
```

KS.6

'Status' in Outcome Parameter Set := SELECT NEXT
'Start' in Outcome Parameter Set := C
CreateDiscretionaryData ()
Send OUT(GetTLV(TagOf(Outcome Parameter Set)), GetTLV(TagOf(Discretionary Data))) Signal

6.3.2 State 1 – Idle

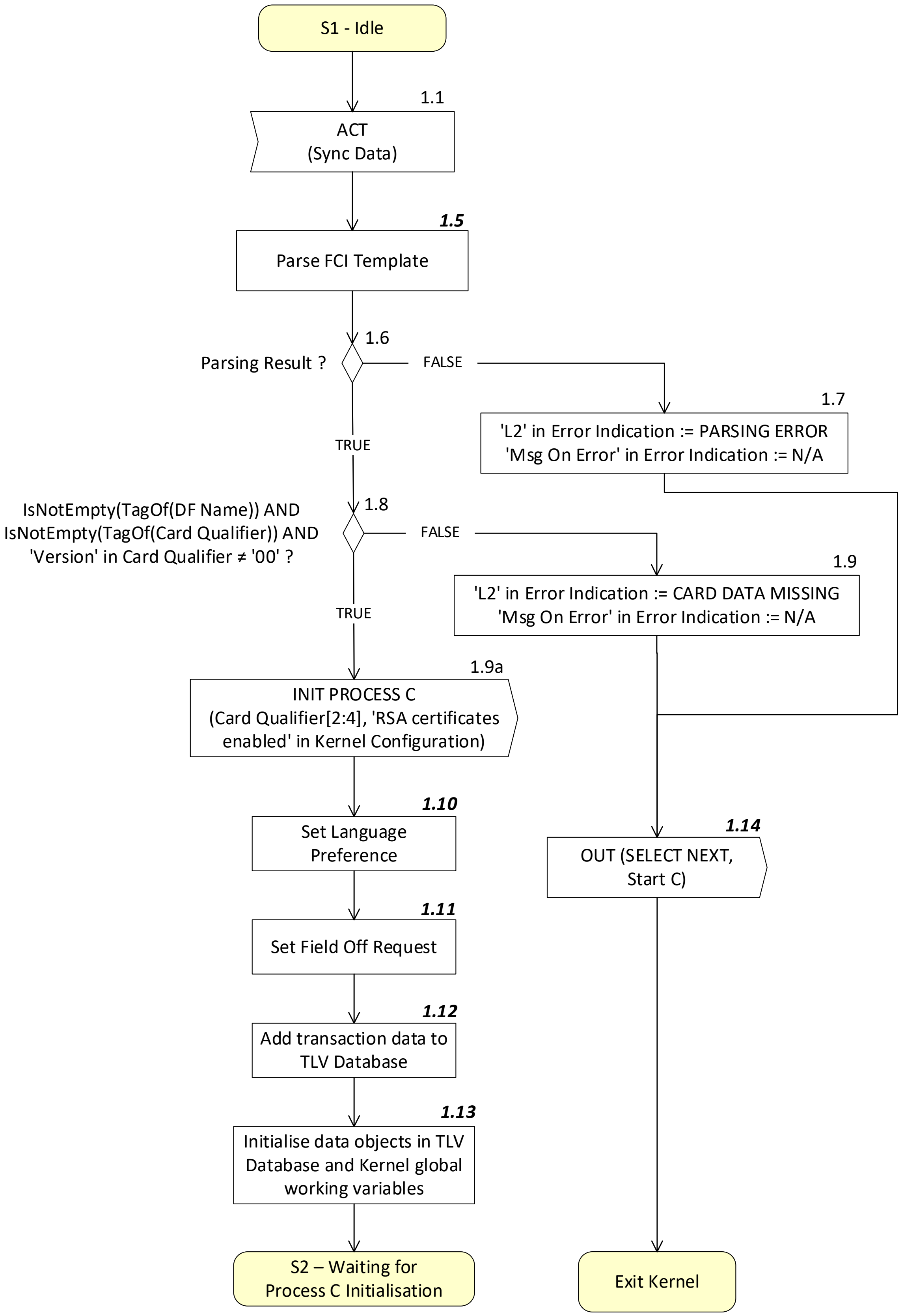
Table 6.1 shows the local variables used in `S1 – Idle`.

Table 6.1—State 1 Local Variables

Name	Length	Format	Description
Sync Data	var.	b	List of data objects returned with ACT Signal
T	var.	b	Tag of TLV-coded string
L	var.	b	Length of TLV-coded string
Parsing Result	1	b	Boolean used to store result of parsing the FCI

Figure 6.3 shows the flow diagram of `S1 – Idle`. Symbols in this diagram are labelled 1.X.

Figure 6.3—State 1 Flow Diagram



1.5

Parse and store the File Control Information Template:

```
Parsing Result := FALSE
FOR every TLV in Sync Data
{
    IF      [T = TagOf(File Control Information Template)]
    THEN
            Parsing Result := ParseAndStoreCardResponse(TLV)
            EXIT loop
    ENDIF
}
```

1.10

If the Language Preference is returned from the Card, then copy it to 'Language Preference' in User Interface Request Data 1 and User Interface Request Data 2:

```
IF      [IsNotEmpty(TagOf(Language Preference))]
THEN
        'Language Preference' in User Interface Request Data 1 := Language Preference
        'Language Preference' in User Interface Request Data 2 := Language Preference
        If the length of Language Preference is less than 8 bytes, then pad 'Language
        Preference' in User Interface Request Data 1 and User Interface Request Data 2 with
        trailing hexadecimal zeroes to 8 bytes.
ENDIF
```

1.11

```
IF ['Support for field off detection' in Card Qualifier is set]
THEN
        'Field Off Request' in Outcome Parameter Set := Hold Time Value
ENDIF
```

1.12

Add the transaction data provided in the ACT Signal to the TLV Database:

```
FOR every TLV in Sync Data
{
    IF      [T ≠ TagOf(File Control Information Template)]
    THEN
            IF      [IsKnown(T)
                    AND update conditions of T include ACT Signal]
            THEN
                    Store LV in the TLV Database for tag T
            ENDIF
    ENDIF
}
```

1.13

CVM Results := '000000'

'Decision' in Kernel Decision := ACCEPT

Terminal Verification Results := '0000000000'

SET 'Kernel 8 processing and TVR format' in Terminal Verification Results

EDA Status := '00'

Terminal Capabilities[1] := Card Data Input Capability

Terminal Capabilities[2] := '00'

Terminal Capabilities[3] := Security Capability

RRP Counter := '00'

Relay Resistance Time Excess := '0000'

Kernel Qualifier := '0200000000000000'

'Local authentication enabled' in Kernel Qualifier := 'Local authentication' in Terminal Capabilities

Generate unpredictable number as specified in [EMV Book E] section 8.1 and store in Unpredictable Number.

Crypto Read Data Counter := '00'

Crypto Read Record Counter := '00'

Initialise Extended SDA Tag List Related Data with an empty string

The following data objects are specific to data storage and data exchange processing and are only initialised if the DE/DS Implementation Option is implemented:

```
    Read Data Status := '80'
    Write Data Status := '00'
    Initialise(Data Needed)
    Initialise(Data To Send)
    Initialise(Tags To Read Yet)
    Initialise(Read Data Tags To Validate Yet)
    Initialise(Data Envelopes To Write Yet)

IF      [IsPresent(TagOf(Tags To Read))]
        IF      [IsEmptyList(Tags To Read)]
        THEN
                AddToList(TagOf(Tags To Read), Data Needed))
        ELSE
                AddListToList(Tags To Read, Tags To Read Yet)
        ENDIF
ENDIF
IF      [IsPresent(TagOf(Data Envelopes To Write))]
        IF      [IsEmptyList(Data Envelopes To Write)]
        THEN
                AddToList(TagOf(Data Envelopes To Write), Data Needed))
        ELSE
                AddListToList(Data Envelopes To Write, Data Envelopes To Write Yet)
        ENDIF
ENDIF
```

<u>1.14</u>

```
'Status' in Outcome Parameter Set := SELECT NEXT
'Start' in Outcome Parameter Set := C
CreateDiscretionaryData ()
Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
            GetTLV(TagOf(Discretionary Data))) Signal
```

6.3.3 State 2 – Waiting for Process C Initialisation

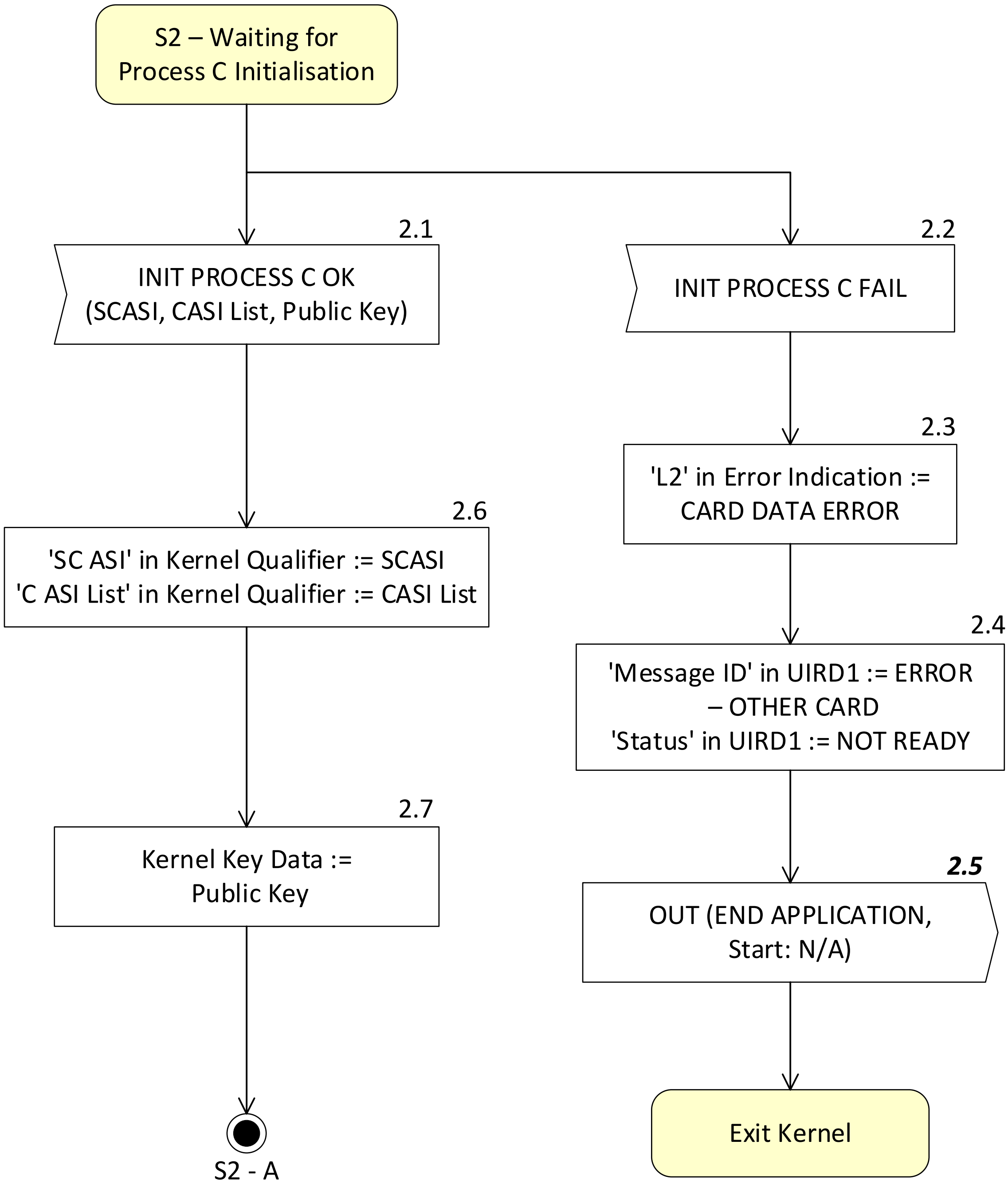
Table 6.2 shows the local variables used in `S2 – Waiting for Process C Initialisation`.

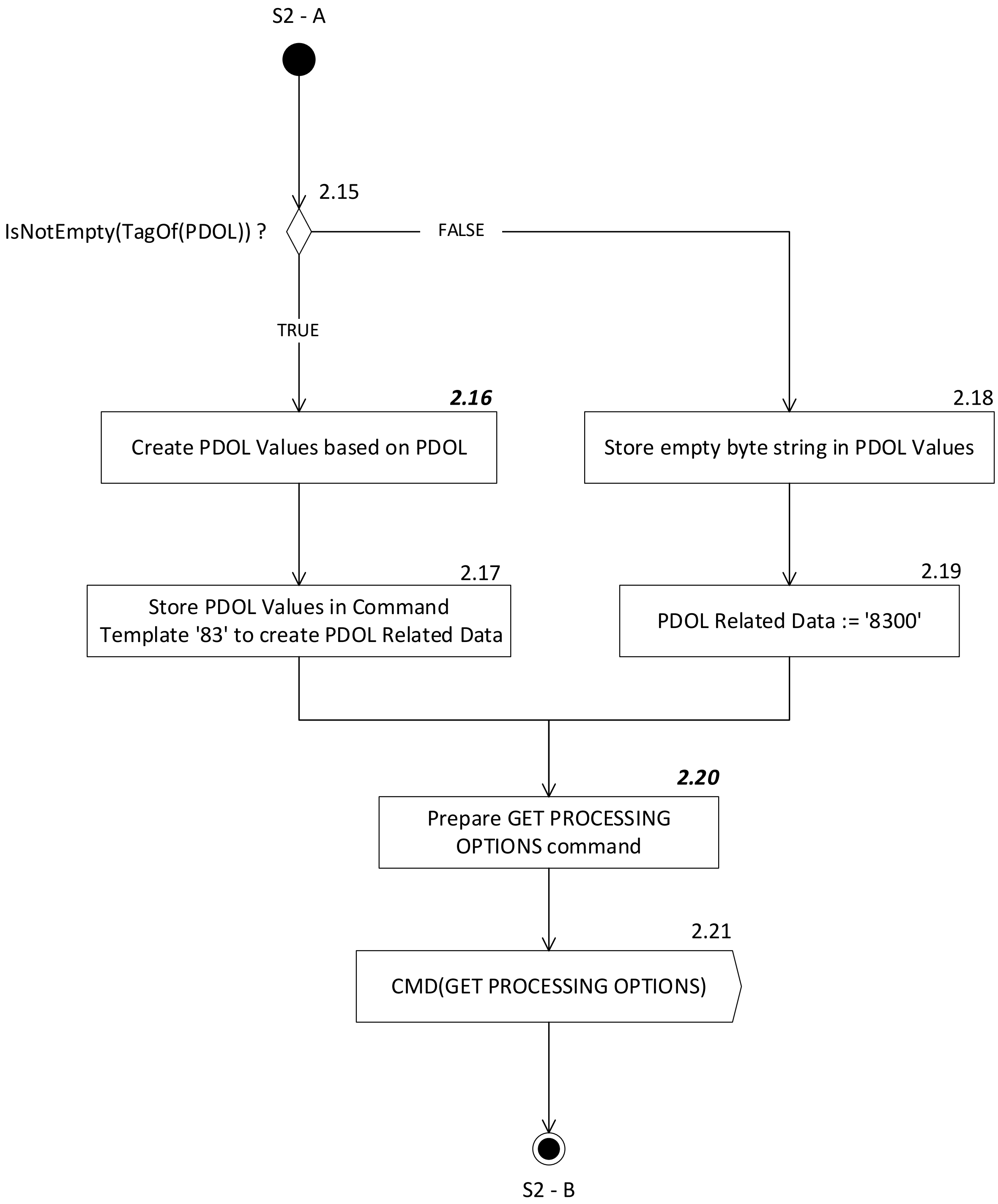
Table 6.2—State 2 Local Variables

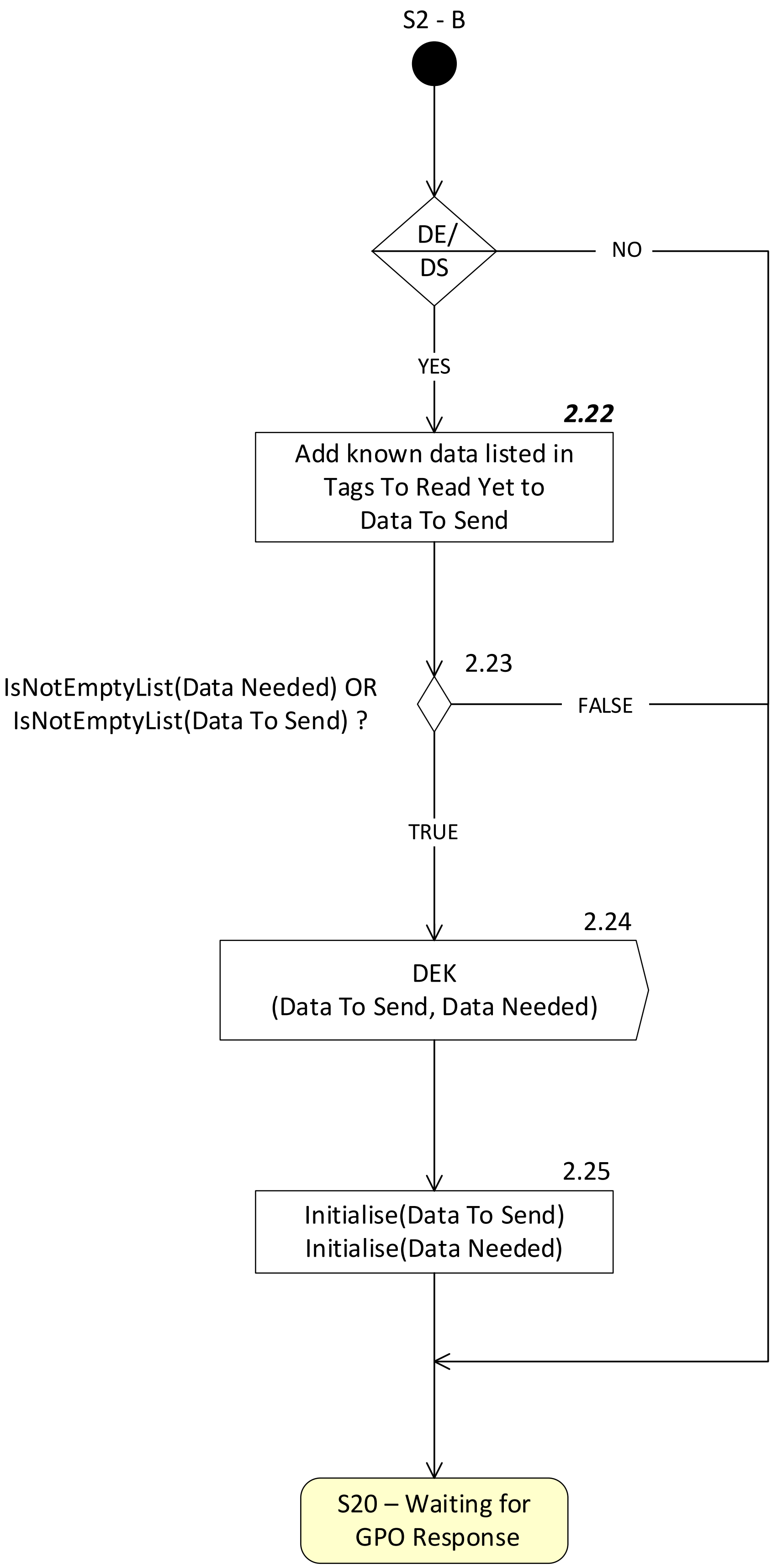
Name	Length	Format	Description
CASI List	3	b	Certificate ASI list returned by Process C
SCASI	1	b	Secure channel ASI selected by Process C
Public Key	var.	b	Public key returned by Process C

Figure 6.4 shows the flow diagram of `S2 – Waiting for Process C Initialisation`. Symbols in this diagram are labelled 2.X.

Figure 6.4—State 2 Flow Diagram




S2 - A
2.15
IsNotEmpty(TagOf(PDOL)) ?
FALSE
TRUE
2.16
Create PDOL Values based on PDOL
2.18
Store empty byte string in PDOL Values
2.17
Store PDOL Values in Command Template '83' to create PDOL Related Data
2.19
PDOL Related Data := '8300'
2.20
Prepare GET PROCESSING OPTIONS command
2.21
CMD(GET PROCESSING OPTIONS)
S2 - B


S2 - B
DE/
DS
NO
YES
2.22
Add known data listed in
Tags To Read Yet to
Data To Send
2.23
IsNotEmptyList(Data Needed) OR
IsNotEmptyList(Data To Send) ?
FALSE
TRUE
2.24
DEK
(Data To Send, Data Needed)
2.25
Initialise(Data To Send)
Initialise(Data Needed)
S20 – Waiting for
GPO Response

Note that the symbols 2.22, 2.23, 2.24, and 2.25 are only implemented for the DE/DS Implementation Option.

2.5

'Status' in Outcome Parameter Set := END APPLICATION

CreateDiscretionaryData ()

SET 'UI Request on Outcome Present' in Outcome Parameter Set

```
Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
         GetTLV(TagOf(Discretionary Data)),
         GetTLV(TagOf(User Interface Request Data 1))) Signal
```

2.16

Use PDOL to create PDOL Values as a concatenated list of data objects without tags or lengths following the rules specified in section 4.1.4.

2.20

Prepare GET PROCESSING OPTIONS command as specified in section 5.4.

2.22

```
FOR every entry T in Tags To Read Yet
{
    IF    [IsNotEmpty(T)]
    THEN
          AddToList(GetTLV(T), Data To Send)
          RemoveFromList(T, Tags To Read Yet)
    ENDIF
}
```

6.3.4 State 20 – Waiting for GPO Response

Table 6.3 shows the local variables used in `S20 – Waiting for GPO Response`.

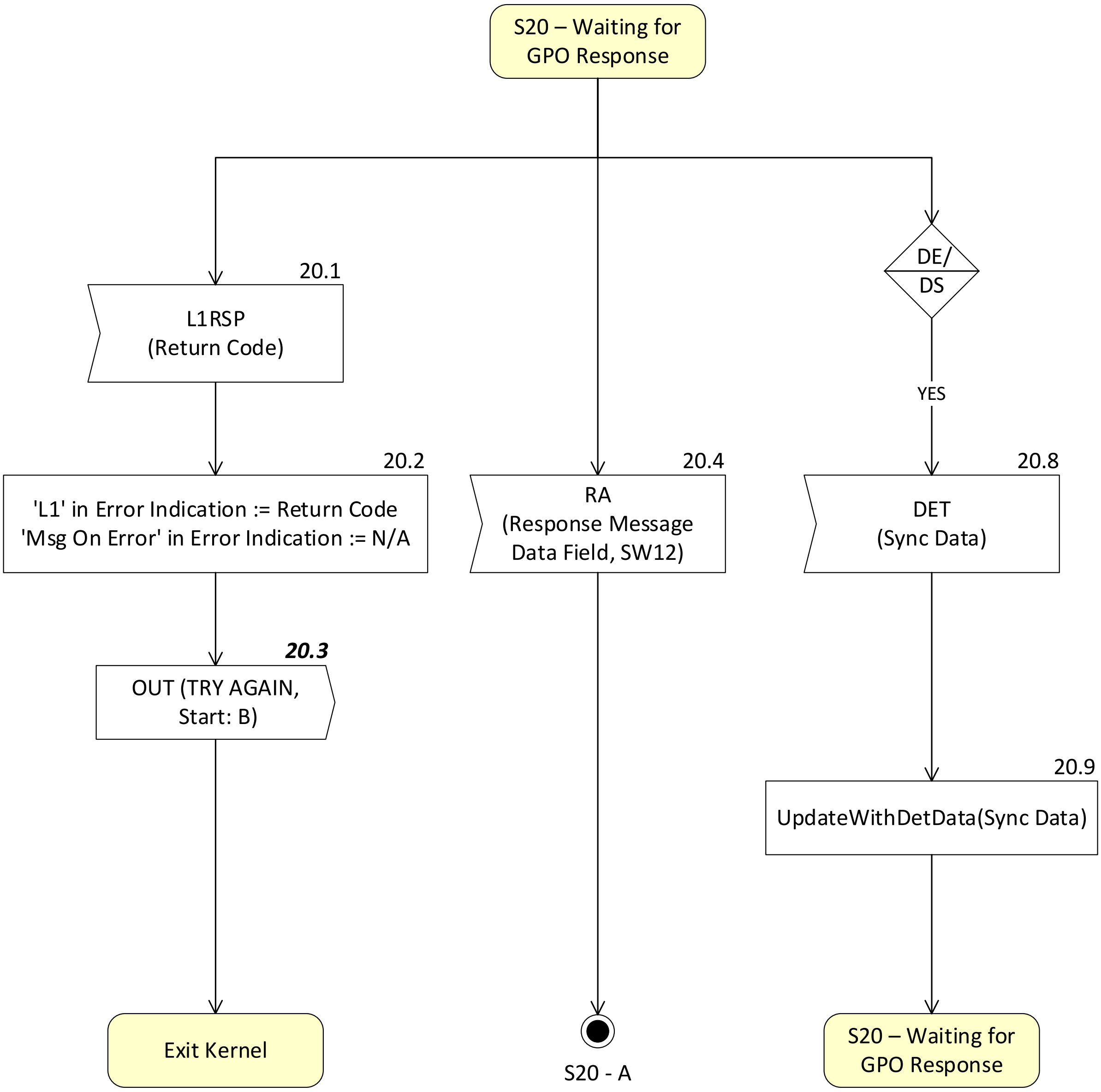
Table 6.3—State 20 Local Variables

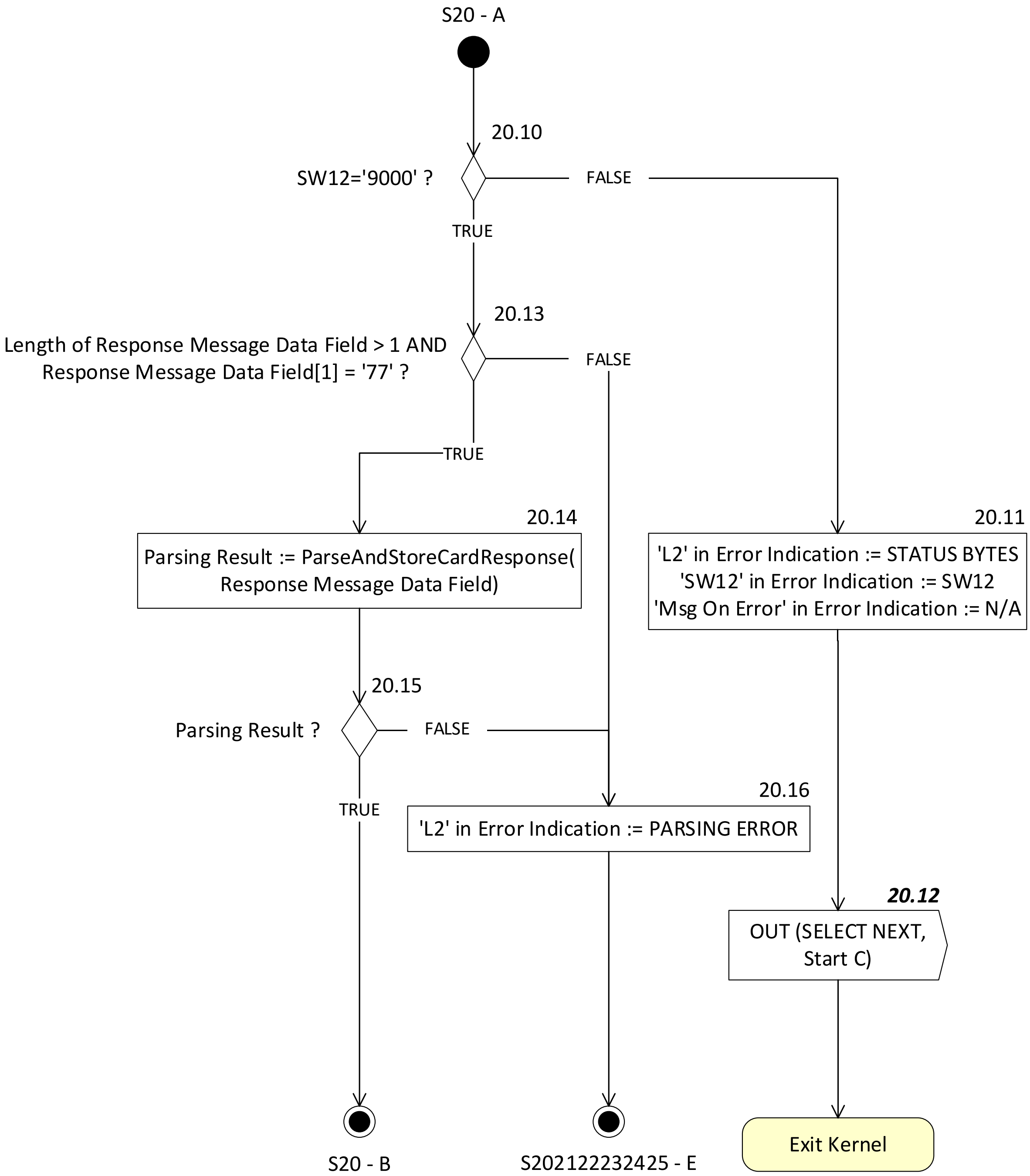
Name	Length	Format	Description
Return Code	1	b	Value returned with L1RSP Signal (TIMEOUT ERROR, PROTOCOL ERROR, TRANSMISSION ERROR)
Sync Data [7]	var.	b	List of data objects returned with DET Signal
Parsing Result	1	b	Boolean used to store result of parsing a TLV-coded string
SW12	2	b	Status bytes
Response Message Data Field	var. up to 256	b	TLV-coded string included in R-APDU of GET PROCESSING OPTIONS

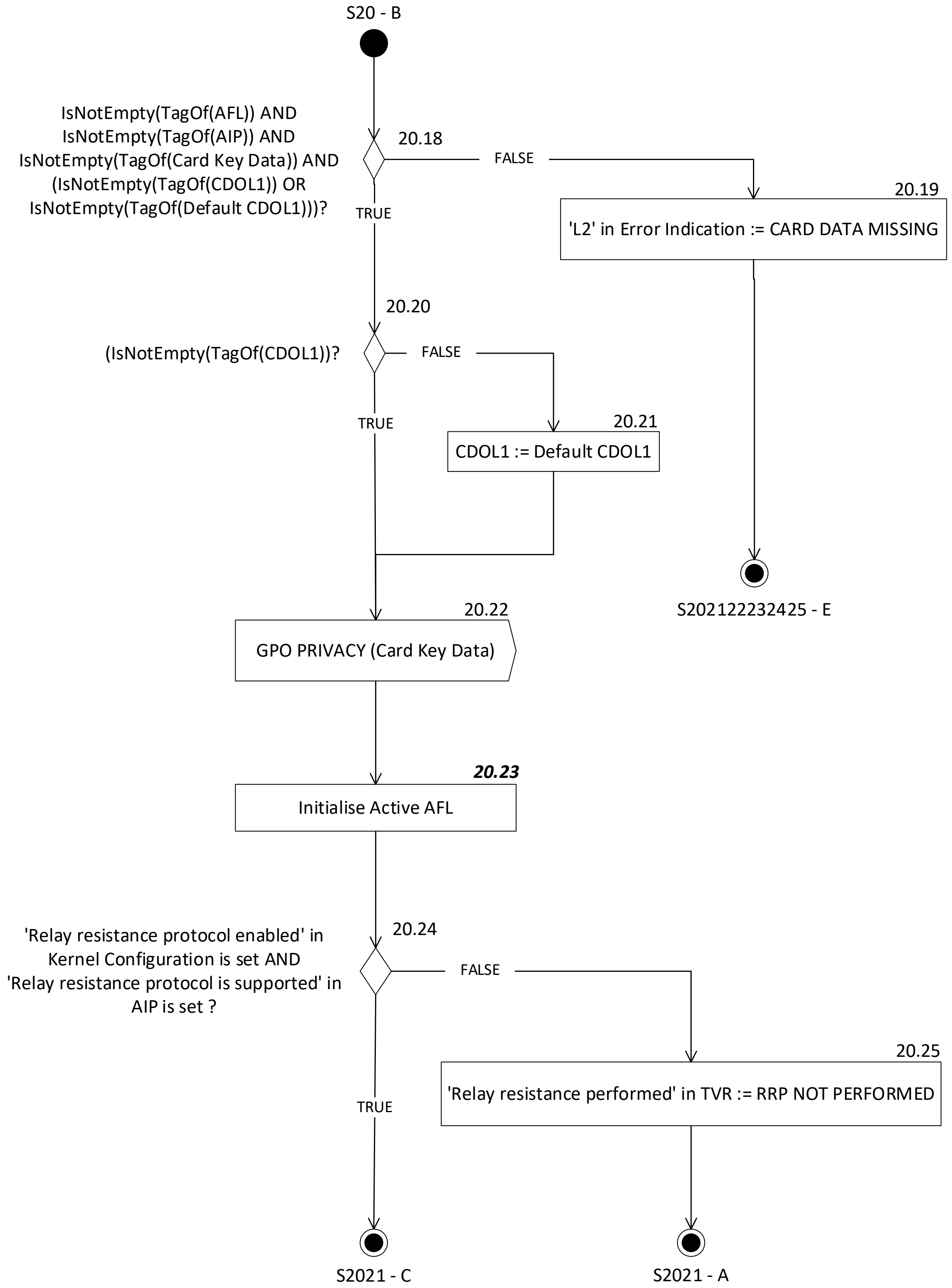
Figure 6.5 shows the flow diagram of `S20 – Waiting for GPO Response`. Symbols in this diagram are labelled 20.X.

[7] Only implemented for the DE/DS Implementation Option

Figure 6.5—State 20 Flow Diagram




S20 - A
20.10
SW12='9000' ?
FALSE
TRUE
20.13
Length of Response Message Data Field > 1 AND
Response Message Data Field[1] = '77' ?
FALSE
TRUE
20.14
Parsing Result := ParseAndStoreCardResponse(
Response Message Data Field)
20.11
'L2' in Error Indication := STATUS BYTES
'SW12' in Error Indication := SW12
'Msg On Error' in Error Indication := N/A
20.15
Parsing Result ?
FALSE
TRUE
20.16
'L2' in Error Indication := PARSING ERROR
20.12
OUT (SELECT NEXT,
Start C)
S20 - B
S202122232425 - E
Exit Kernel


S20 - B
IsNotEmpty(TagOf(AFL)) AND
IsNotEmpty(TagOf(AIP)) AND
IsNotEmpty(TagOf(Card Key Data)) AND
(IsNotEmpty(TagOf(CDOL1)) OR
IsNotEmpty(TagOf(Default CDOL1)))?
20.18
FALSE
TRUE
20.19
'L2' in Error Indication := CARD DATA MISSING
20.20
(IsNotEmpty(TagOf(CDOL1))?
FALSE
TRUE
20.21
CDOL1 := Default CDOL1
S202122232425 - E
20.22
GPO PRIVACY (Card Key Data)
20.23
Initialise Active AFL
'Relay resistance protocol enabled' in
Kernel Configuration is set AND
'Relay resistance protocol is supported' in
AIP is set ?
20.24
FALSE
TRUE
20.25
'Relay resistance performed' in TVR := RRP NOT PERFORMED
S2021 - C
S2021 - A

Note that the symbols 20.8 and 20.9 are only implemented for the DE/DS Implementation Option.

20.3

'Status' in Outcome Parameter Set := TRY AGAIN

'Start' in Outcome Parameter Set := B

CreateDiscretionaryData ()

Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
    GetTLV(TagOf(Discretionary Data))) Signal

20.12

'Field Off Request' in Outcome Parameter Set := N/A

'Status' in Outcome Parameter Set := SELECT NEXT

'Start' in Outcome Parameter Set := C

CreateDiscretionaryData ()

Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
    GetTLV(TagOf(Discretionary Data))) Signal

20.23

```
Initialise(Active AFL)
FOR i := 1 TO GetLength(TagOf(Application File Locator)) div 4
{
    IF      [Application File Locator [(i*4)-3][8:4] ≤ 10]
    THEN
            Parse Application File Locator [(i*4)-3:i*4] into individual records and add
            each record to Active AFL.
    ENDIF
}
```

6.3.5 State 21 – Waiting for Exchange Relay Resistance Data Response

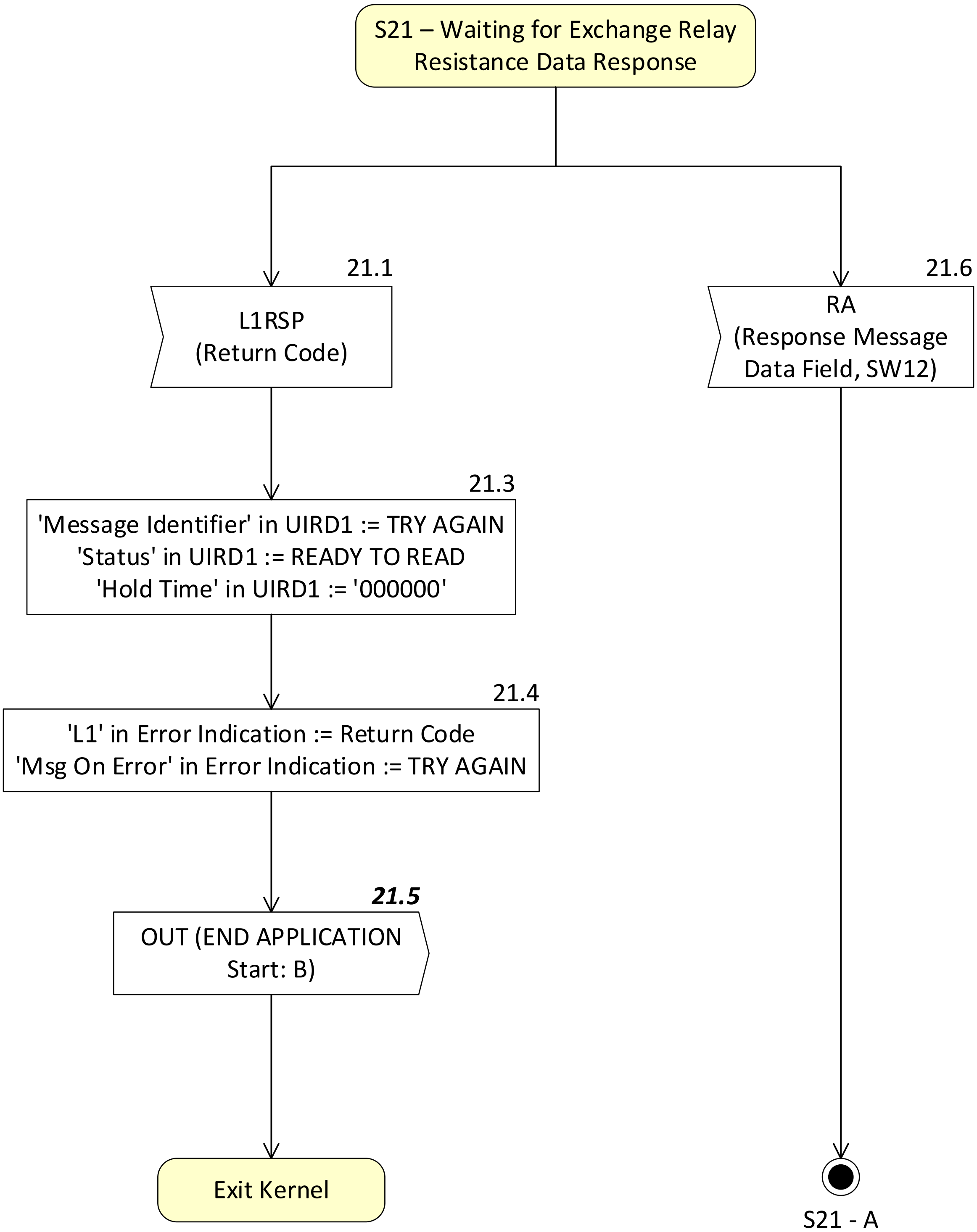
Table 6.4 shows the local variables used in `S21 – Waiting for Exchange Relay Resistance Data Response`.

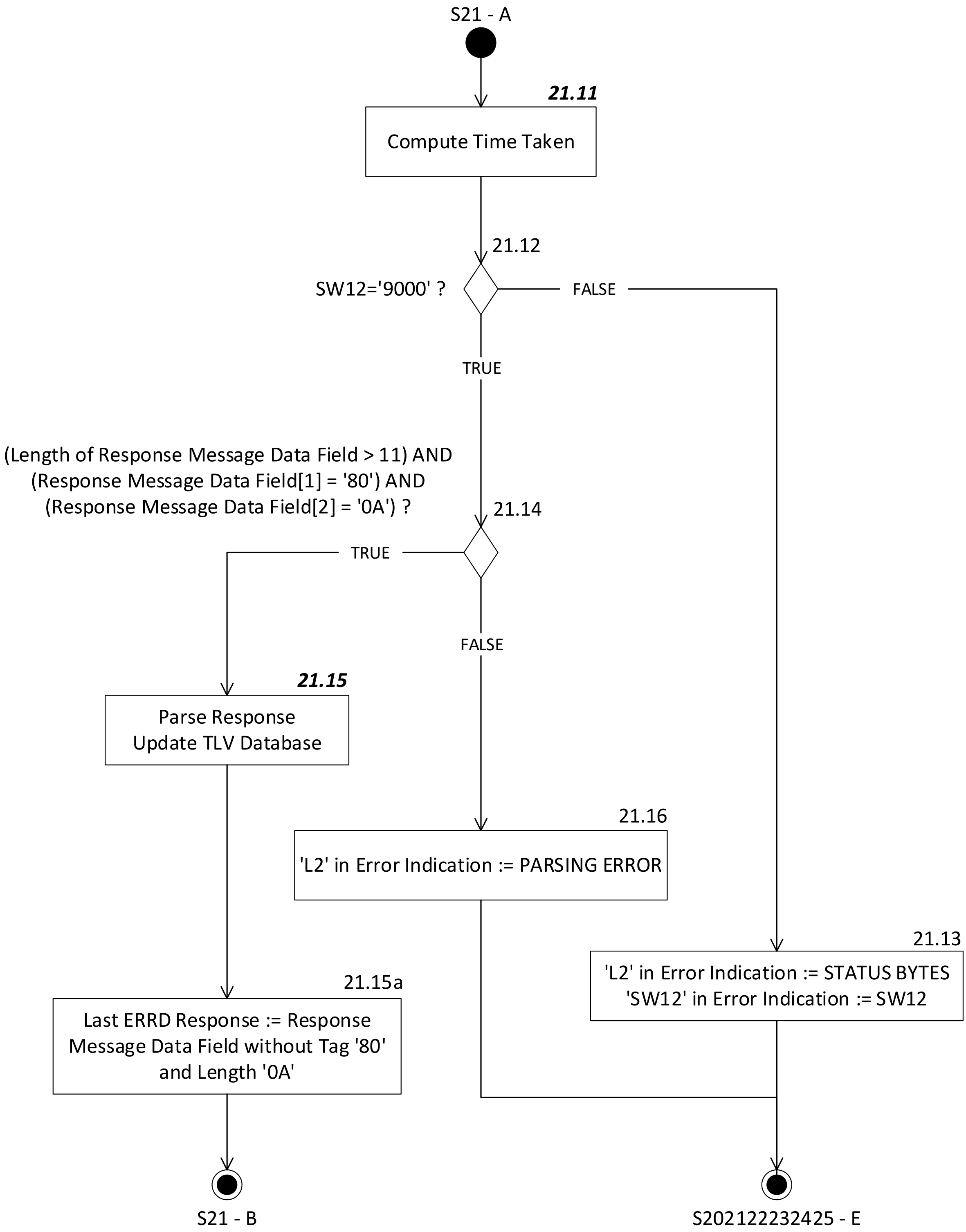
Table 6.4—State 21 Local Variables

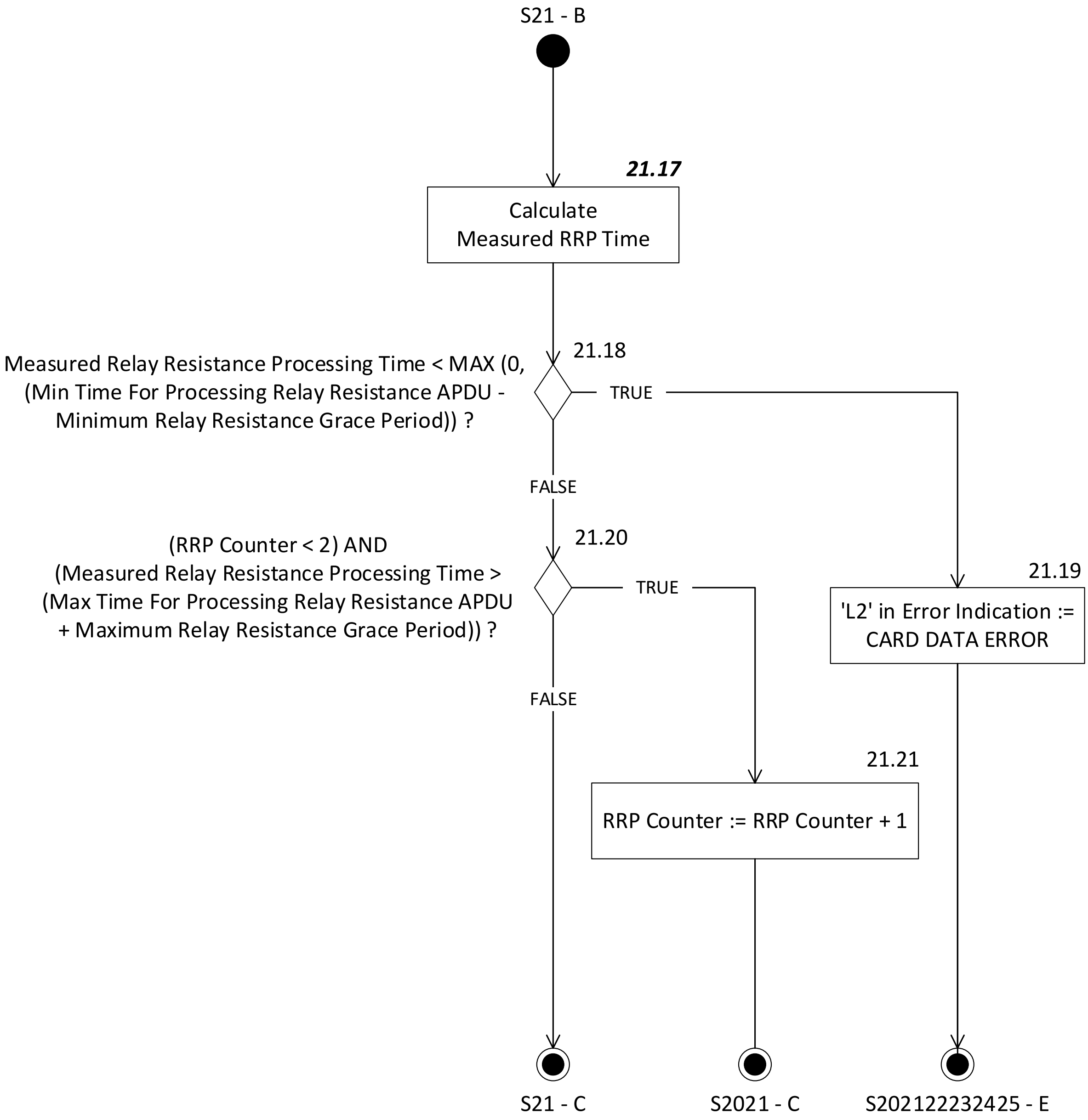
Name	Length	Format	Description
Return Code	1	b	Value returned with L1RSP Signal (TIMEOUT ERROR, PROTOCOL ERROR, TRANSMISSION ERROR)
T	var.	b	Tag of TLV-coded string
SW12	2	b	Status bytes
Response Message Data Field	var. up to 256	b	TLV-coded string included in R-APDU of EXCHANGE RELAY RESISTANCE DATA
Time Taken	4	b	Time to process the EXCHANGE RELAY RESISTANCE DATA command. Time Taken is expressed in microseconds.

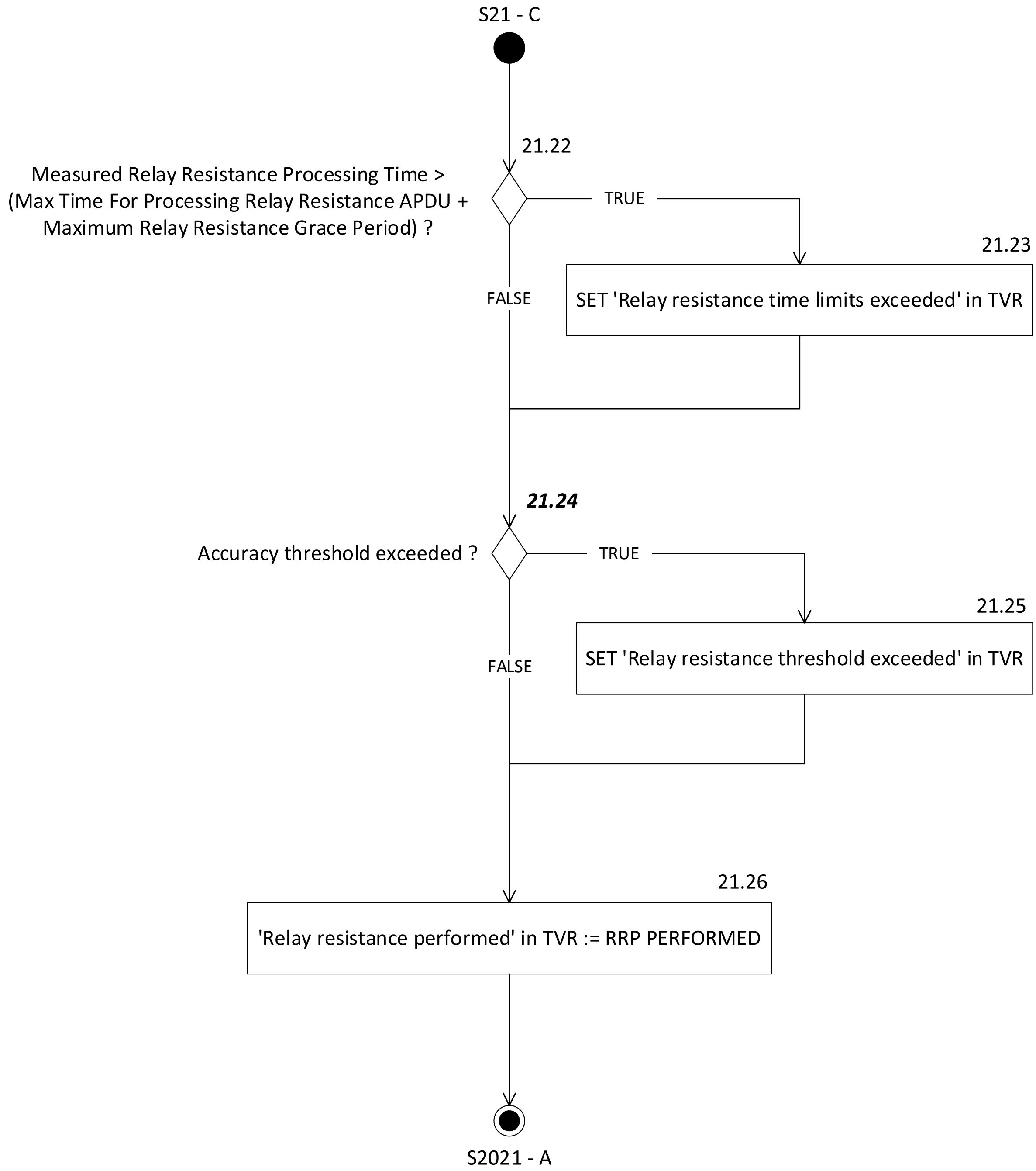
Figure 6.6 shows the flow diagram of `S21 – Waiting for Exchange Relay Resistance Data Response`. Symbols in this diagram are labelled 21.X.

Figure 6.6—State 21 Flow Diagram




S21 - A
21.11
Compute Time Taken
21.12
SW12='9000' ?
FALSE
TRUE
(Length of Response Message Data Field > 11) AND
(Response Message Data Field[1] = '80') AND
(Response Message Data Field[2] = '0A') ?
21.14
TRUE
FALSE
21.15
Parse Response
Update TLV Database
21.16
'L2' in Error Indication := PARSING ERROR
21.13
'L2' in Error Indication := STATUS BYTES
'SW12' in Error Indication := SW12
21.15a
Last ERRD Response := Response Message Data Field without Tag '80' and Length '0A'
S21 - B
S202122232425 - E


S21 - B
21.17
Calculate
Measured RRP Time
21.18
Measured Relay Resistance Processing Time < MAX (0,
(Min Time For Processing Relay Resistance APDU -
Minimum Relay Resistance Grace Period)) ?
TRUE
FALSE
21.20
(RRP Counter < 2) AND
(Measured Relay Resistance Processing Time >
(Max Time For Processing Relay Resistance APDU
+ Maximum Relay Resistance Grace Period)) ?
TRUE
FALSE
21.19
'L2' in Error Indication :=
CARD DATA ERROR
21.21
RRP Counter := RRP Counter + 1
S21 - C
S2021 - C
S202122232425 - E


S21 - C
21.22
Measured Relay Resistance Processing Time > (Max Time For Processing Relay Resistance APDU + Maximum Relay Resistance Grace Period) ?
TRUE
21.23
SET 'Relay resistance time limits exceeded' in TVR
FALSE
21.24
Accuracy threshold exceeded ?
TRUE
21.25
SET 'Relay resistance threshold exceeded' in TVR
FALSE
21.26
'Relay resistance performed' in TVR := RRP PERFORMED
S2021 - A

21.5

'Status' in Outcome Parameter Set := END APPLICATION

'Start' in Outcome Parameter Set := B

SET 'UI Request on Restart Present' in Outcome Parameter Set

CreateDiscretionaryData ()

Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
GetTLV(TagOf(Discretionary Data)),
GetTLV(TagOf(User Interface Request Data 1))) Signal

21.11

Compute Time Taken as the difference between current time in microseconds and Start Time.

21.15

Device Relay Resistance Entropy := Response Message Data Field[3:6]

Min Time For Processing Relay Resistance APDU := Response Message Data Field[7:8]

Max Time For Processing Relay Resistance APDU := Response Message Data Field[9:10]

Device Estimated Transmission Time For Relay Resistance R-APDU := Response Message Data Field[11:12]

21.17

Measured Relay Resistance Processing Time := MAX (0, (Time Taken div 100) – Terminal Expected Transmission Time For Relay Resistance C-APDU – MIN (Device Estimated Transmission Time For Relay Resistance R-APDU, Terminal Expected Transmission Time For Relay Resistance R-APDU))

Relay Resistance Time Excess := MAX (0, Measured Relay Resistance Processing Time – Max Time For Processing Relay Resistance APDU)

Note:

The implementation should compensate for any known fixed timing latency. All implementations will have some inevitable delay between starting the timer and sending the C-APDU and between receiving the R-APDU and stopping the timer. If this latency is predictable and can be compensated for by the implementation then it does not need to be compensated by increasing the maximum grace period.

21.24

Accuracy threshold exceeded ?:

[(Device Estimated Transmission Time For Relay Resistance R-APDU ≠ 0)
AND

(Terminal Expected Transmission Time For Relay Resistance R-APDU ≠ 0)
AND

((((Device Estimated Transmission Time For Relay Resistance R-APDU * 100) div Terminal Expected Transmission Time For Relay Resistance R-APDU) < Relay Resistance Transmission Time Mismatch Threshold)
OR

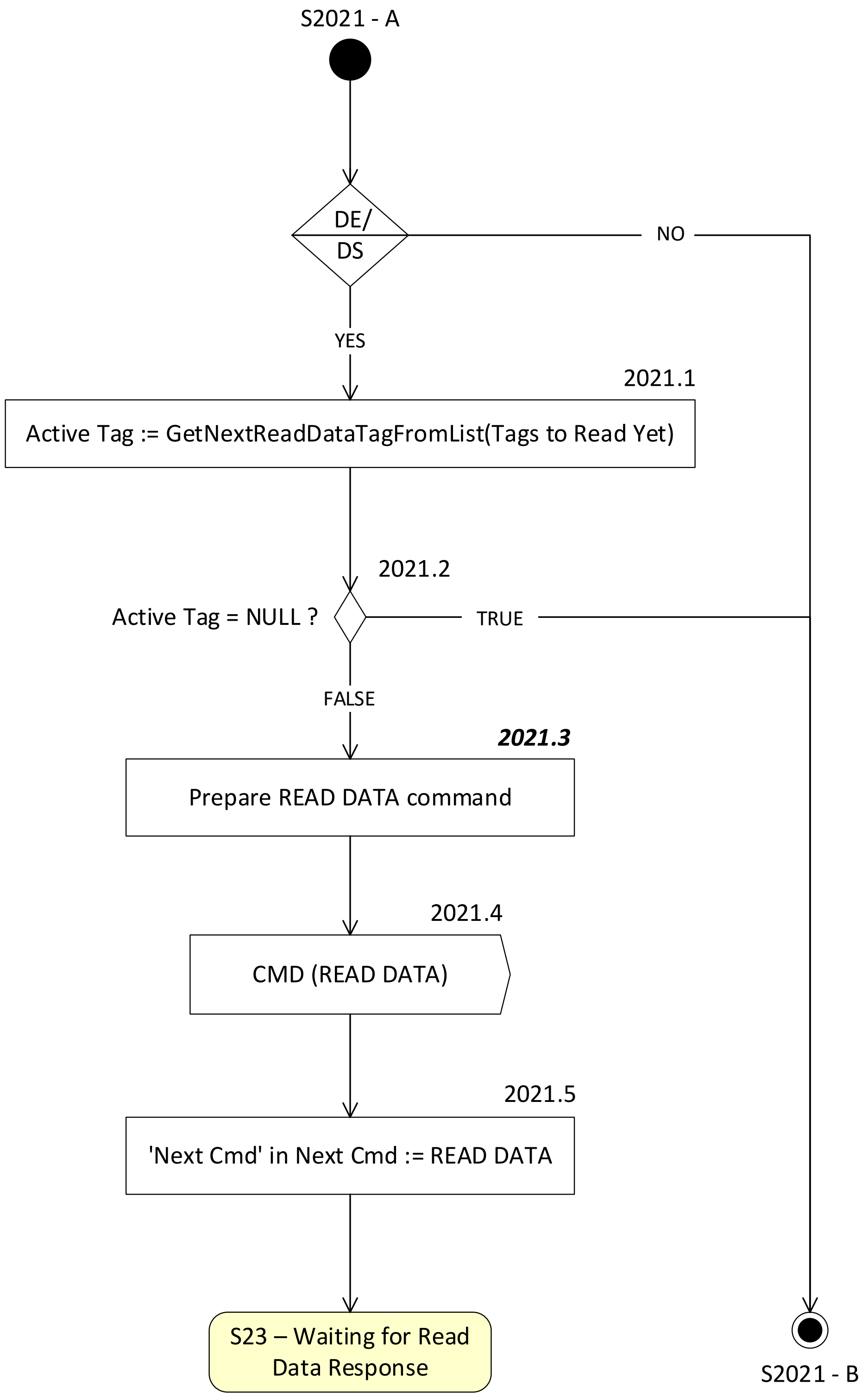
(((Terminal Expected Transmission Time For Relay Resistance R-APDU * 100) div Device Estimated Transmission Time For Relay Resistance R-APDU) < Relay Resistance Transmission Time Mismatch Threshold)
OR

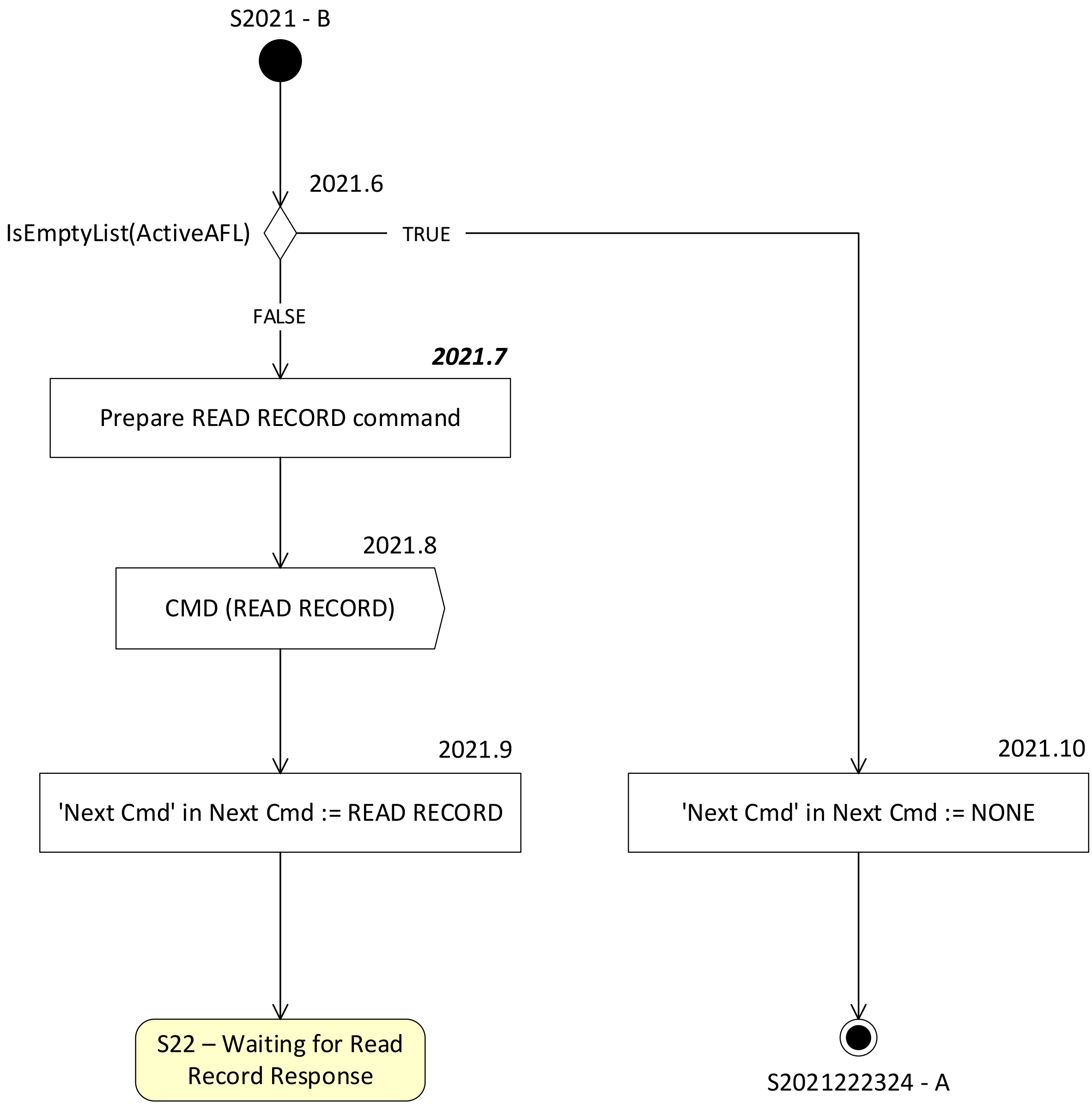
(MAX (0, (Measured Relay Resistance Processing Time – Min Time For Processing Relay Resistance APDU)) > Relay Resistance Accuracy Threshold))]

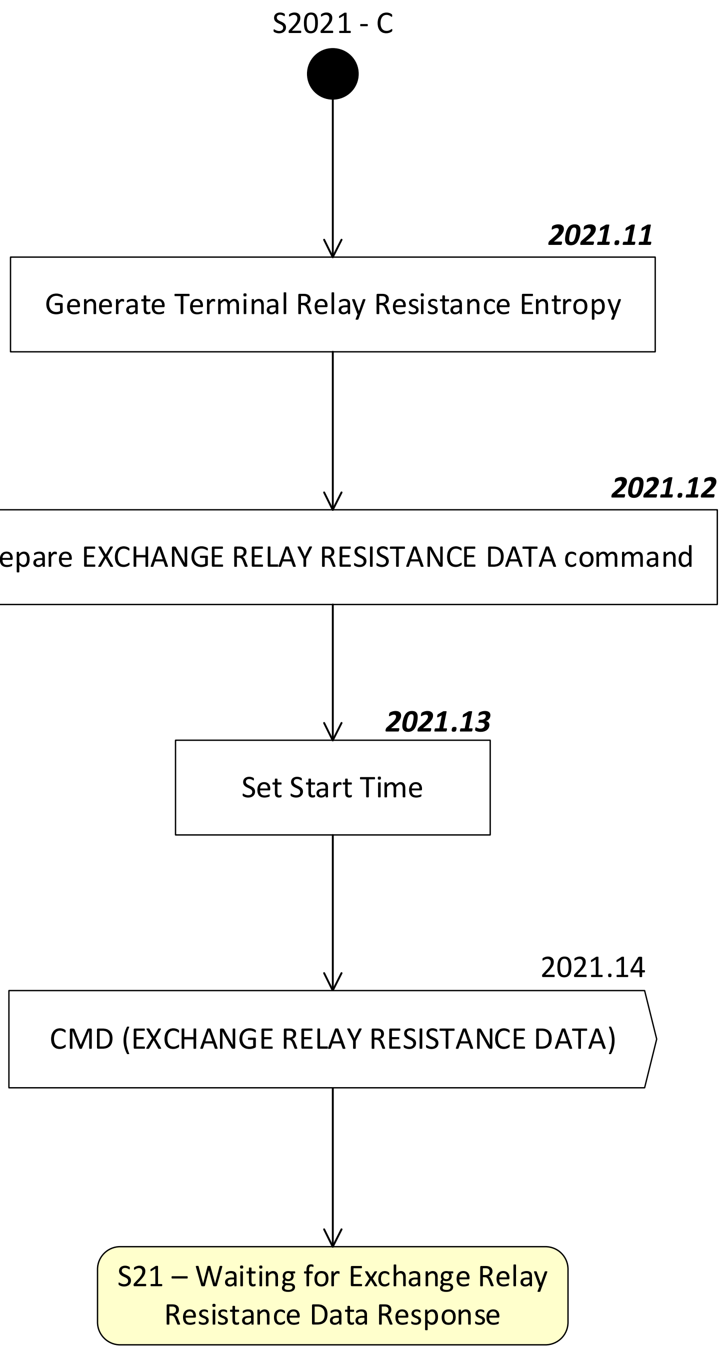
6.3.6 States 20 and 21 – Common Processing

Figure 6.7 shows the flow diagram for common processing between states 20 and 21. Symbols in this diagram are labelled 2021.X.

Figure 6.7—States 20 and 21 – Common Processing – Flow Diagram




S2021 - B
2021.6
IsEmptyList(ActiveAFL)
TRUE
FALSE
2021.7
Prepare READ RECORD command
2021.8
CMD (READ RECORD)
2021.9
'Next Cmd' in Next Cmd := READ RECORD
2021.10
'Next Cmd' in Next Cmd := NONE
S22 – Waiting for Read Record Response
S2021222324 - A


S2021 - C
2021.11
Generate Terminal Relay Resistance Entropy
2021.12
Prepare EXCHANGE RELAY RESISTANCE DATA command
2021.13
Set Start Time
2021.14
CMD (EXCHANGE RELAY RESISTANCE DATA)
S21 – Waiting for Exchange Relay Resistance Data Response

Note that the symbols 2021.1, 2021.2, 2021.3, 2021.4 and 2021.5 are only implemented for the DE/DS Implementation Option.

2021.3

Build READ DATA command for Active Tag as defined in section 5.5

AddToList(Active Tag, Read Data Tags To Validate Yet)

2021.7

Build READ RECORD command for the first entry in the Active AFL as defined in section 5.6.

2021.11

Generate unpredictable number as specified in [EMV Book E] section 8.1 and store in Unpredictable Number.

Terminal Relay Resistance Entropy := Unpredictable Number

2021.12

Prepare EXCHANGE RELAY RESISTANCE DATA command as specified in section 5.2

2021.13

Store current time in microseconds in Start Time

6.3.7 State 22 – Waiting for Read Record Response

Table 6.5 shows the local variables used in `S22 – Waiting for Read Record Response`.

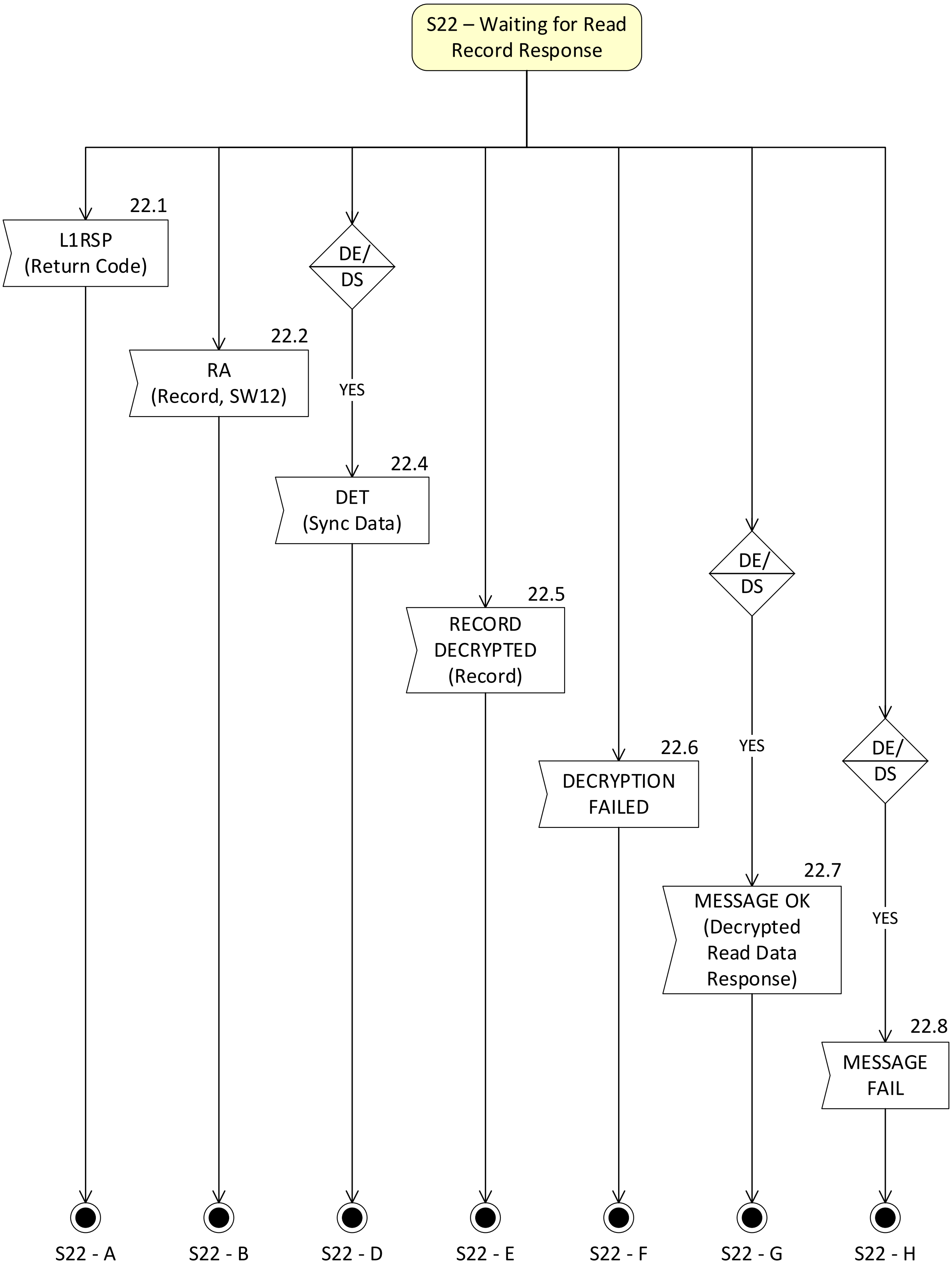
Table 6.5—State 22 Local Variables

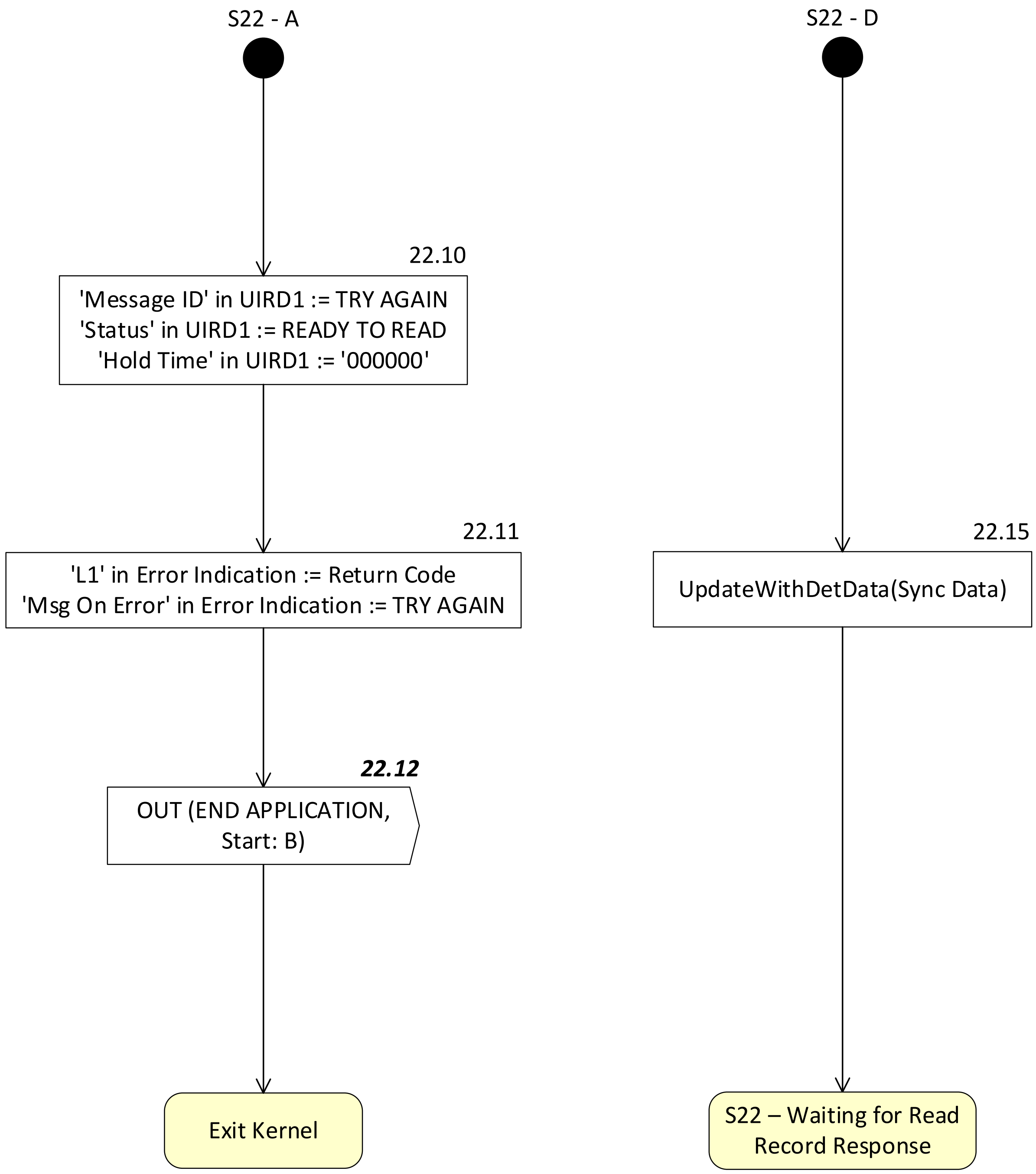
Name	Length	Format	Description
Return Code	1	b	Value returned with L1RSP Signal (TIMEOUT ERROR, PROTOCOL ERROR, TRANSMISSION ERROR)
Sync Data [8]	var.	b	List of data objects returned with DET Signal
Parsing Result	1	b	Boolean used to store result of parsing a TLV-coded string
P2	1	b	P2 parameter of GENERATE AC command
SW12	2	b	Status bytes
Record	var. up to 254	b	Response Message Data Field of the R-APDU of READ RECORD or decrypted record returned with RECORD DECRYPTED Signal
Decrypted Read Data Response [8]	var. up to 248	b	Decrypted read data response returned with MESSAGE OK Signal.
Signed Flag	1	b	Boolean used to indicate if current record is signed
T	var.	b	Tag of TLV-coded string

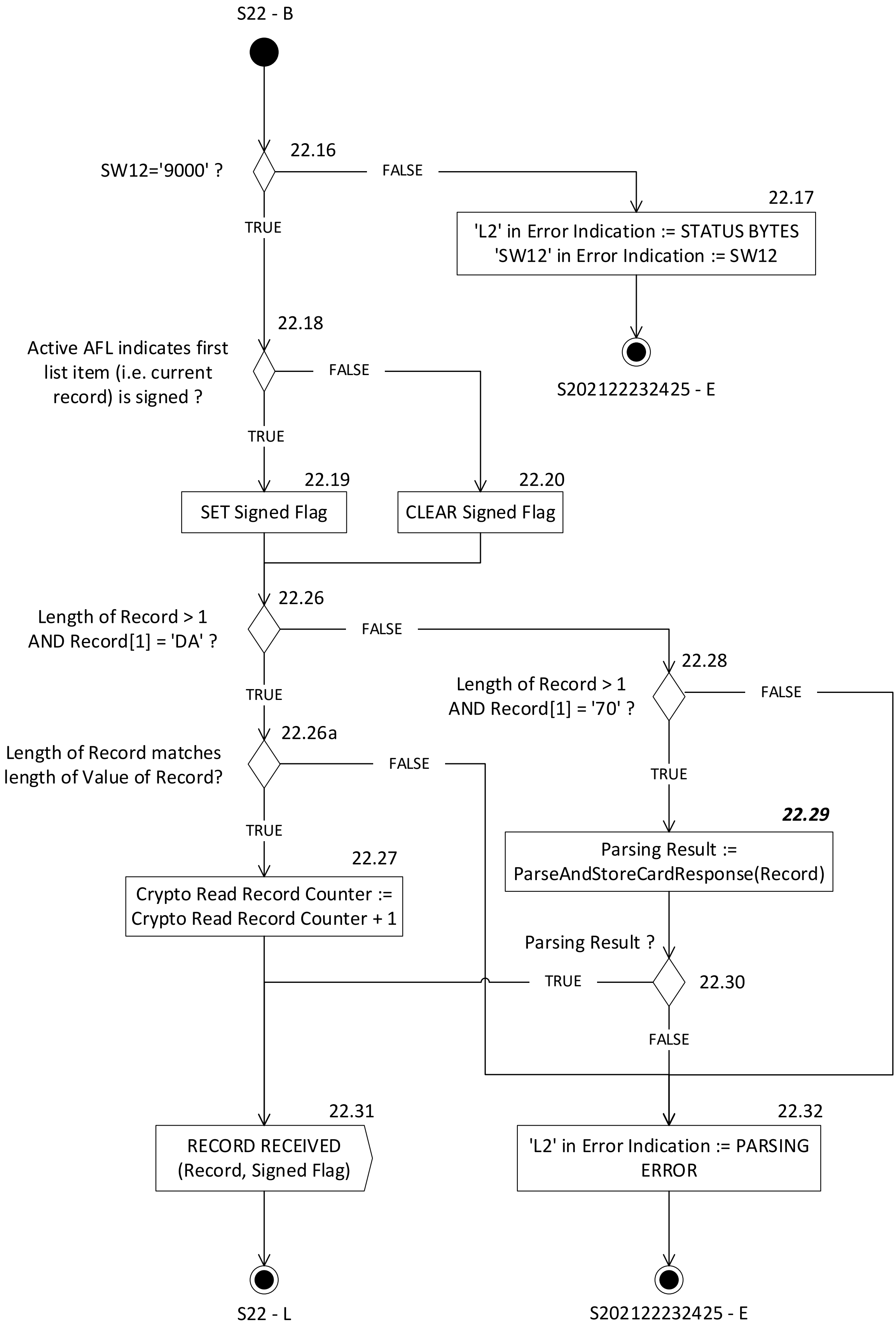
Figure 6.8 shows the flow diagram of `S22 – Waiting for Read Record Response`. Symbols in this diagram are labelled 22.X.

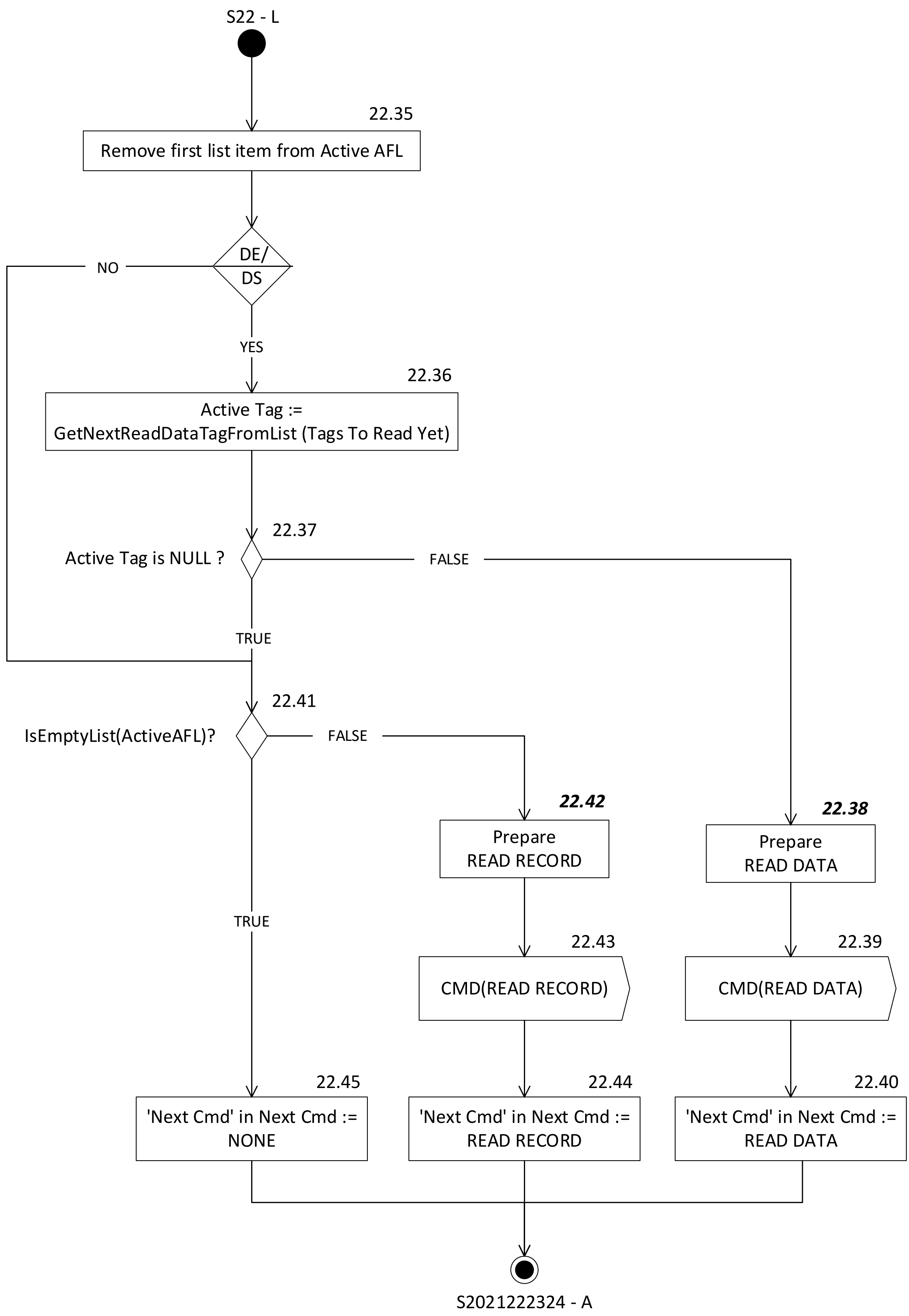
[8] Only implemented for the DE/DS Implementation Option

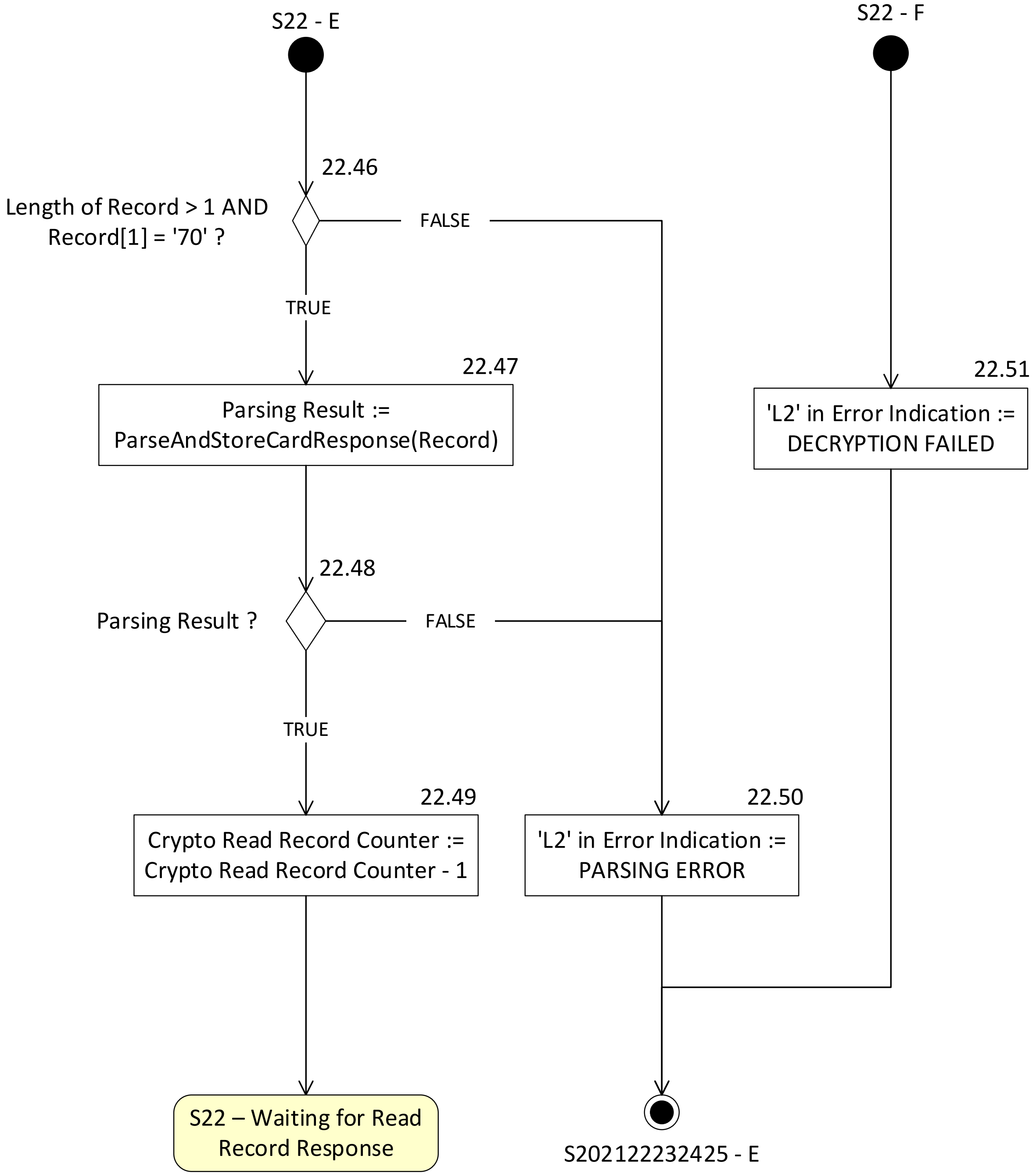
Figure 6.8—State 22 Flow Diagram

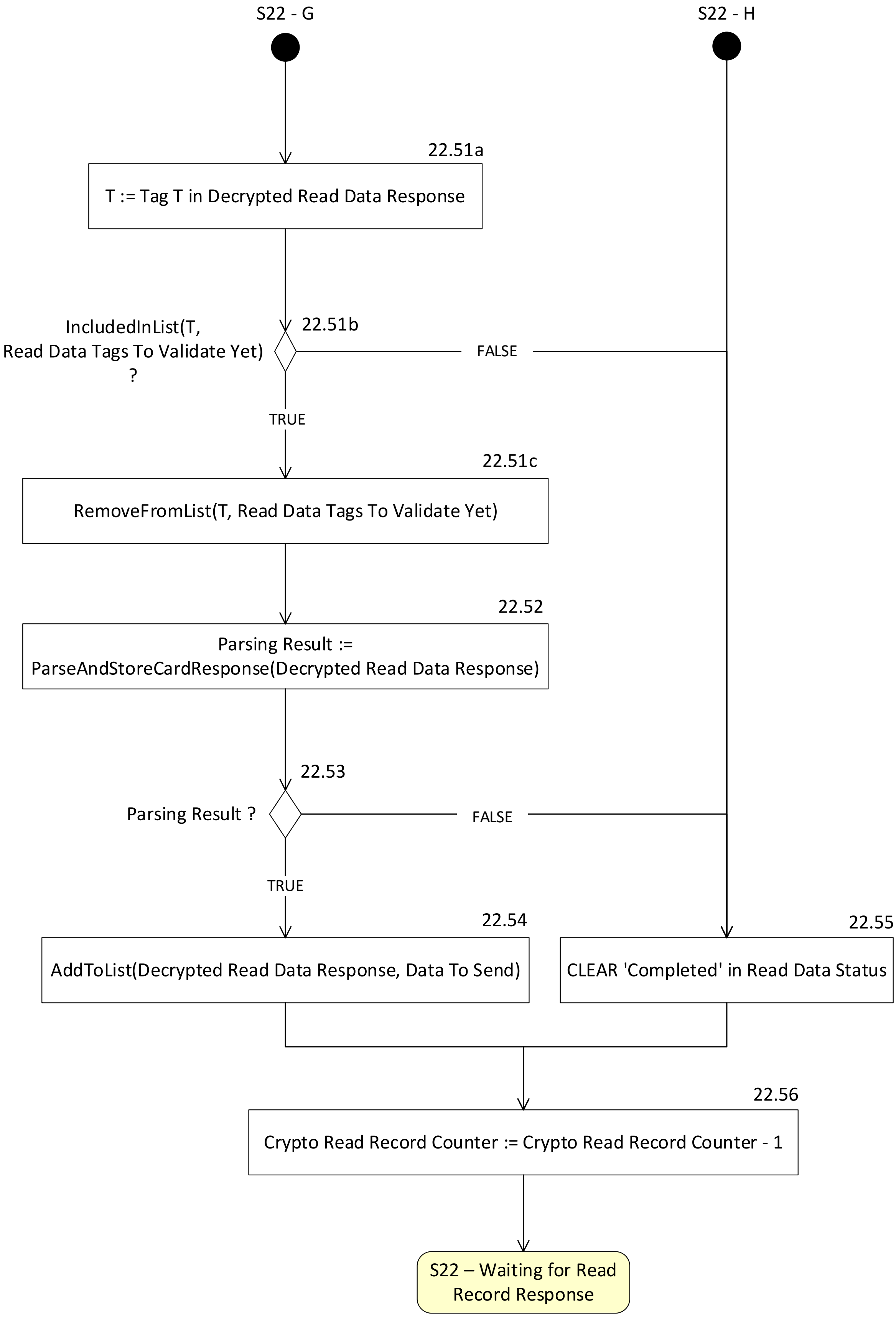



S22 - A
22.10
'Message ID' in UIRD1 := TRY AGAIN
'Status' in UIRD1 := READY TO READ
'Hold Time' in UIRD1 := '000000'
22.11
'L1' in Error Indication := Return Code
'Msg On Error' in Error Indication := TRY AGAIN
22.12
OUT (END APPLICATION, Start: B)
Exit Kernel
S22 - D
22.15
UpdateWithDetData(Sync Data)
S22 – Waiting for Read Record Response


S22 - B
22.16
SW12='9000' ?
FALSE
TRUE
22.17
'L2' in Error Indication := STATUS BYTES
'SW12' in Error Indication := SW12
S202122232425 - E
22.18
Active AFL indicates first list item (i.e. current record) is signed ?
FALSE
TRUE
22.19
SET Signed Flag
22.20
CLEAR Signed Flag
22.26
Length of Record > 1 AND Record[1] = 'DA' ?
FALSE
TRUE
22.28
Length of Record > 1 AND Record[1] = '70' ?
FALSE
TRUE
22.26a
Length of Record matches length of Value of Record?
FALSE
TRUE
22.29
Parsing Result := ParseAndStoreCardResponse(Record)
22.27
Crypto Read Record Counter := Crypto Read Record Counter + 1
Parsing Result ?
22.30
TRUE
FALSE
22.31
RECORD RECEIVED (Record, Signed Flag)
22.32
'L2' in Error Indication := PARSING ERROR
S22 - L
S202122232425 - E


S22 - L
22.35
Remove first list item from Active AFL
DE/
DS
NO
YES
22.36
Active Tag :=
GetNextReadDataTagFromList (Tags To Read Yet)
22.37
Active Tag is NULL ?
FALSE
TRUE
22.41
IsEmptyList(ActiveAFL)?
FALSE
TRUE
22.42
Prepare
READ RECORD
22.38
Prepare
READ DATA
22.43
CMD(READ RECORD)
22.39
CMD(READ DATA)
22.45
'Next Cmd' in Next Cmd :=
NONE
22.44
'Next Cmd' in Next Cmd :=
READ RECORD
22.40
'Next Cmd' in Next Cmd :=
READ DATA
S2021222324 - A


S22 - E
S22 - F
22.46
Length of Record > 1 AND Record[1] = '70' ?
FALSE
TRUE
22.47
Parsing Result := ParseAndStoreCardResponse(Record)
22.51
'L2' in Error Indication := DECRYPTION FAILED
22.48
Parsing Result ?
FALSE
TRUE
22.49
Crypto Read Record Counter := Crypto Read Record Counter - 1
22.50
'L2' in Error Indication := PARSING ERROR
S22 – Waiting for Read Record Response
S202122232425 - E


S22 - G
S22 - H
22.51a
T := Tag T in Decrypted Read Data Response
IncludedInList(T,
Read Data Tags To Validate Yet)
?
22.51b
FALSE
TRUE
22.51c
RemoveFromList(T, Read Data Tags To Validate Yet)
22.52
Parsing Result :=
ParseAndStoreCardResponse(Decrypted Read Data Response)
22.53
Parsing Result ?
FALSE
TRUE
22.54
AddToList(Decrypted Read Data Response, Data To Send)
22.55
CLEAR 'Completed' in Read Data Status
22.56
Crypto Read Record Counter := Crypto Read Record Counter - 1
S22 – Waiting for Read
Record Response

Note that the symbols 22.4, 22.7, 22.8, 22.15, 22.36, 22.37, 22.38, 22.39, 22.40, 22.51a, 22.51b, 22.51c, 22.52, 22.53, 22.54, 22.55 and 22.56 are only implemented for the DE/DS Implementation Option.

22.12

'Status' in Outcome Parameter Set := END APPLICATION

'Start' in Outcome Parameter Set := B

SET 'UI Request on Restart Present' in Outcome Parameter Set

CreateDiscretionaryData ()

```
Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
         GetTLV(TagOf(Discretionary Data)),
         GetTLV(TagOf(User Interface Request Data 1))) Signal
```

22.29

Note that when DE/DS is not implemented, the implementation may postpone the parsing of the Record and the corresponding update of the TLV Database until after the GENERATE AC command, but no later than S2627 – A before symbol S2627.1.

22.38

Prepare READ DATA command for Active Tag as specified in section 5.5

AddToList(Active Tag, Read Data Tags To Validate Yet)

22.42

Prepare READ RECORD command for first list item in Active AFL as specified in section 5.6

6.3.8 State 23 – Waiting for Read Data Response

State 23 is a state specific to data storage processing and is only implemented if DE/DS Implementation Option is used.

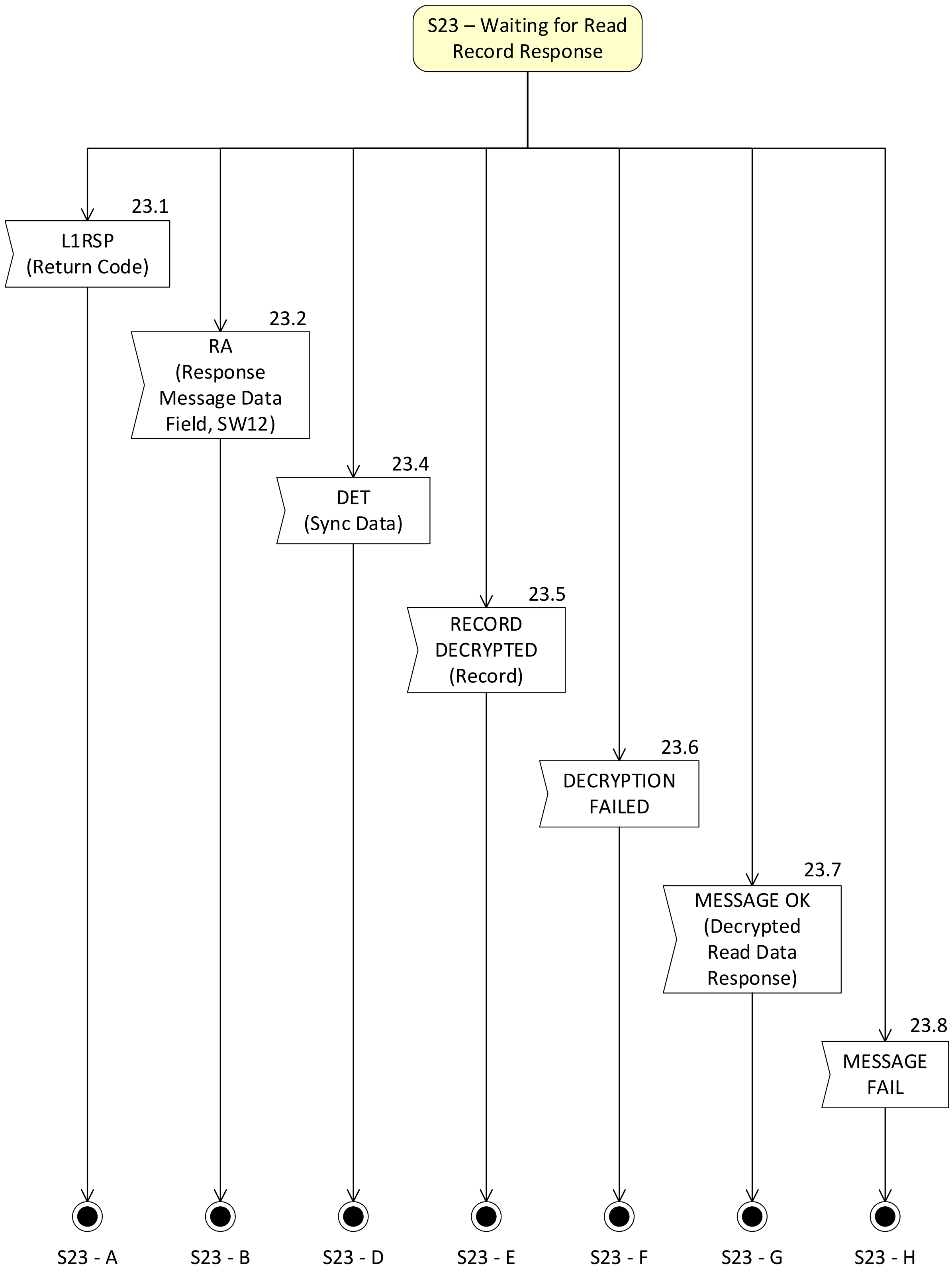
Table 6.6 shows the local variables used in `S23 – Waiting for Read Data Response`.

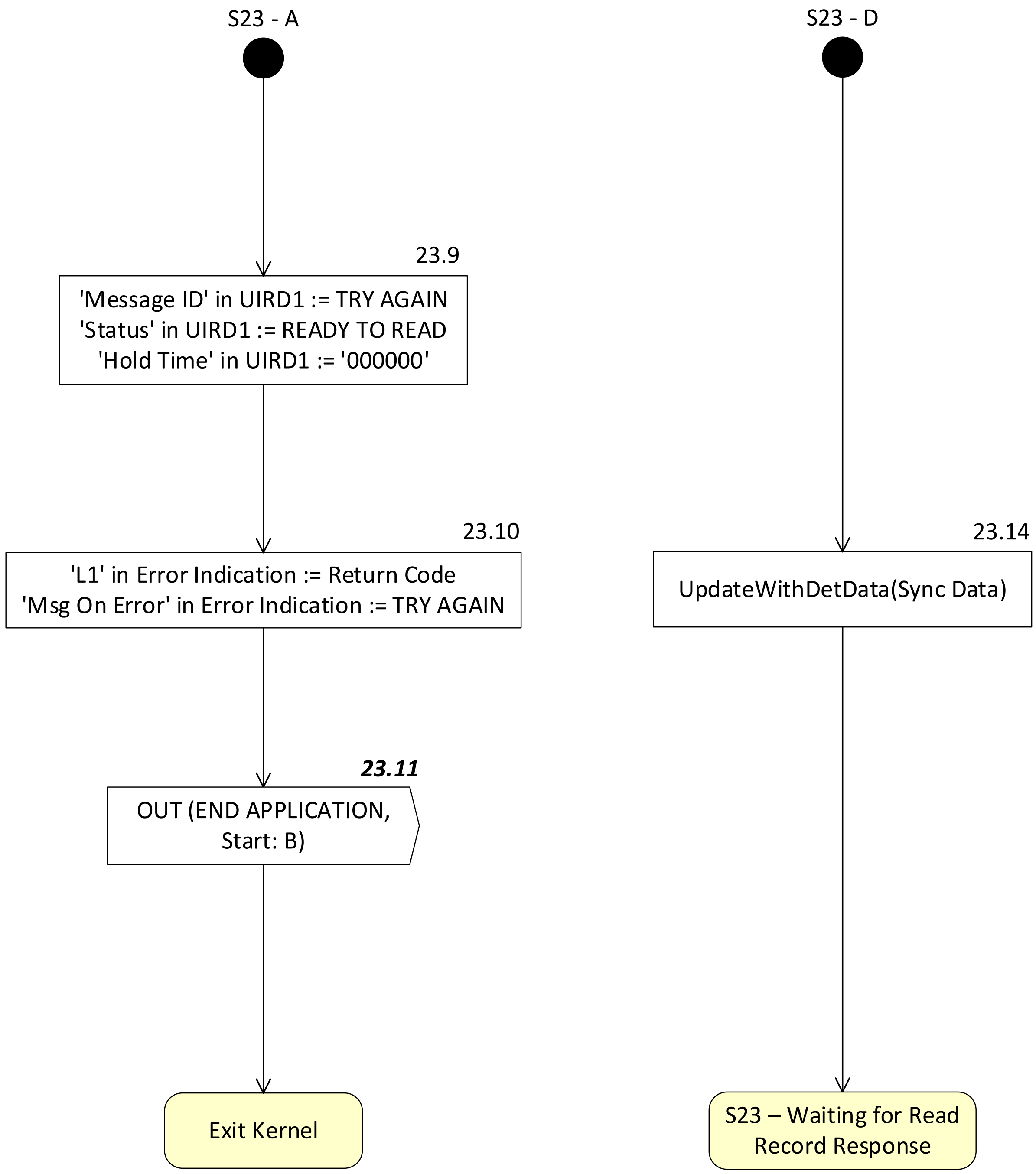
Table 6.6—State 23 Local Variables

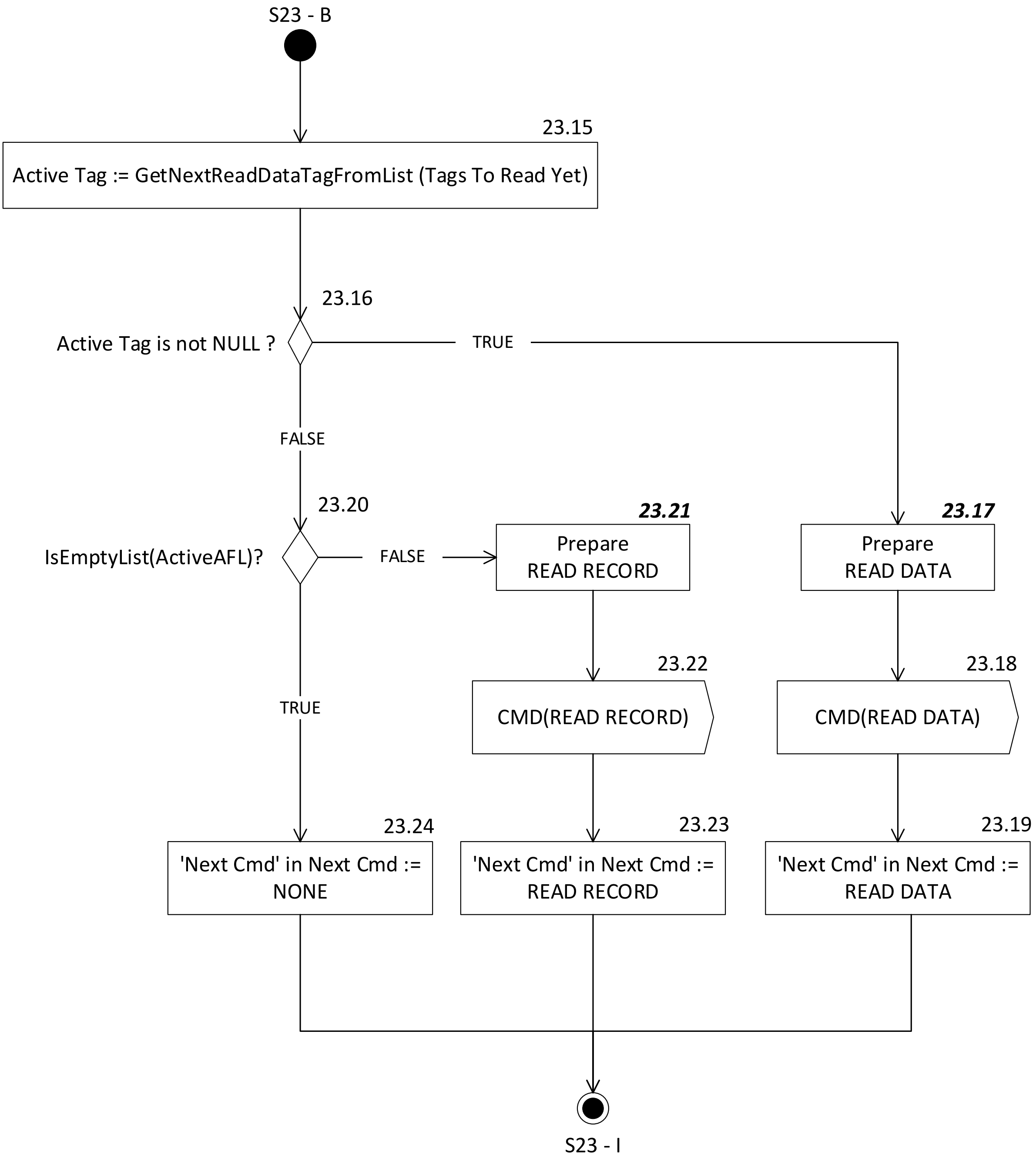
Name	Length	Format	Description
Return Code	1	b	Value returned with L1RSP Signal (TIMEOUT ERROR, PROTOCOL ERROR, TRANSMISSION ERROR)
Sync Data	var.	b	List of data objects returned with DET Signal
Parsing Result	1	b	Boolean used to store result of parsing a TLV-coded string
P2	1	b	P2 parameter of GENERATE AC command
SW12	2	b	Status bytes
Record	var. up to 254	b	Decrypted record returned in RECORD DECRYPTED Signal
Decrypted Read Data Response	var. up to 248	b	Decrypted read data response returned with MESSAGE OK Signal.
Response Message Data Field	var. up to 256	b	TLV-coded string included in R-APDU of READ DATA
T	var.	b	Tag of TLV-coded string

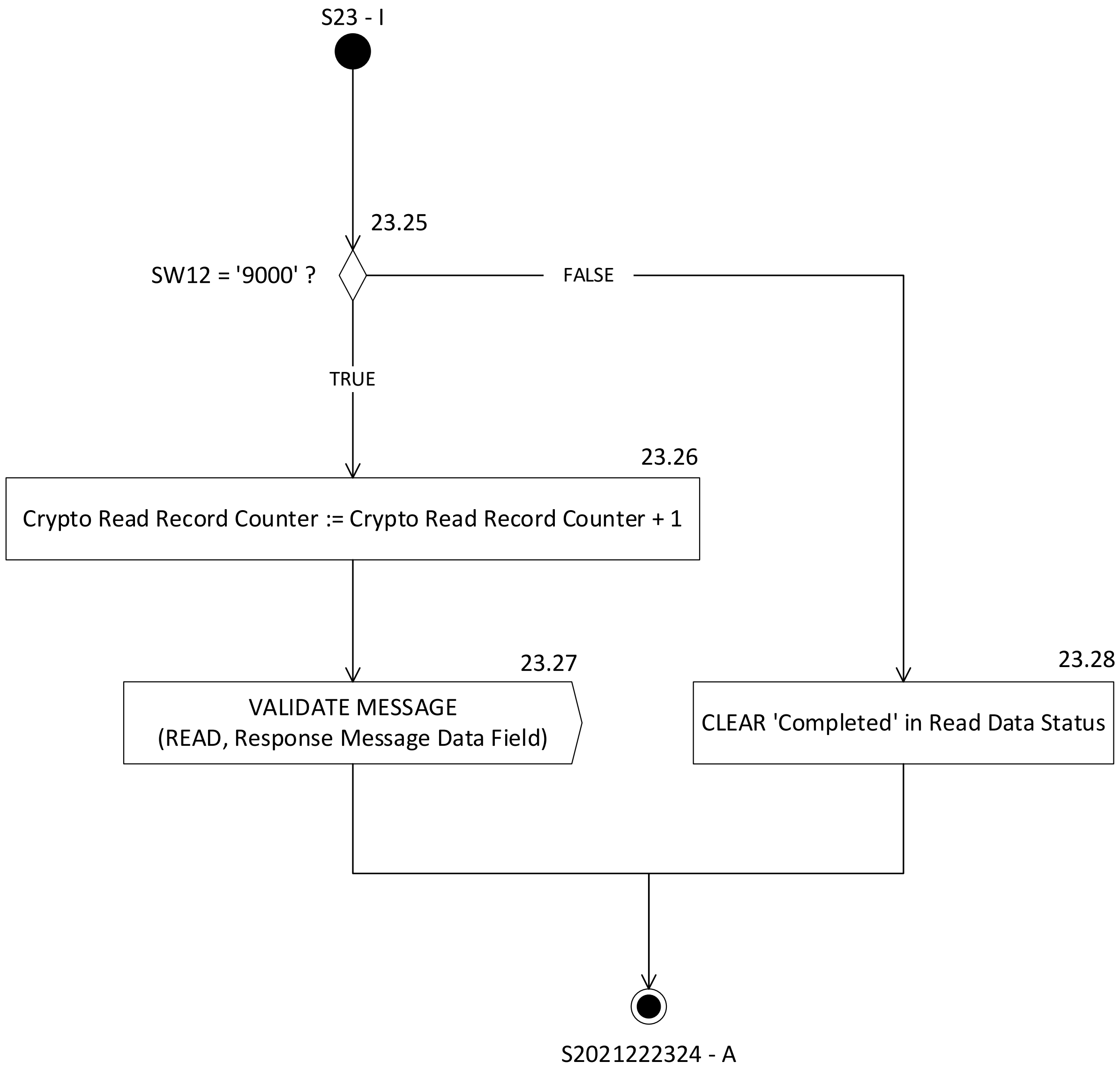
Figure 6.9 shows the flow diagram of `S23 – Waiting for Read Data Response`. Symbols in this diagram are labelled 23.X.

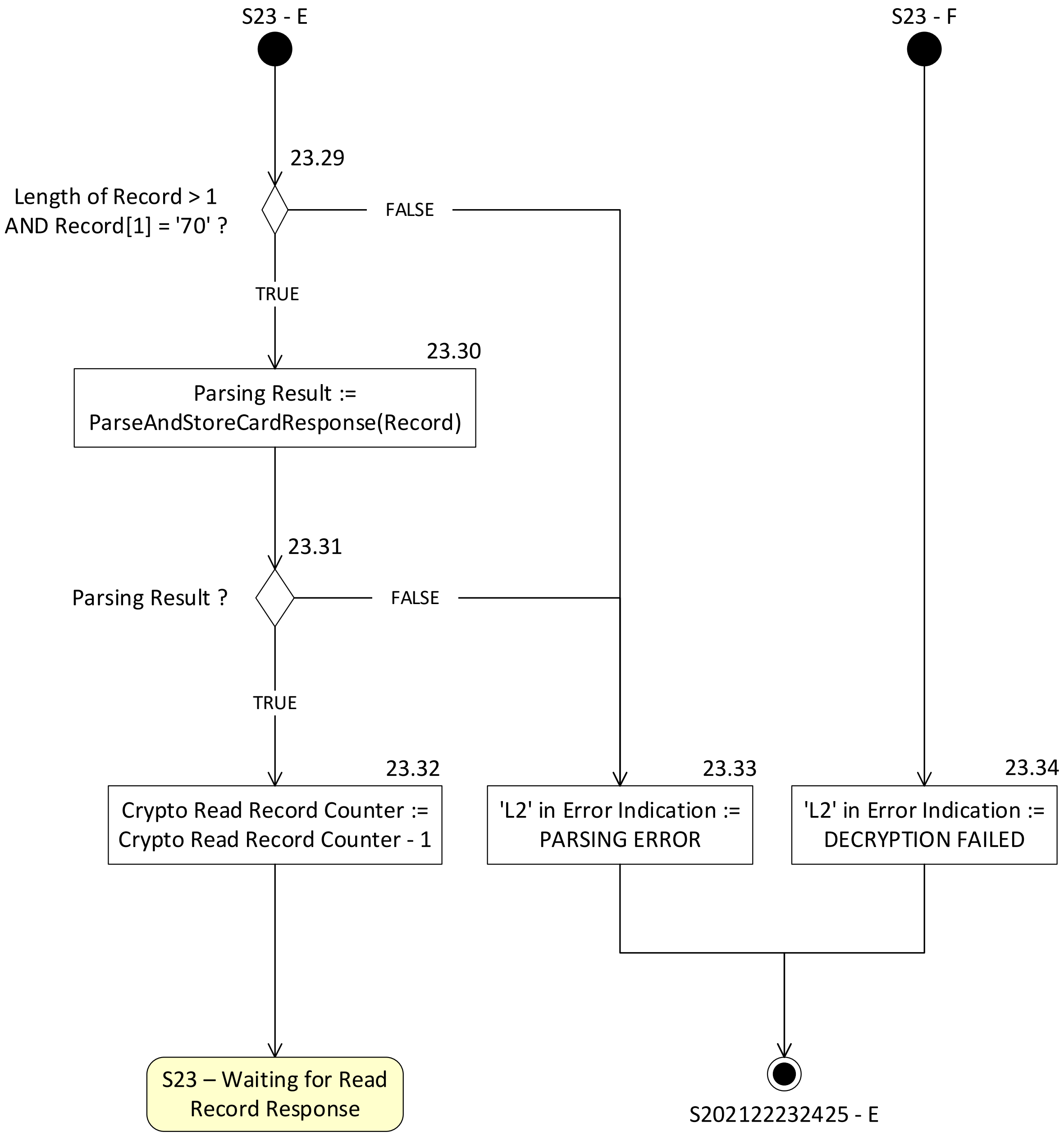
Figure 6.9—State 23 Flow Diagram

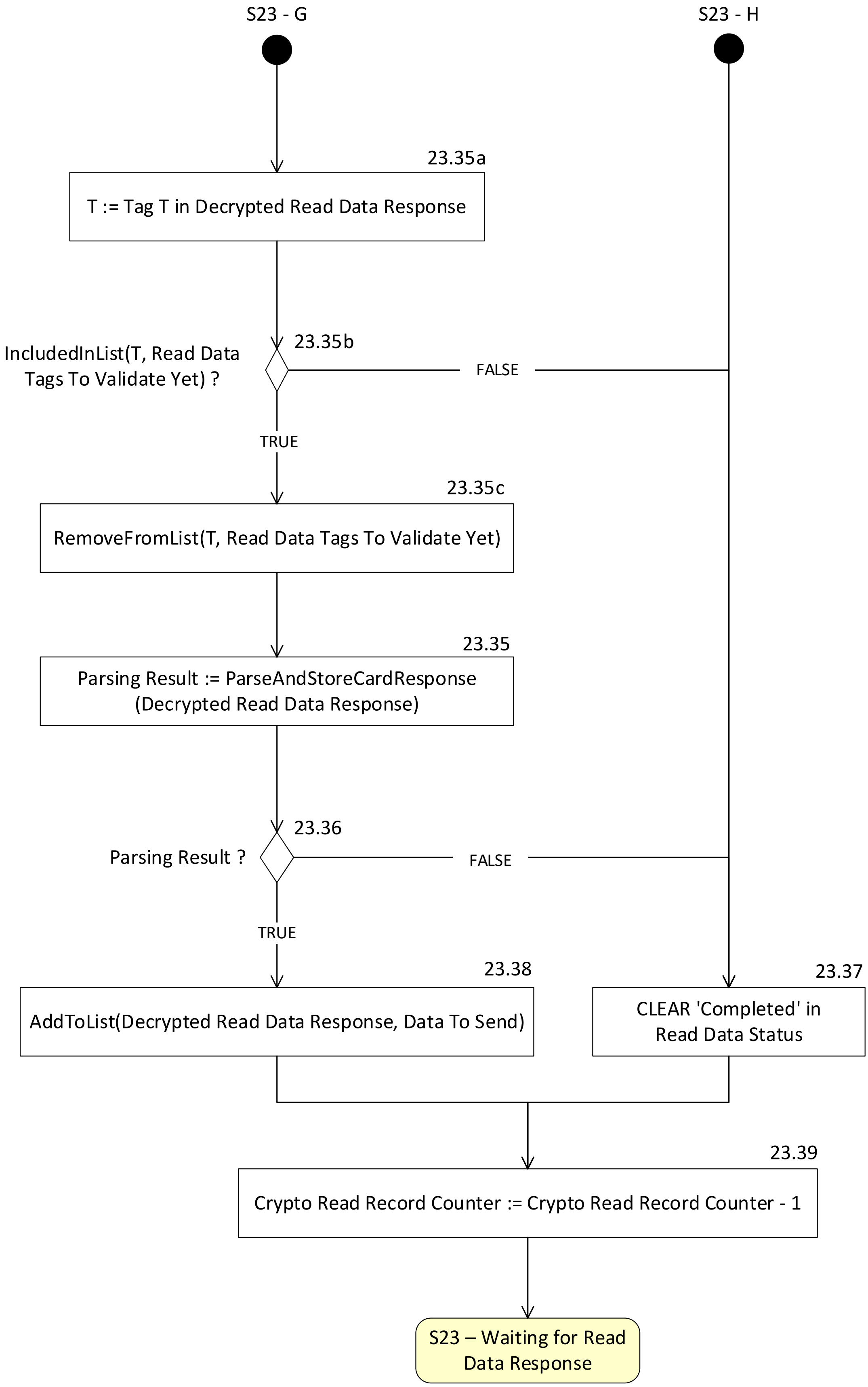



S23 - A
S23 - D
23.9
'Message ID' in UIRD1 := TRY AGAIN
'Status' in UIRD1 := READY TO READ
'Hold Time' in UIRD1 := '000000'
23.10
'L1' in Error Indication := Return Code
'Msg On Error' in Error Indication := TRY AGAIN
23.14
UpdateWithDetData(Sync Data)
23.11
OUT (END APPLICATION, Start: B)
Exit Kernel
S23 – Waiting for Read Record Response


S23 - B
23.15
Active Tag := GetNextReadDataTagFromList (Tags To Read Yet)
23.16
Active Tag is not NULL ?
TRUE
FALSE
23.20
IsEmptyList(ActiveAFL)?
FALSE
TRUE
23.21
Prepare
READ RECORD
23.17
Prepare
READ DATA
23.22
CMD(READ RECORD)
23.18
CMD(READ DATA)
23.24
'Next Cmd' in Next Cmd :=
NONE
23.23
'Next Cmd' in Next Cmd :=
READ RECORD
23.19
'Next Cmd' in Next Cmd :=
READ DATA
S23 - I


S23 - I
23.25
SW12 = '9000' ?
FALSE
TRUE
23.26
Crypto Read Record Counter := Crypto Read Record Counter + 1
23.27
VALIDATE MESSAGE
(READ, Response Message Data Field)
23.28
CLEAR 'Completed' in Read Data Status
S2021222324 - A


S23 - E
S23 - F
23.29
Length of Record > 1
AND Record[1] = '70' ?
FALSE
TRUE
23.30
Parsing Result :=
ParseAndStoreCardResponse(Record)
23.31
Parsing Result ?
FALSE
TRUE
23.32
Crypto Read Record Counter :=
Crypto Read Record Counter - 1
23.33
'L2' in Error Indication :=
PARSING ERROR
23.34
'L2' in Error Indication :=
DECRYPTION FAILED
S23 – Waiting for Read
Record Response
S202122232425 - E


S23 - G
S23 - H
23.35a
T := Tag T in Decrypted Read Data Response
23.35b
IncludedInList(T, Read Data Tags To Validate Yet) ?
FALSE
TRUE
23.35c
RemoveFromList(T, Read Data Tags To Validate Yet)
23.35
Parsing Result := ParseAndStoreCardResponse (Decrypted Read Data Response)
23.36
Parsing Result ?
FALSE
TRUE
23.38
AddToList(Decrypted Read Data Response, Data To Send)
23.37
CLEAR 'Completed' in Read Data Status
23.39
Crypto Read Record Counter := Crypto Read Record Counter - 1
S23 – Waiting for Read Data Response

23.11

'Status' in Outcome Parameter Set := END APPLICATION

'Start' in Outcome Parameter Set := B

SET 'UI Request on Restart Present' in Outcome Parameter Set

CreateDiscretionaryData ()

Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
    GetTLV(TagOf(Discretionary Data)),
    GetTLV(TagOf(User Interface Request Data 1))) Signal

23.17

Prepare READ DATA command for Active Tag as specified in section 5.5

AddToList(Active Tag, Read Data Tags To Validate Yet)

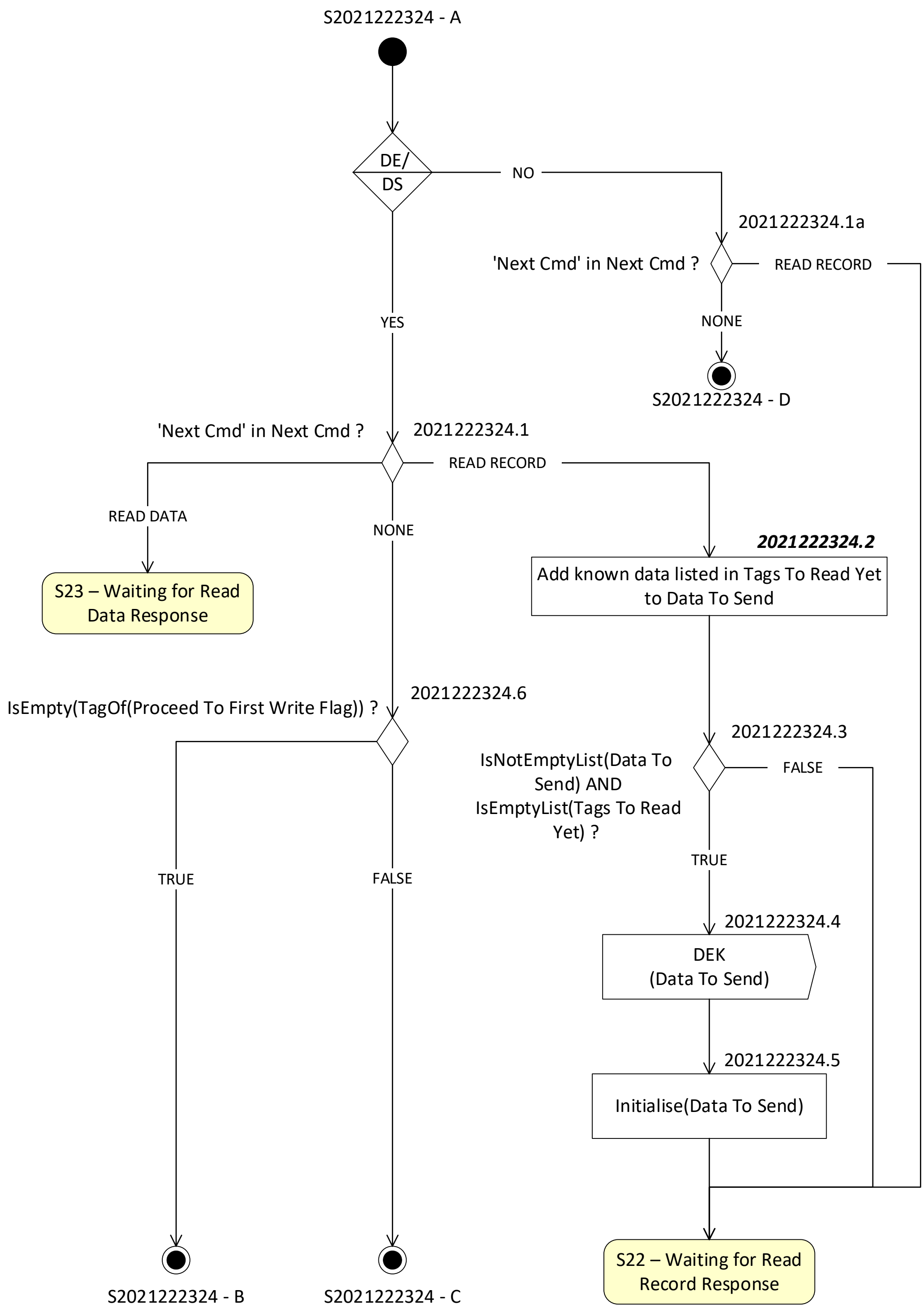
23.21

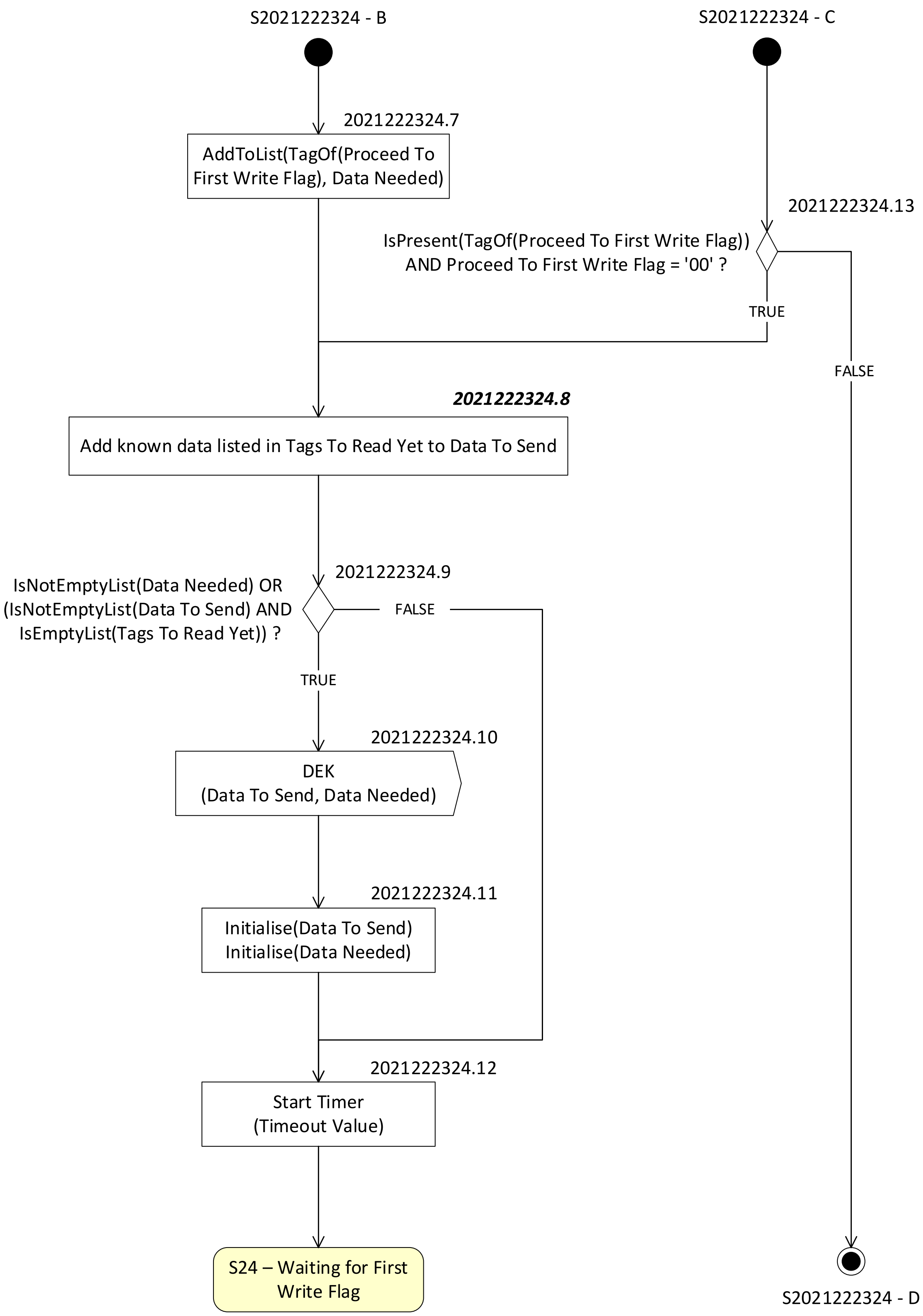
Prepare READ RECORD command for first list item in Active AFL as specified in section 5.6

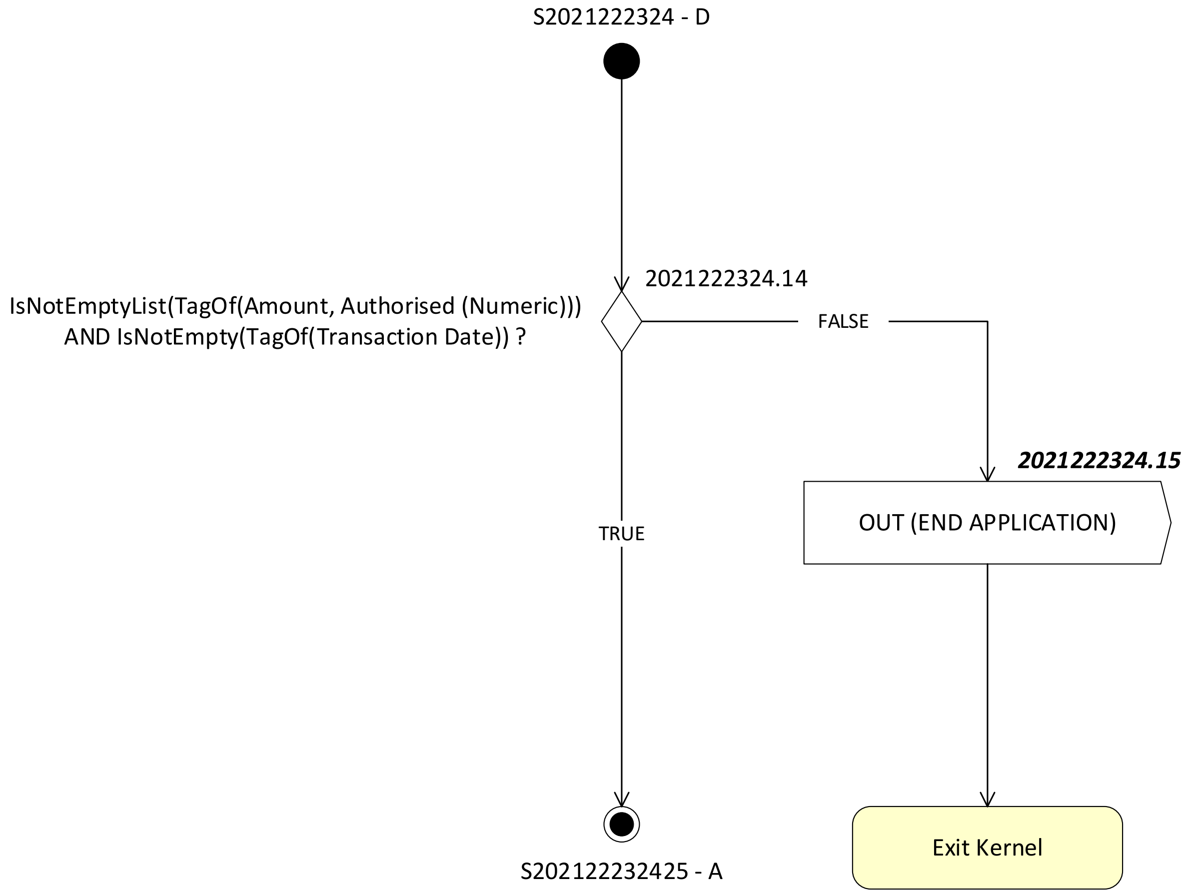
6.3.9 States 20, 21, 22, 23, and 24 – Common Processing

Figure 6.10 shows the flow diagram for common processing between states 20, 21, 22, 23, and 24. Symbols in this diagram are labelled 2021222324.X.

Figure 6.10—States 20, 21, 22, 23, and 24 – Common Processing – Flow Diagram




S2021222324 - B
S2021222324 - C
2021222324.7
AddToList(TagOf(Proceed To First Write Flag), Data Needed)
2021222324.13
IsPresent(TagOf(Proceed To First Write Flag)) AND Proceed To First Write Flag = '00' ?
TRUE
FALSE
2021222324.8
Add known data listed in Tags To Read Yet to Data To Send
2021222324.9
IsNotEmptyList(Data Needed) OR (IsNotEmptyList(Data To Send) AND IsEmptyList(Tags To Read Yet)) ?
FALSE
TRUE
2021222324.10
DEK (Data To Send, Data Needed)
2021222324.11
Initialise(Data To Send)
Initialise(Data Needed)
2021222324.12
Start Timer (Timeout Value)
S24 – Waiting for First Write Flag
S2021222324 - D


S2021222324 - D
2021222324.14
IsNotEmptyList(TagOf(Amount, Authorised (Numeric)))
AND IsNotEmpty(TagOf(Transaction Date)) ?
FALSE
2021222324.15
OUT (END APPLICATION)
TRUE
S202122232425 - A
Exit Kernel

Note that the symbols 2021222324.1, 2021222324.2, 2021222324.3, 2021222324.4, 2021222324.5, 2021222324.6, 2021222324.7, 2021222324.8, 2021222324.9, 2021222324.10, 2021222324.11, 2021222324.12, and 2021222324.13 are only implemented for the DE/DS Implementation Option.

2021222324.2

```
FOR every T in Tags To Read Yet
{
	IF	[IsNotEmpty(T)]
	THEN
		AddToList(GetTLV(T), Data To Send)
		RemoveFromList(T, Tags To Read Yet)
	ENDIF
}
```

2021222324.8

```
FOR every T in Tags To Read Yet
{
	IF	[IsNotEmpty(T)]
	THEN
		AddToList(GetTLV(T), Data To Send)
		RemoveFromList(T, Tags To Read Yet)
	ENDIF
}
```

2021222324.15

```
'Status' in Outcome Parameter Set := END APPLICATION
'L3' in Error Indication := TRANSACTION DATA MISSING
'Msg On Error' in Error Indication := N/A
CreateDiscretionaryData ()
Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
		GetTLV(TagOf(Discretionary Data))) Signal
```

6.3.10 State 24 – Waiting for First Write Flag

State 24 is a state specific to data exchange processing and is only implemented if DE/DS Implementation Option is used.

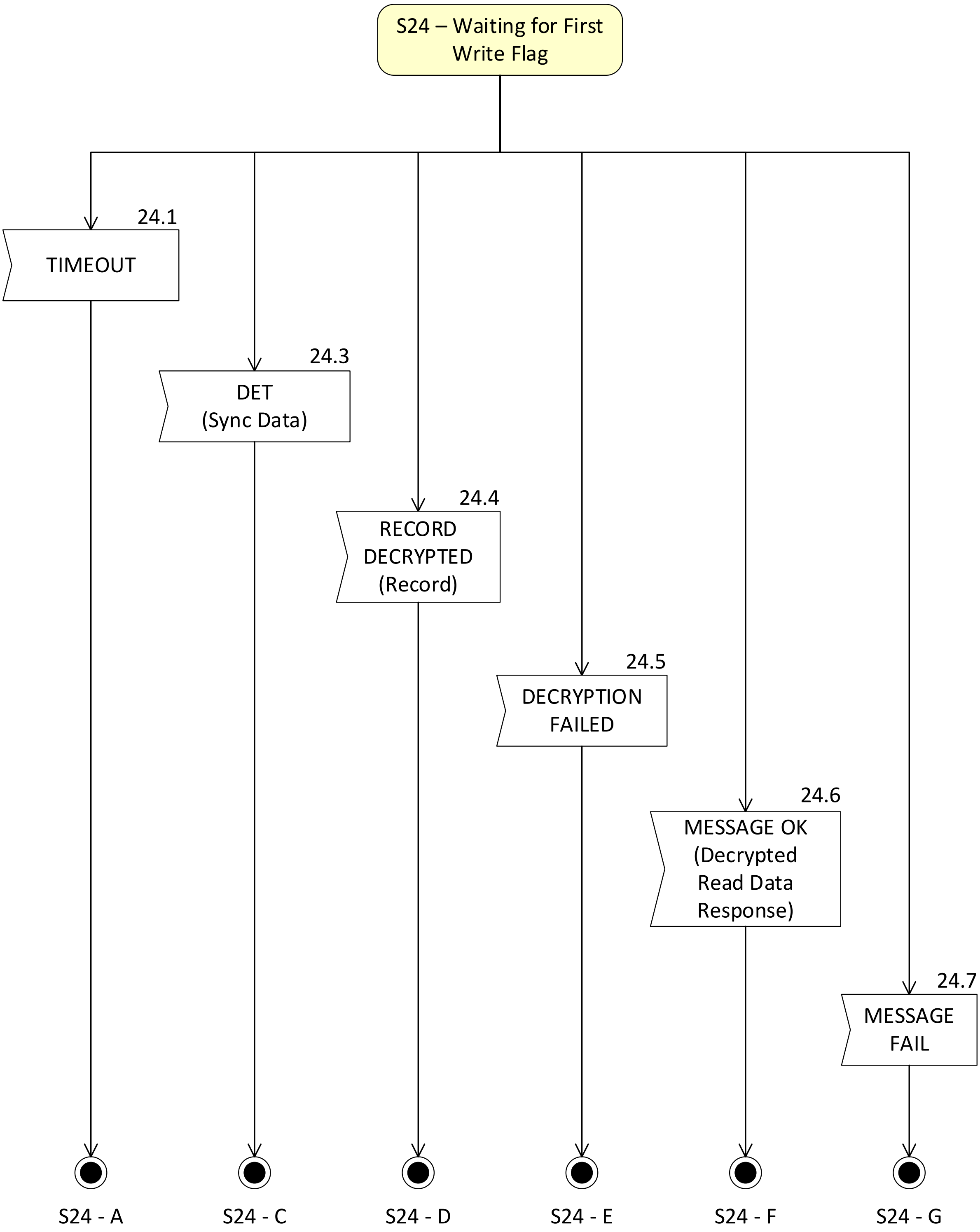
Table 6.7 shows the local variables used in `S24 – Waiting for First Write Flag`.

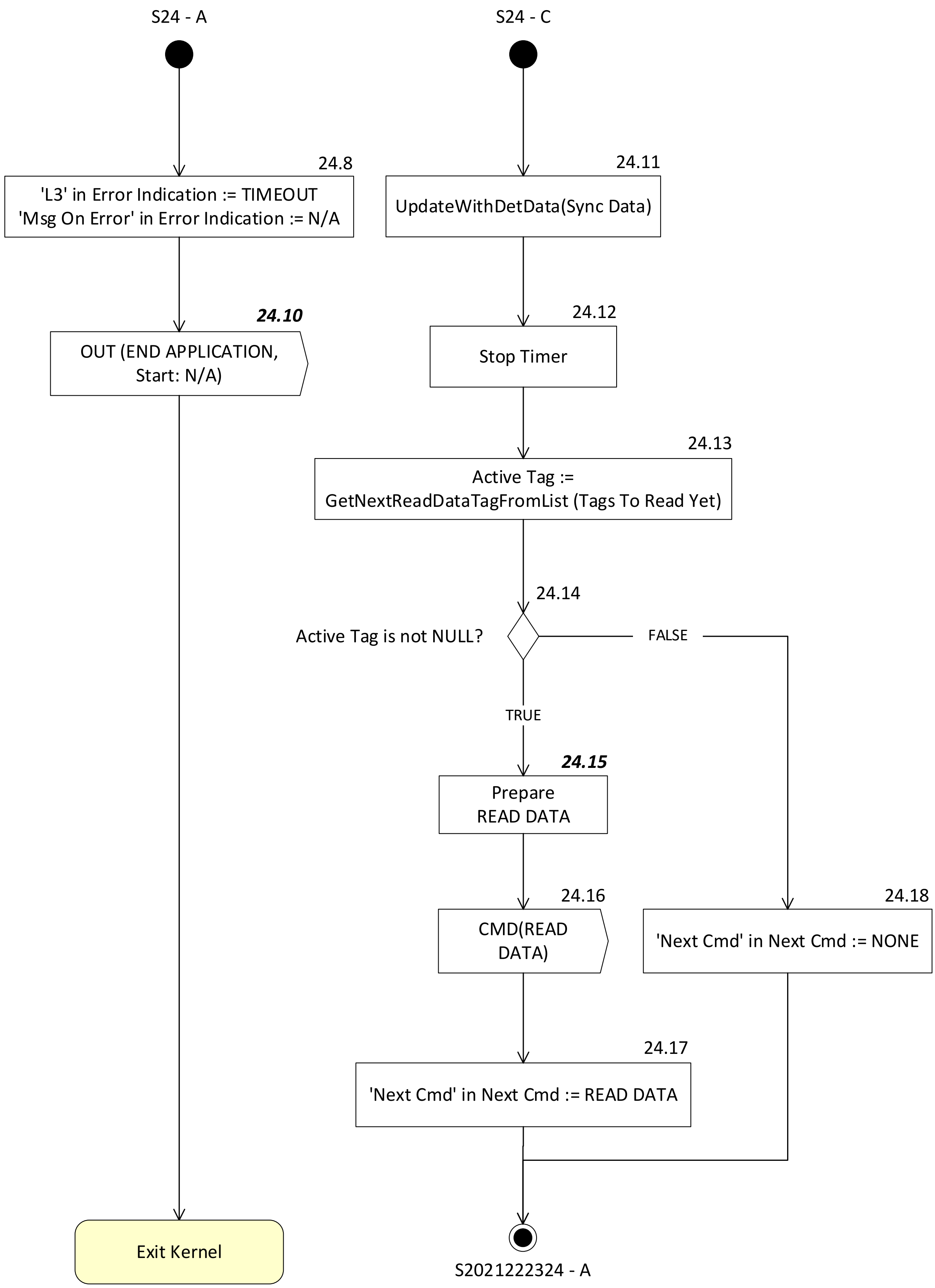
Table 6.7—State 24 Local Variables

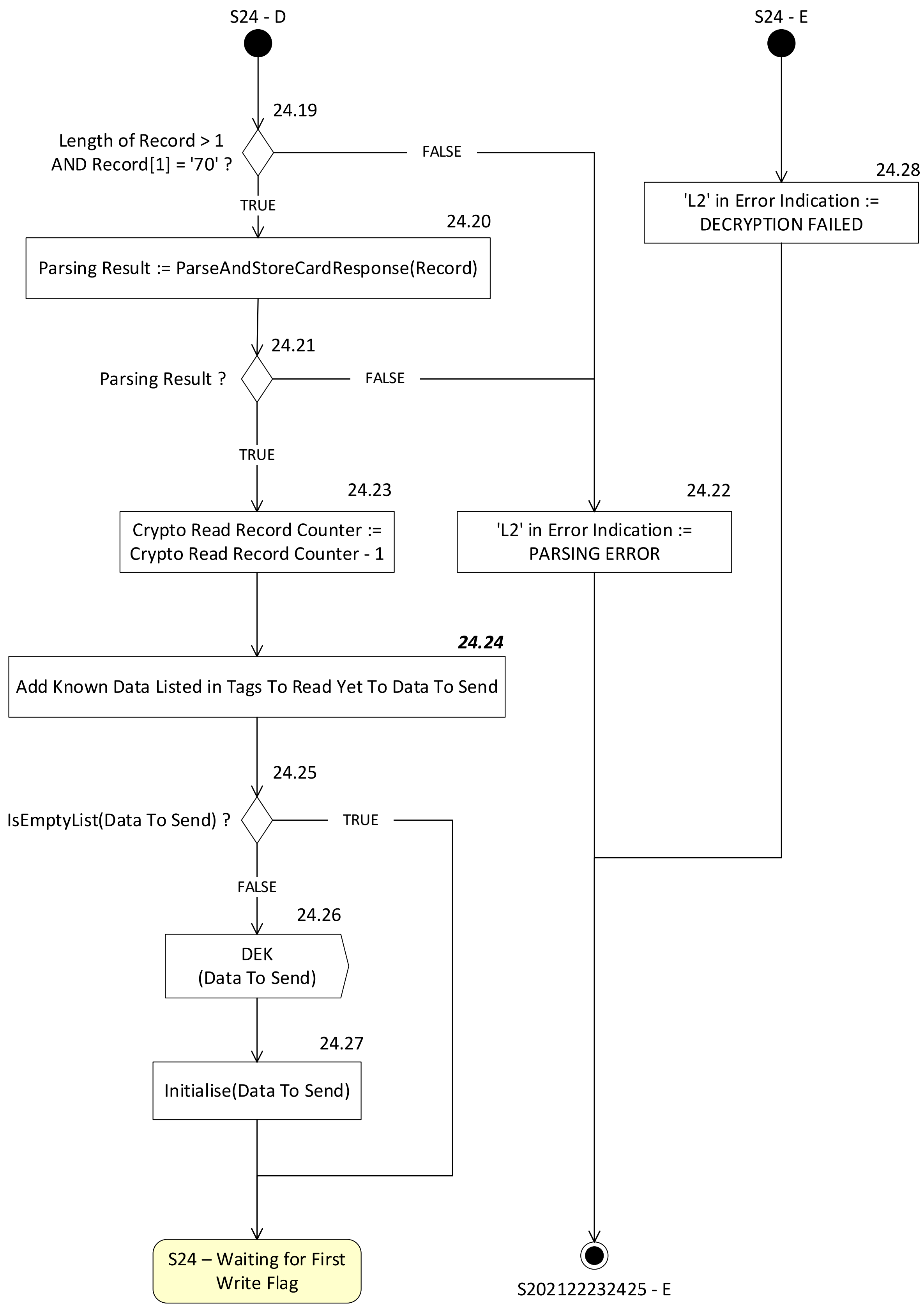
Name	Length	Type	Description
Sync Data	var.	b	List of data objects returned with DET Signal
T	var.	b	Tag of TLV-coded string
P2	1	b	P2 parameter of GENERATE AC command
Decrypted Read Data Response	var. up to 248	b	Decrypted read data response returned with MESSAGE OK Signal.
Record	var. up to 254	b	Decrypted record returned in RECORD DECRYPTED Signal
Parsing Result	1	b	Boolean used to store result of parsing a TLV-coded string

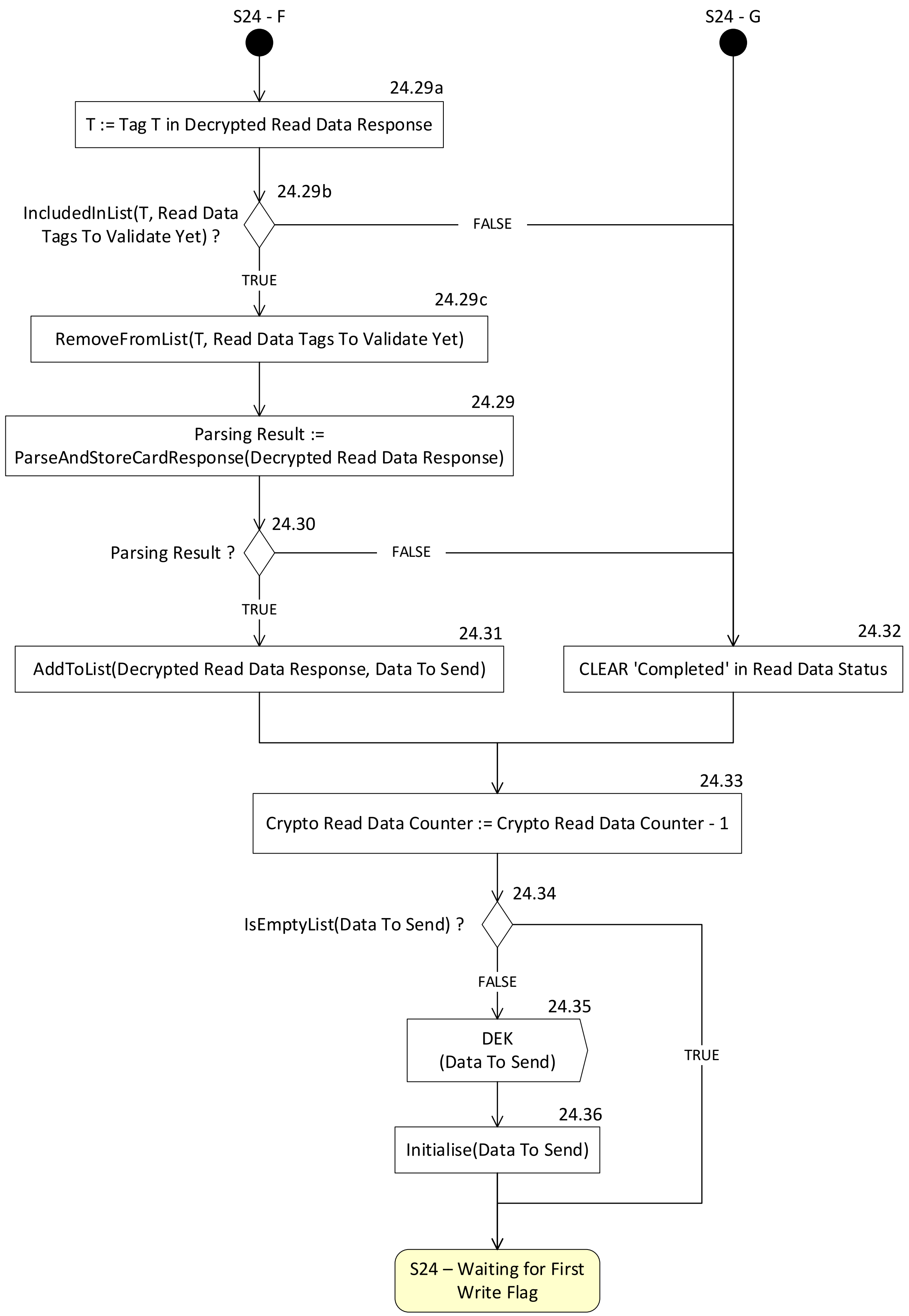
Figure 6.11 shows the flow diagram of `S24 – Waiting for First Write Flag`. Symbols in this diagram are labelled 24.X.

Figure 6.11—State 24 Flow Diagram




S24 - A
S24 - C
24.8
'L3' in Error Indication := TIMEOUT
'Msg On Error' in Error Indication := N/A
24.10
OUT (END APPLICATION, Start: N/A)
Exit Kernel
24.11
UpdateWithDetData(Sync Data)
24.12
Stop Timer
24.13
Active Tag := GetNextReadDataTagFromList (Tags To Read Yet)
24.14
Active Tag is not NULL?
FALSE
TRUE
24.15
Prepare READ DATA
24.16
CMD(READ DATA)
24.17
'Next Cmd' in Next Cmd := READ DATA
24.18
'Next Cmd' in Next Cmd := NONE
S2021222324 - A


S24 - D
S24 - E
24.19
Length of Record > 1
AND Record[1] = '70' ?
FALSE
TRUE
24.28
'L2' in Error Indication :=
DECRYPTION FAILED
24.20
Parsing Result := ParseAndStoreCardResponse(Record)
24.21
Parsing Result ?
FALSE
TRUE
24.23
Crypto Read Record Counter :=
Crypto Read Record Counter - 1
24.22
'L2' in Error Indication :=
PARSING ERROR
24.24
Add Known Data Listed in Tags To Read Yet To Data To Send
24.25
IsEmptyList(Data To Send) ?
TRUE
FALSE
24.26
DEK
(Data To Send)
24.27
Initialise(Data To Send)
S24 – Waiting for First
Write Flag
S202122232425 - E


S24 - F
S24 - G
24.29a
T := Tag T in Decrypted Read Data Response
24.29b
IncludedInList(T, Read Data Tags To Validate Yet) ?
FALSE
TRUE
24.29c
RemoveFromList(T, Read Data Tags To Validate Yet)
24.29
Parsing Result := ParseAndStoreCardResponse(Decrypted Read Data Response)
24.30
Parsing Result ?
FALSE
TRUE
24.31
AddToList(Decrypted Read Data Response, Data To Send)
24.32
CLEAR 'Completed' in Read Data Status
24.33
Crypto Read Data Counter := Crypto Read Data Counter - 1
24.34
IsEmptyList(Data To Send) ?
FALSE
TRUE
24.35
DEK (Data To Send)
24.36
Initialise(Data To Send)
S24 – Waiting for First Write Flag

24.10

```
'Status' in Outcome Parameter Set := END APPLICATION
CreateDiscretionaryData ()
Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
         GetTLV(TagOf(Discretionary Data))) Signal
```

24.15

Prepare READ DATA command for Active Tag as specified in section 5.5

```
AddToList(Active Tag, Read Data Tags To Validate Yet)
```

24.24

```
FOR every T in Tags To Read Yet
{
      IF    [IsNotEmpty(T)]
      THEN
            AddToList(GetTLV(T), Data To Send)
            RemoveFromList(T, Tags To Read Yet)
      ENDIF
}
```

6.3.11 State 25 – Waiting for Decrypted Data

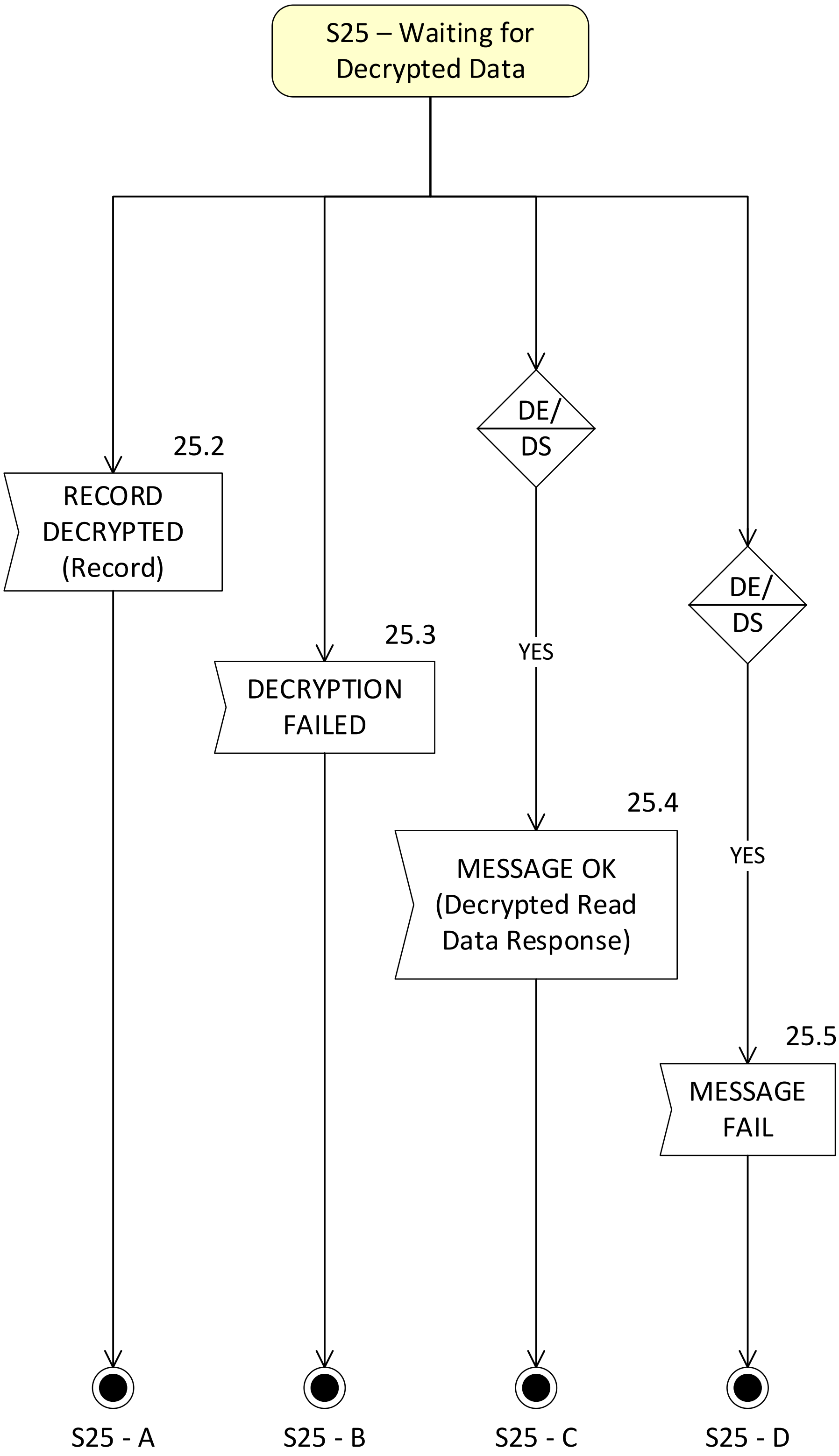
Table 6.8 shows the local variables used in `S25 – Waiting for Decrypted Data`.

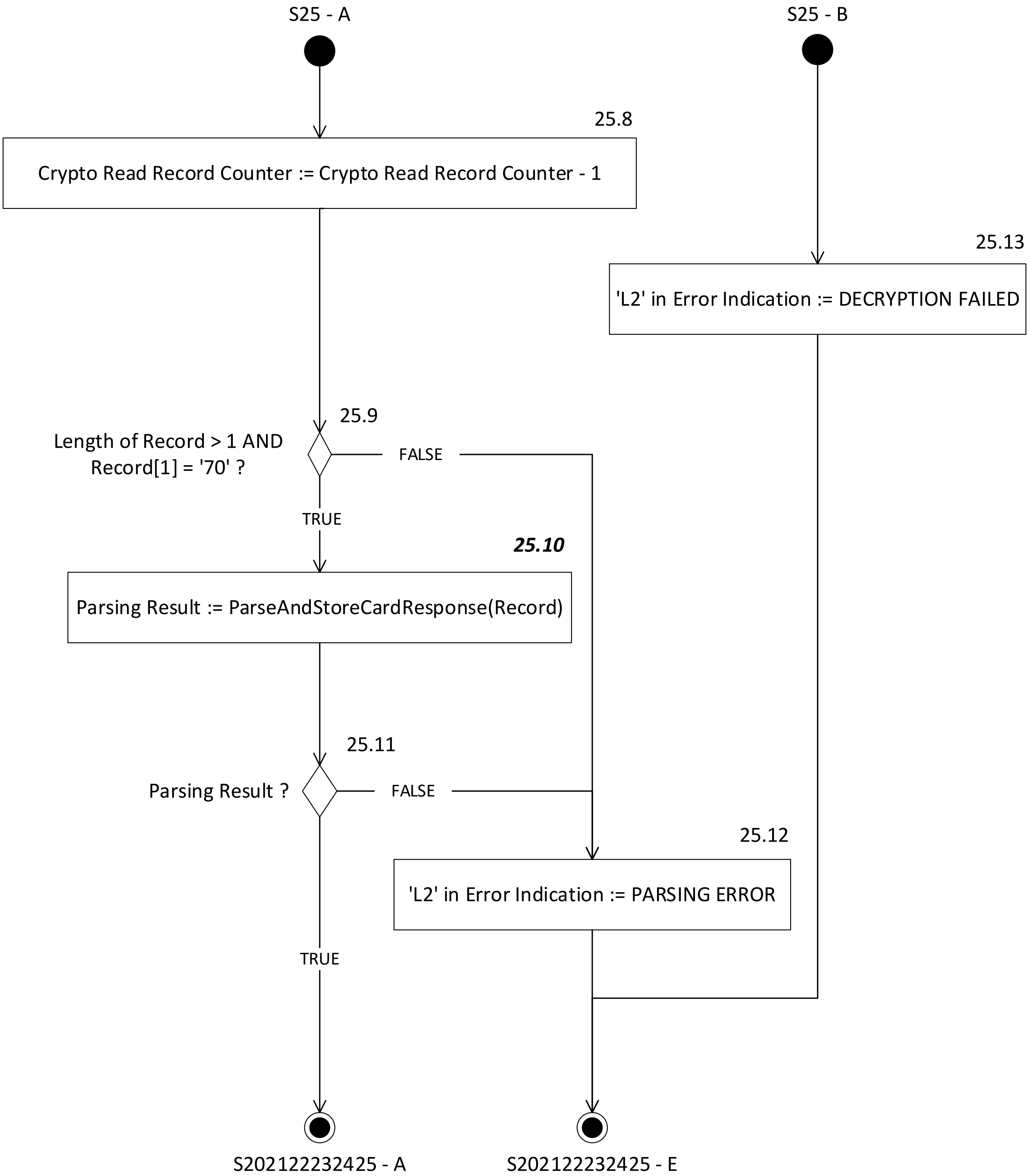
Table 6.8—State 25 Local Variables

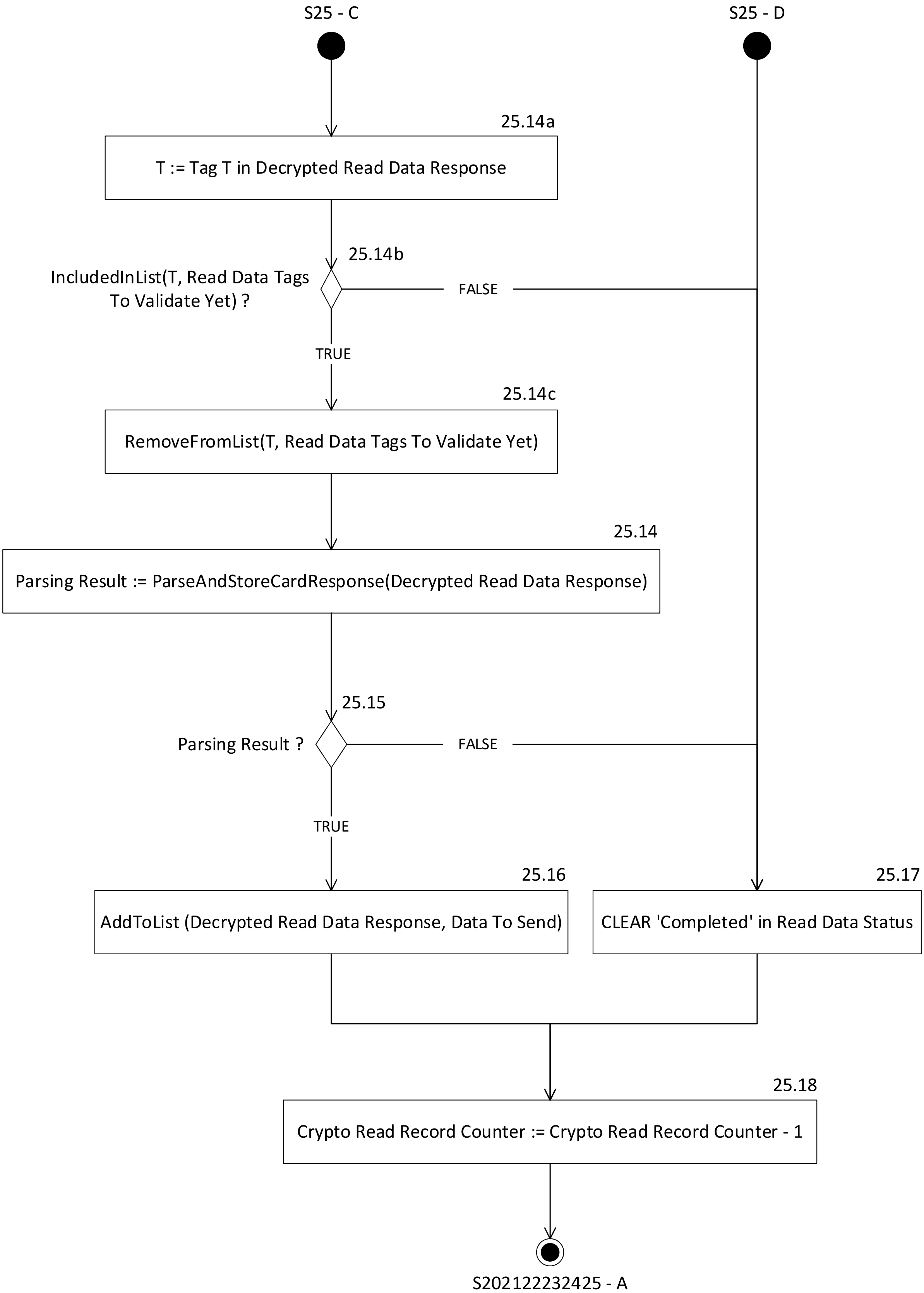
Name	Length	Type	Description
T	var.	b	Tag of TLV-coded string
P2	1	b	P2 parameter of GENERATE AC command
Decrypted Read Data Response	var. up to 248	b	Decrypted read data response returned with MESSAGE OK Signal.
Record	var. up to 254	b	Decrypted record returned in RECORD DECRYPTED Signal
Parsing Result	1	b	Boolean used to store result of parsing a TLV-coded string

Figure 6.12 shows the flow diagram of `S25 – Waiting for Decrypted Data`. Symbols in this diagram are labelled 25.X.

Figure 6.12—State 25 Flow Diagram




S25 - A
S25 - B
25.8
Crypto Read Record Counter := Crypto Read Record Counter - 1
25.13
'L2' in Error Indication := DECRYPTION FAILED
25.9
Length of Record > 1 AND
Record[1] = '70' ?
FALSE
TRUE
25.10
Parsing Result := ParseAndStoreCardResponse(Record)
25.11
Parsing Result ?
FALSE
25.12
'L2' in Error Indication := PARSING ERROR
TRUE
S202122232425 - A
S202122232425 - E


S25 - C
S25 - D
25.14a
T := Tag T in Decrypted Read Data Response
25.14b
IncludedInList(T, Read Data Tags To Validate Yet) ?
FALSE
TRUE
25.14c
RemoveFromList(T, Read Data Tags To Validate Yet)
25.14
Parsing Result := ParseAndStoreCardResponse(Decrypted Read Data Response)
25.15
Parsing Result ?
FALSE
TRUE
25.16
AddToList (Decrypted Read Data Response, Data To Send)
25.17
CLEAR 'Completed' in Read Data Status
25.18
Crypto Read Record Counter := Crypto Read Record Counter - 1
S202122232425 - A

Note that the symbols 25.4, 25.5, 25.14, 25.14a, 25.14b, 25.14c, 25.15, 25.16, 25.17 and 25.18 are only implemented for the DE/DS Implementation Option.

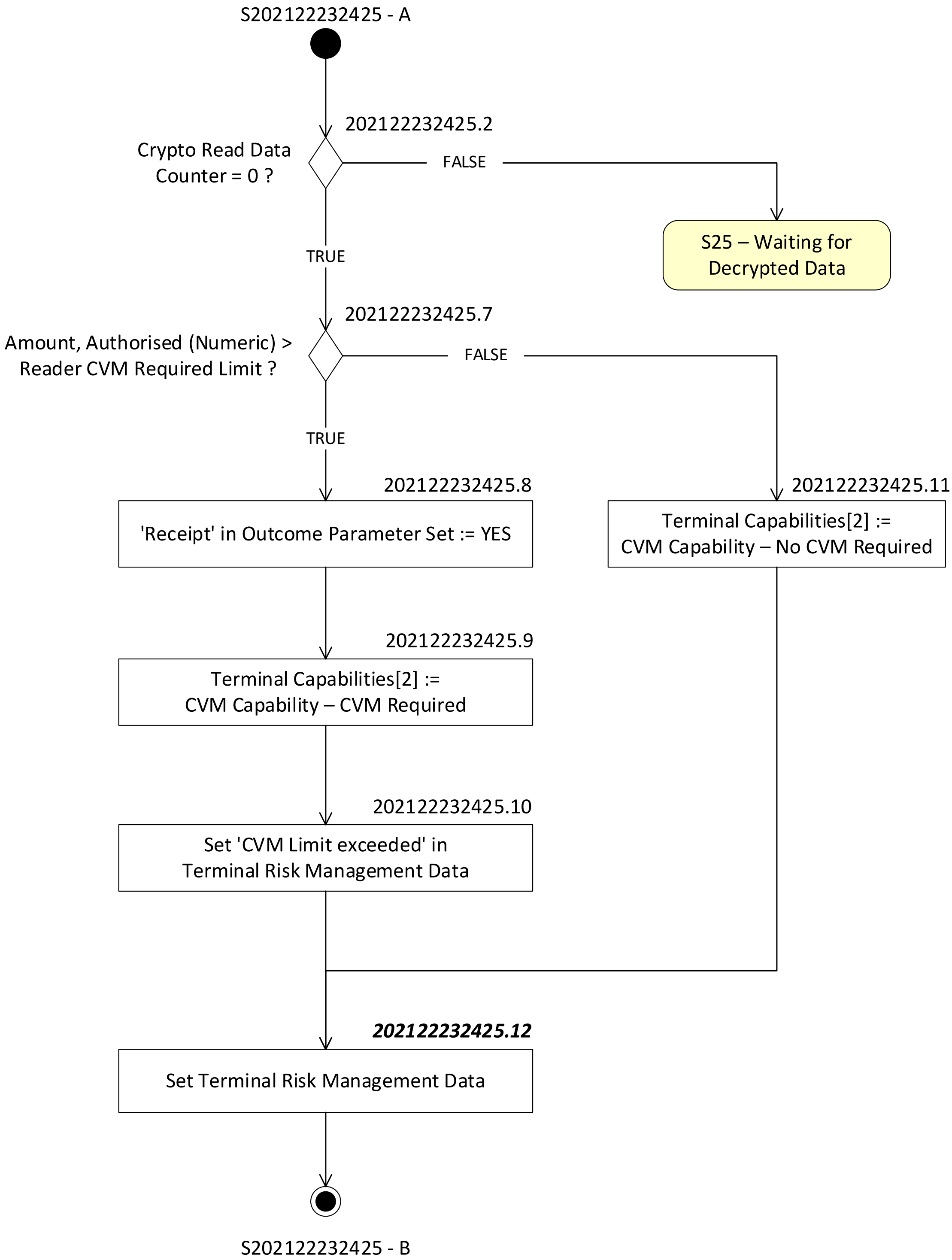
25.10

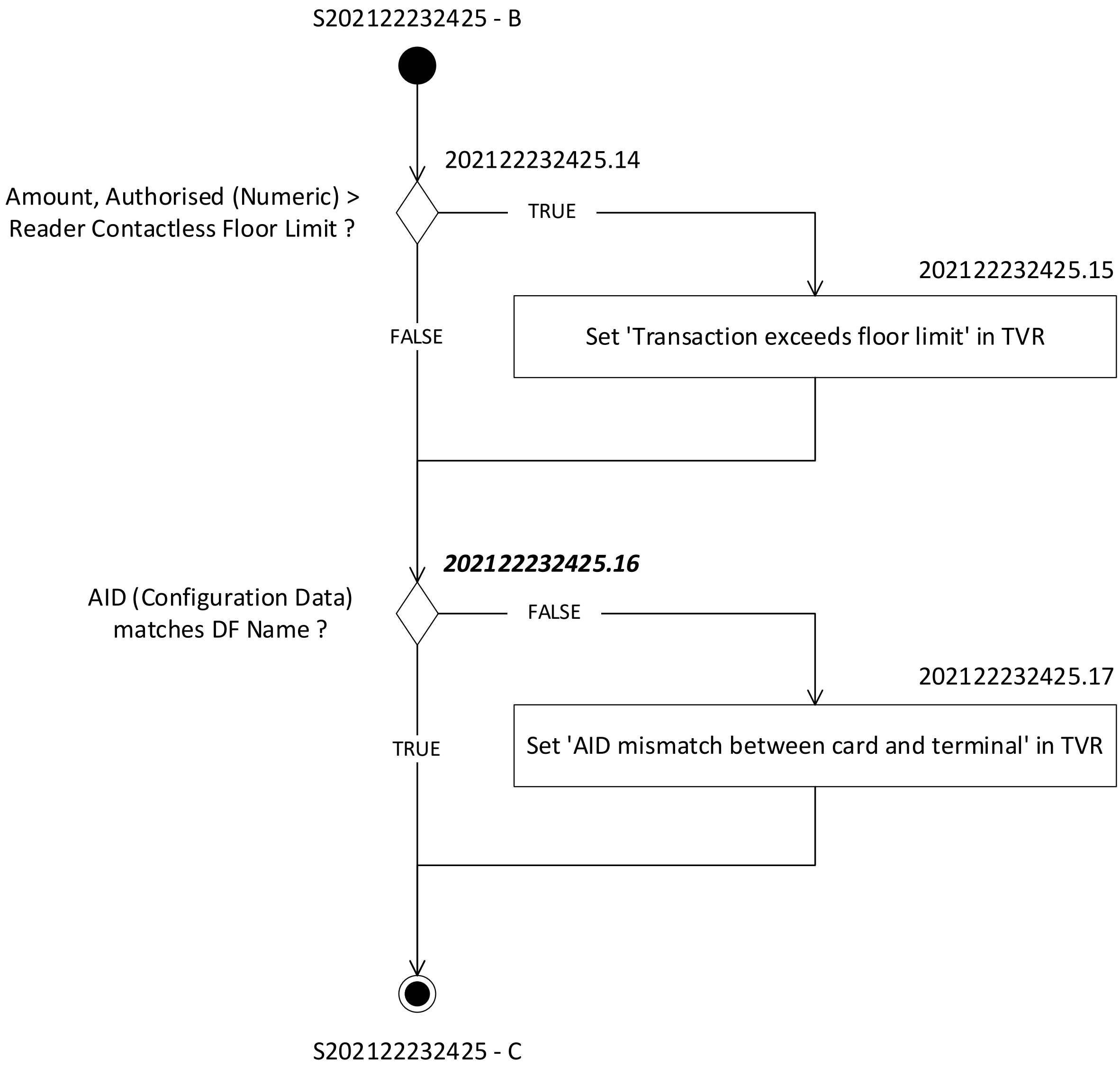
Note that when DE/DS is not implemented, the implementation may postpone the parsing of the Record and the corresponding update of the TLV Database until after the GENERATE AC command, but no later than S2627 – A before symbol S2627.1.

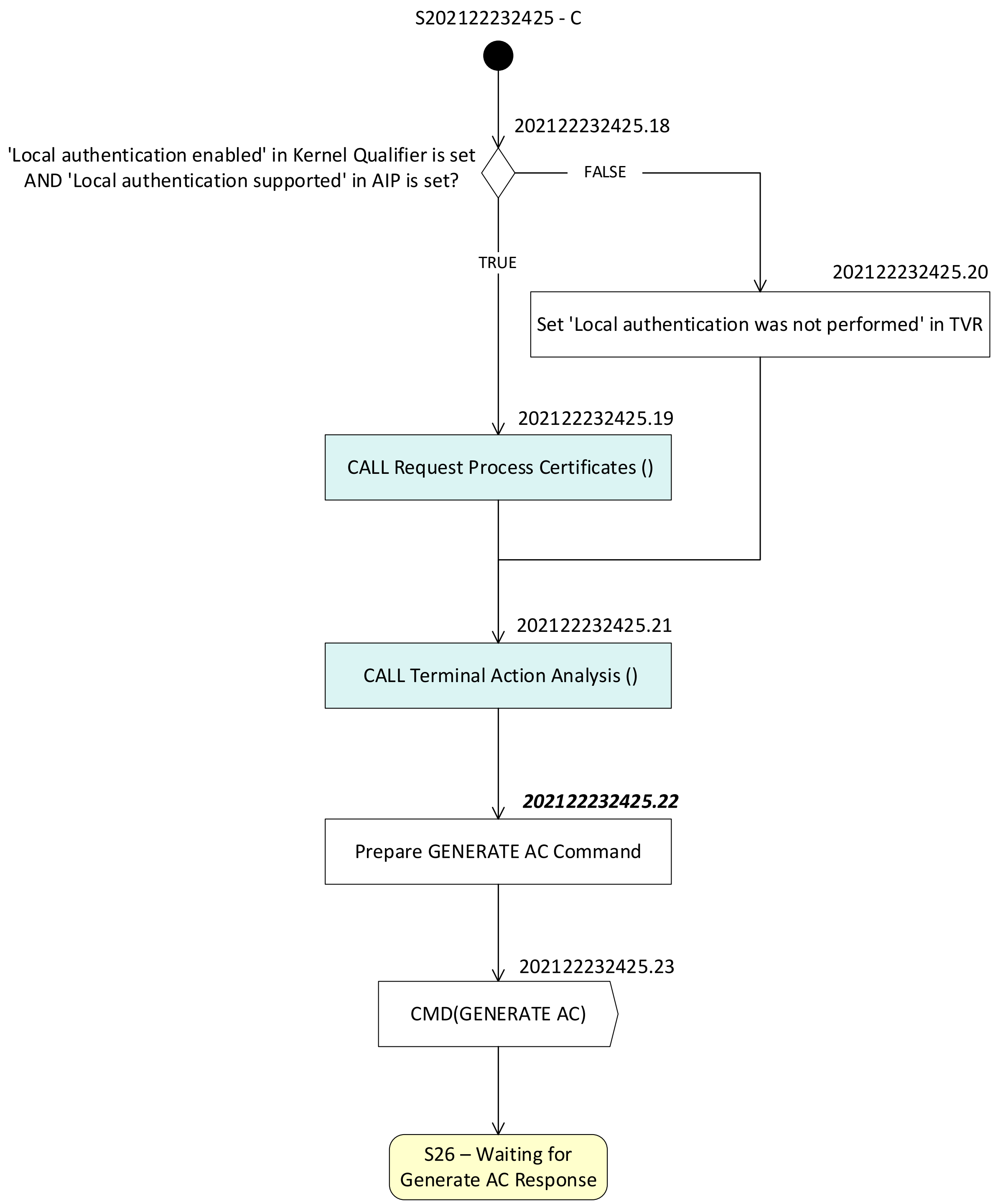
6.3.12 States 20, 21, 22, 23, 24, and 25 – Common Processing

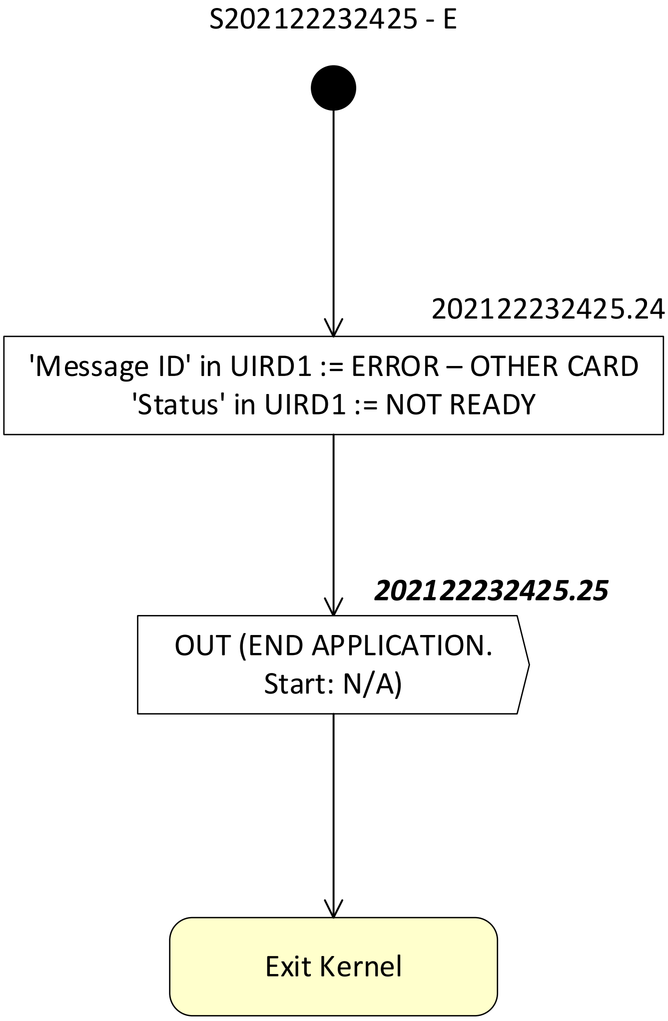
Figure 6.13 shows the flow diagram for common processing between states 20, 21, 22, 23, 24, and 25. Symbols in this diagram are labelled 202122232425.X.

Figure 6.13—States 20, 21, 22, 23, 24, and 25 – Common Processing – Flow Diagram




S202122232425 - B
202122232425.14
Amount, Authorised (Numeric) > Reader Contactless Floor Limit ?
TRUE
202122232425.15
Set 'Transaction exceeds floor limit' in TVR
FALSE
202122232425.16
AID (Configuration Data) matches DF Name ?
FALSE
202122232425.17
Set 'AID mismatch between card and terminal' in TVR
TRUE
S202122232425 - C


S202122232425 - C
202122232425.18
'Local authentication enabled' in Kernel Qualifier is set AND 'Local authentication supported' in AIP is set?
FALSE
TRUE
202122232425.20
Set 'Local authentication was not performed' in TVR
202122232425.19
CALL Request Process Certificates ()
202122232425.21
CALL Terminal Action Analysis ()
202122232425.22
Prepare GENERATE AC Command
202122232425.23
CMD(GENERATE AC)
S26 – Waiting for Generate AC Response


S202122232425 - E
202122232425.24
'Message ID' in UIRD1 := ERROR – OTHER CARD
'Status' in UIRD1 := NOT READY
202122232425.25
OUT (END APPLICATION.
Start: N/A)
Exit Kernel

202122232425.12

Terminal Risk Management Data[1][7] := Terminal Capabilities[2][7]

Terminal Risk Management Data[1][6] := Terminal Capabilities[2][6]

Terminal Risk Management Data[1][4] := Terminal Capabilities[2][4]

Terminal Risk Management Data[1][3] := Terminal Capabilities[2][3]

202122232425.16

AID (Configuration Data) matches DF Name ?:

n := GetLength(TagOf(Application Identifier (Configuration Data)))
[(n ≤ GetLength(TagOf(DF Name))
AND (DF Name[1:n] = Application Identifier (Configuration Data))]

202122232425.22

Reference Control Parameter: = '00'

'AC type' in Reference Control Parameter := 'Decision' in Kernel Decision

IF [IsNotEmptyList(Data Envelopes To Write Yet)]

THEN

P2 := '80'

ELSE

P2 := '00'

ENDIF

Use CDOL1 to create CDOL1 Related Data as a concatenated list of data objects without tags or lengths following the rules specified in section 4.1.4.

Prepare GENERATE AC command as specified in section 5.3.

202122232425.25

'Status' in Outcome Parameter Set := END APPLICATION

CreateDiscretionaryData ()

SET 'UI Request on Outcome Present' in Outcome Parameter Set

Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
GetTLV(TagOf(Discretionary Data)),
GetTLV(TagOf(User Interface Request Data 1))) Signal

6.3.13 State 26 – Waiting for Generate AC Response

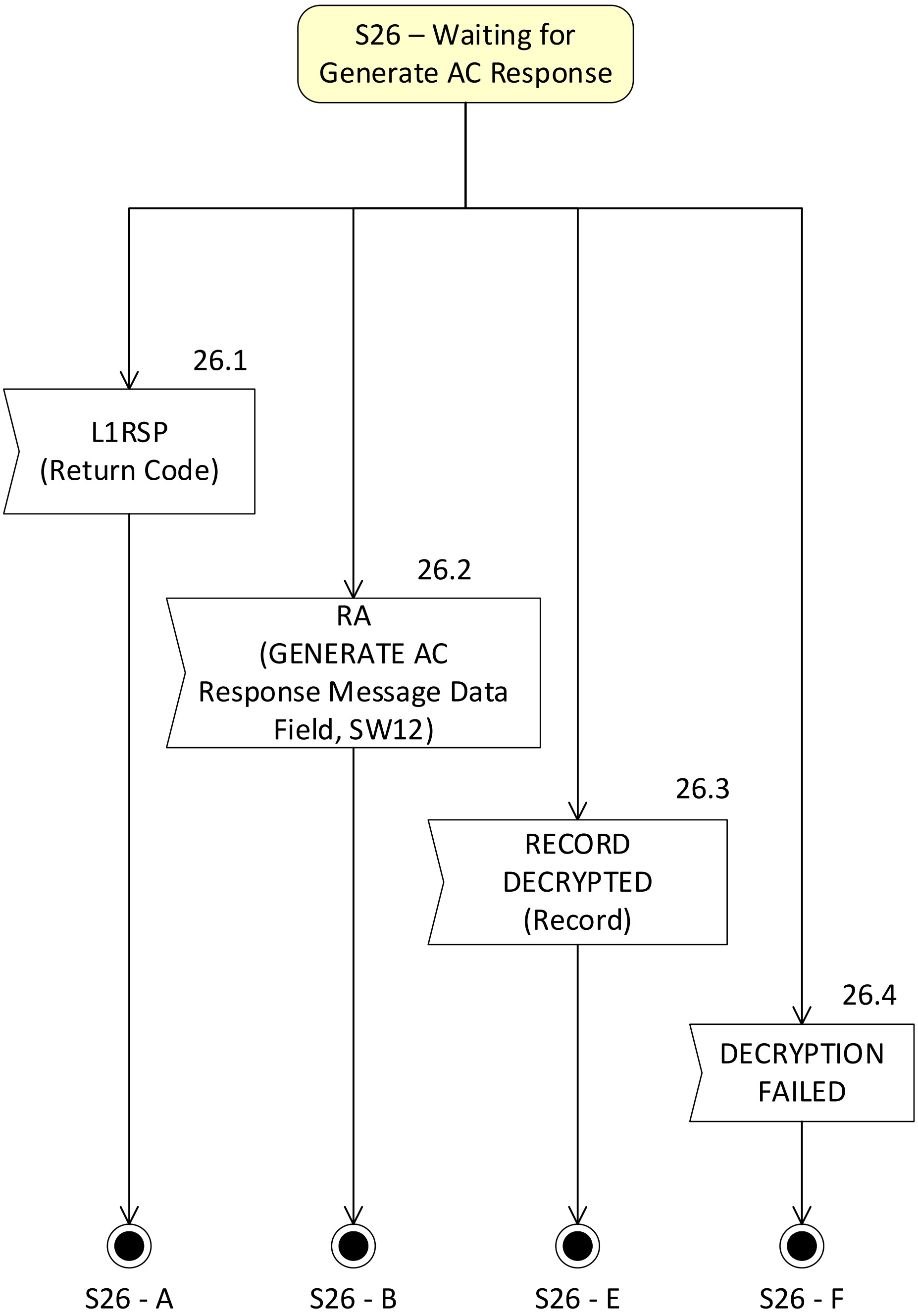
Table 6.9 shows the local variables used in `S26 – Waiting for Generate AC Response`.

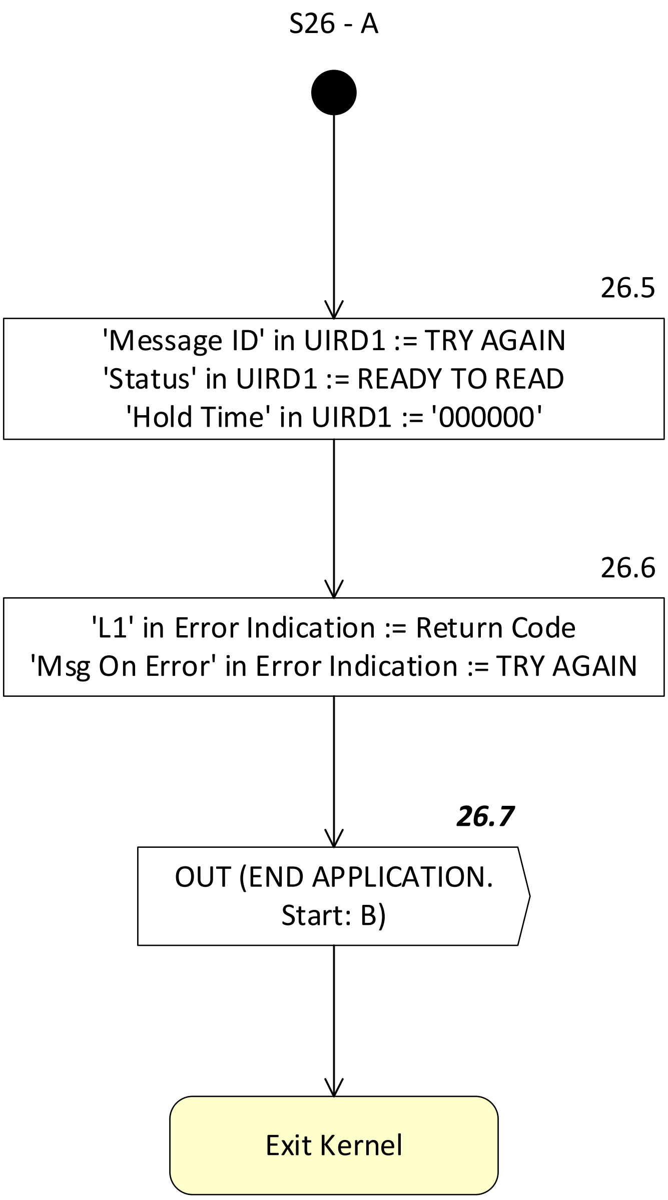
Table 6.9—State 26 Local Variables

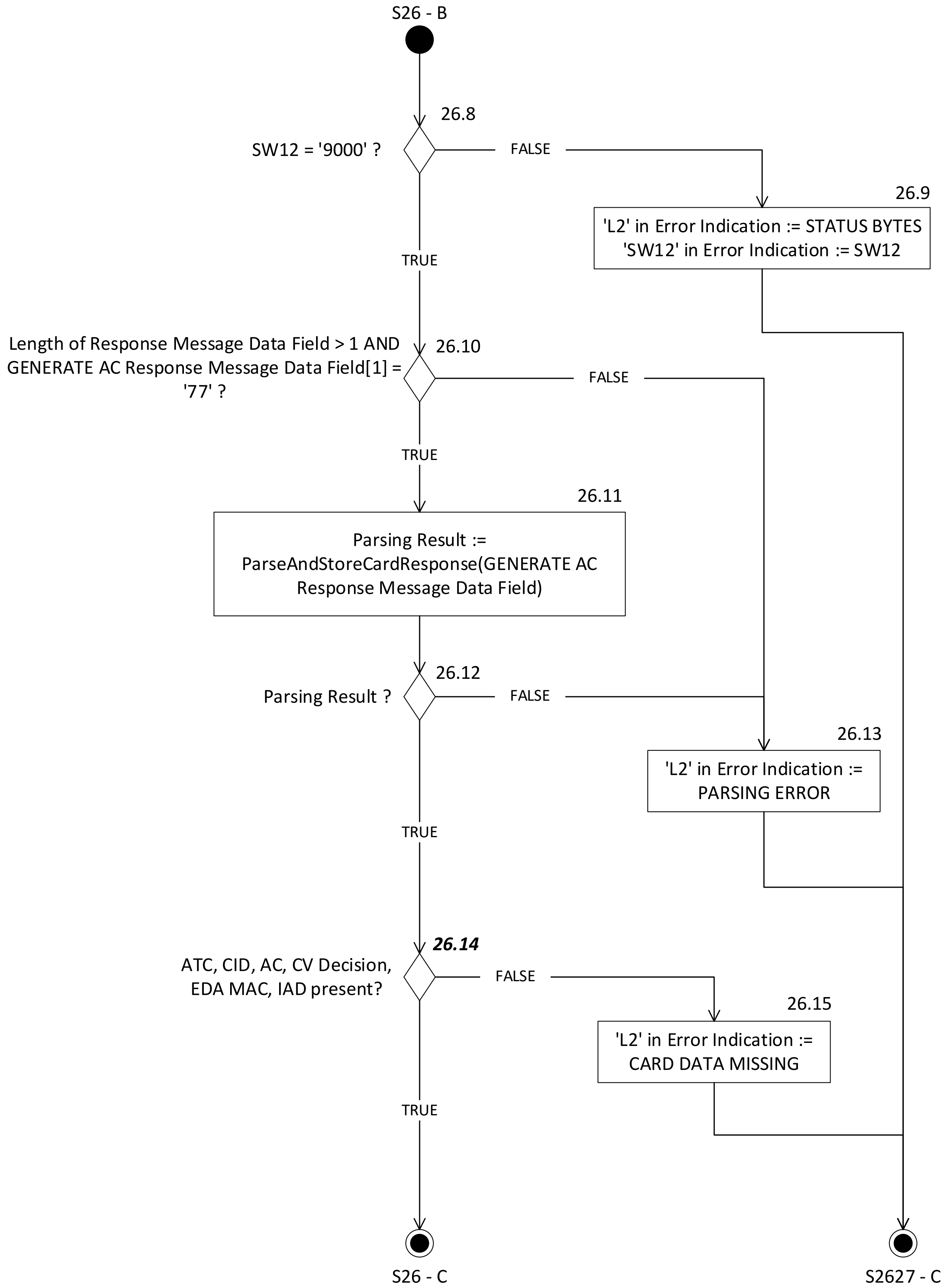
Name	Length	Format	Description
Return Code	1	b	Value returned with L1RSP Signal (TIMEOUT ERROR, PROTOCOL ERROR, TRANSMISSION ERROR)
Parsing Result	1	b	Boolean used to store result of parsing a TLV-coded string
Record	var. up to 254	b	Decrypted record returned in RECORD DECRYPTED Signal
SW12	2	b	Status bytes
T	var.	b	Tag of TLV-coded string

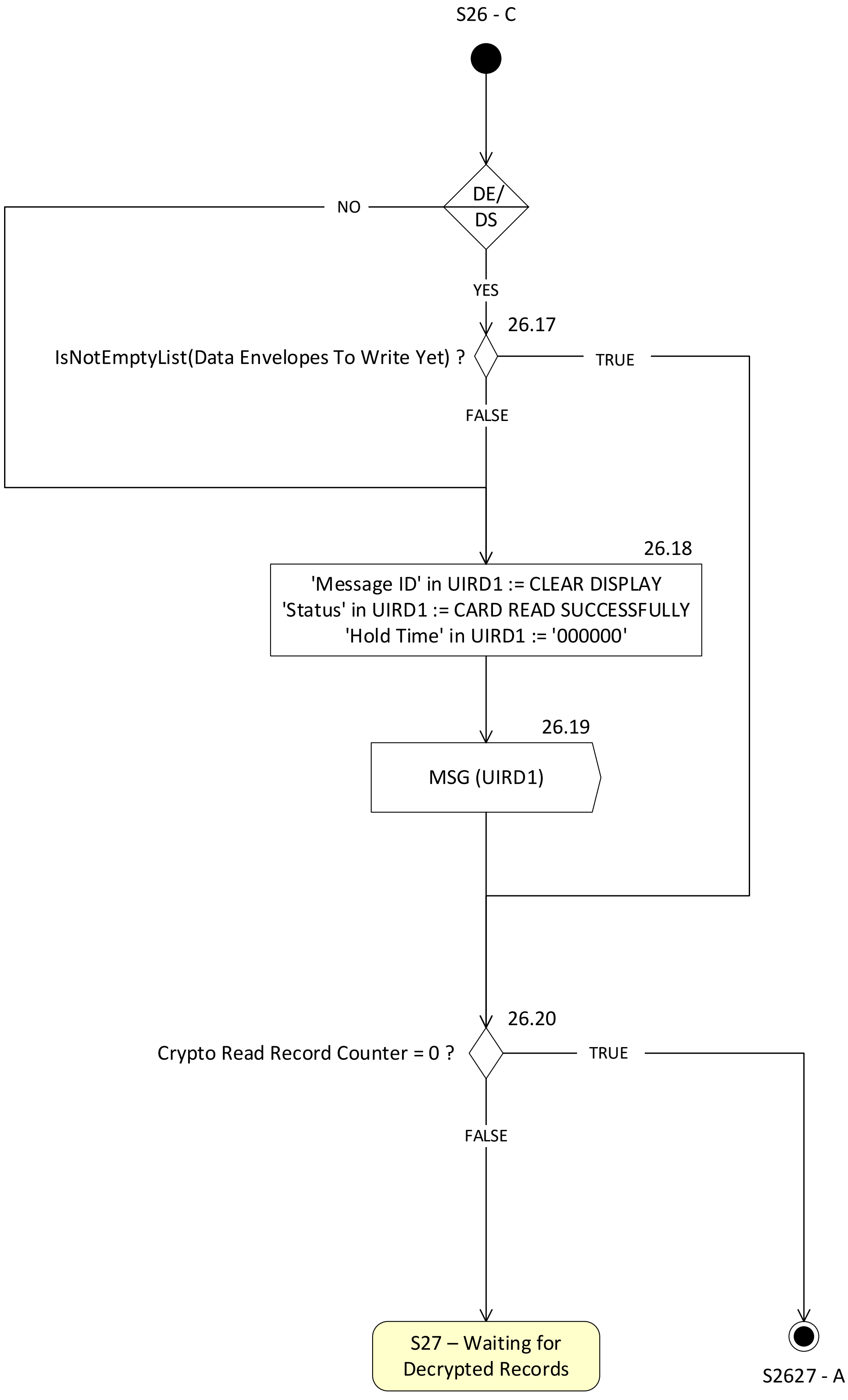
Figure 6.14 shows the flow diagram of `S26 – Waiting for Generate AC Response`. Symbols in this diagram are labelled 26.X.

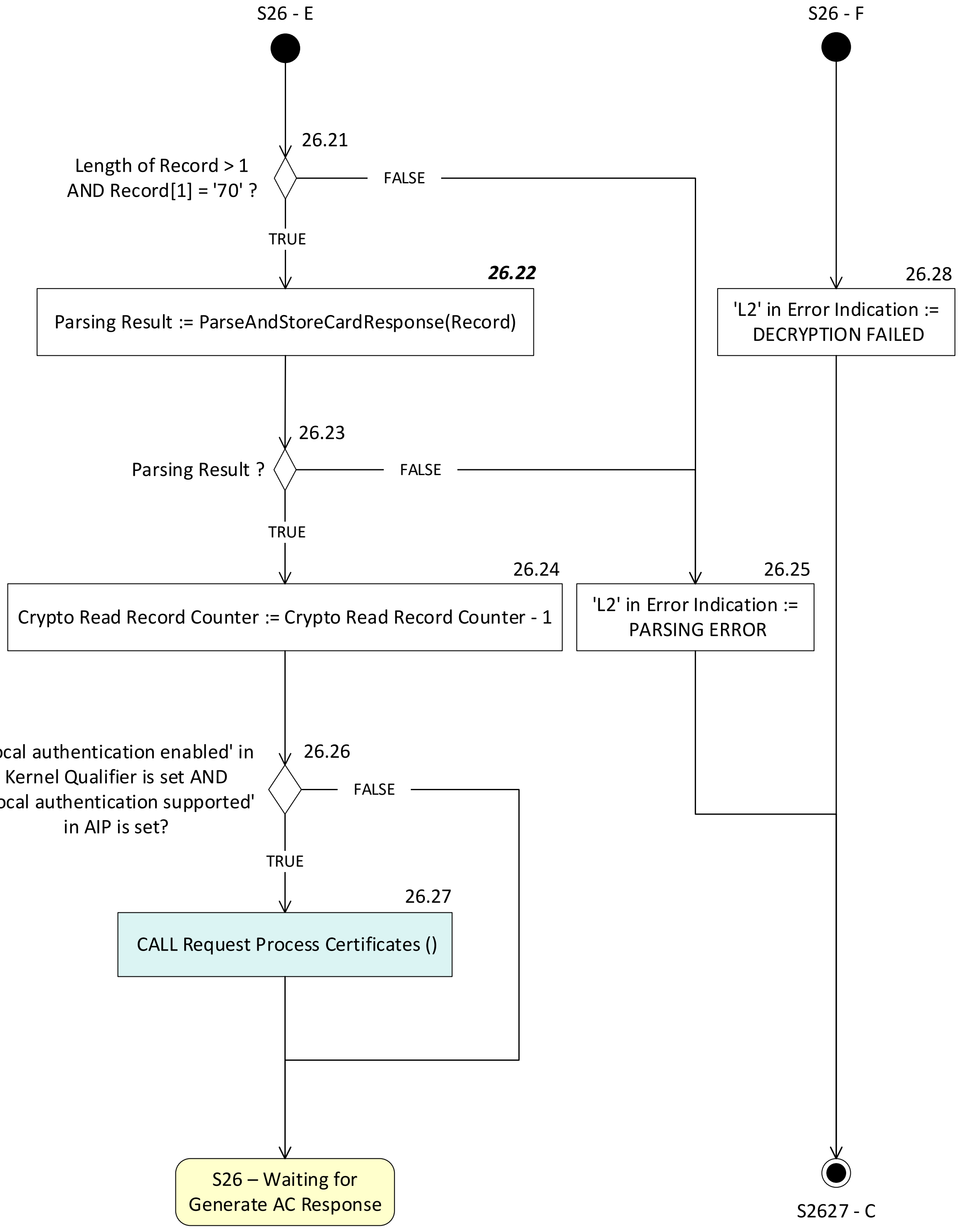
Figure 6.14—State 26 Flow Diagram




S26 - A
26.5
'Message ID' in UIRD1 := TRY AGAIN
'Status' in UIRD1 := READY TO READ
'Hold Time' in UIRD1 := '000000'
26.6
'L1' in Error Indication := Return Code
'Msg On Error' in Error Indication := TRY AGAIN
26.7
OUT (END APPLICATION.
Start: B)
Exit Kernel


S26 - B
26.8
SW12 = '9000' ?
FALSE
26.9
'L2' in Error Indication := STATUS BYTES
'SW12' in Error Indication := SW12
TRUE
Length of Response Message Data Field > 1 AND GENERATE AC Response Message Data Field[1] = '77' ?
26.10
FALSE
TRUE
26.11
Parsing Result := ParseAndStoreCardResponse(GENERATE AC Response Message Data Field)
26.12
Parsing Result ?
FALSE
26.13
'L2' in Error Indication := PARSING ERROR
TRUE
26.14
ATC, CID, AC, CV Decision, EDA MAC, IAD present?
FALSE
26.15
'L2' in Error Indication := CARD DATA MISSING
TRUE
S26 - C
S2627 - C


S26 - C
DE/
DS
NO
YES
26.17
IsNotEmptyList(Data Envelopes To Write Yet) ?
TRUE
FALSE
26.18
'Message ID' in UIRD1 := CLEAR DISPLAY
'Status' in UIRD1 := CARD READ SUCCESSFULLY
'Hold Time' in UIRD1 := '000000'
26.19
MSG (UIRD1)
26.20
Crypto Read Record Counter = 0 ?
TRUE
FALSE
S27 – Waiting for Decrypted Records
S2627 - A


S26 - E
S26 - F
26.21
Length of Record > 1
AND Record[1] = '70' ?
FALSE
TRUE
26.22
Parsing Result := ParseAndStoreCardResponse(Record)
26.28
'L2' in Error Indication :=
DECRYPTION FAILED
26.23
Parsing Result ?
FALSE
TRUE
26.24
Crypto Read Record Counter := Crypto Read Record Counter - 1
26.25
'L2' in Error Indication :=
PARSING ERROR
'Local authentication enabled' in
Kernel Qualifier is set AND
'Local authentication supported'
in AIP is set?
26.26
FALSE
TRUE
26.27
CALL Request Process Certificates ()
S26 – Waiting for
Generate AC Response
S2627 - C

Note that the symbol 26.17 is only implemented for the DE/DS implementation option.

26.7

'Status' in Outcome Parameter Set := END APPLICATION

'Start' in Outcome Parameter Set := B

SET 'UI Request on Restart Present' in Outcome Parameter Set

CreateDiscretionaryData ()

```
Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
         GetTLV(TagOf(Discretionary Data)),
         GetTLV(TagOf(User Interface Request Data 1))) Signal
```

26.14

ATC, CID, AC, CV Decision, EDA MAC, IAD present ?:

```
[
IsNotEmpty(TagOf(Application Transaction Counter))
AND IsNotEmpty(TagOf(Cryptogram Information Data))
AND IsNotEmpty(TagOf(Application Cryptogram))
AND IsNotEmpty(TagOf(Cardholder Verification Decision))
AND IsNotEmpty(TagOf(Enhanced Data Authentication MAC))
AND IsNotEmpty(TagOf(Issuer Application Data))
]
```

26.22

Note that when DE/DS is not implemented, the implementation may postpone the parsing of the Record and the corresponding update of the TLV Database until after the GENERATE AC response is received, but no later than S2627 – A before symbol S2627.1.

6.3.14 State 27 – Waiting for Decrypted Records

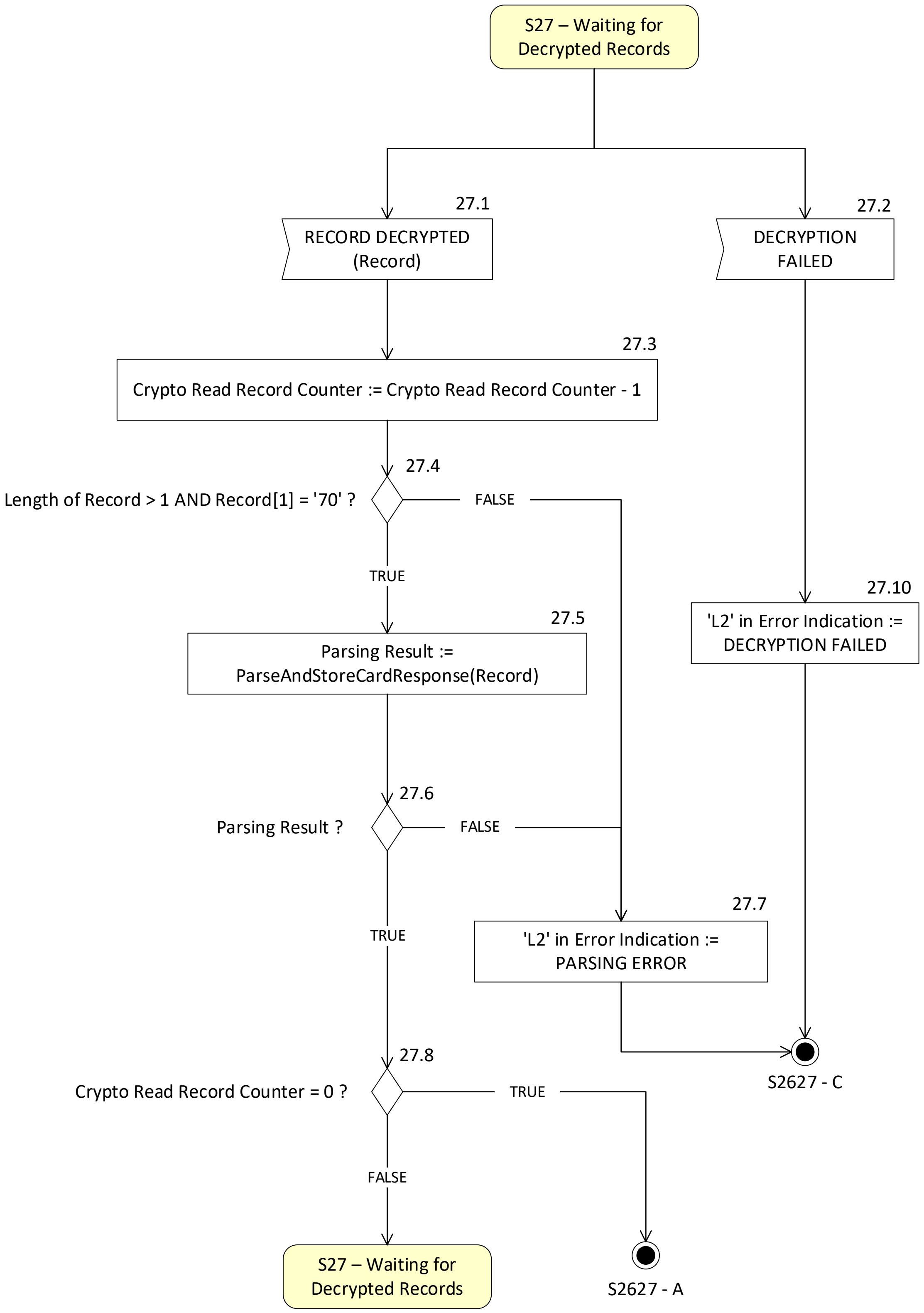
Table 6.10 shows the local variables used in `S27 – Waiting for Decrypted Records`.

Table 6.10—State 27 Local Variables

Name	Length	Format	Description
Parsing Result	1	b	Boolean used to store result of parsing a TLV-coded string
Record	var. up to 254	b	Decrypted record returned in RECORD DECRYPTED Signal
T	var.	b	Tag of TLV-coded string

Figure 6.15 shows the flow diagram of `S27 – Waiting for Decrypted Records`. Symbols in this diagram are labelled 27.X.

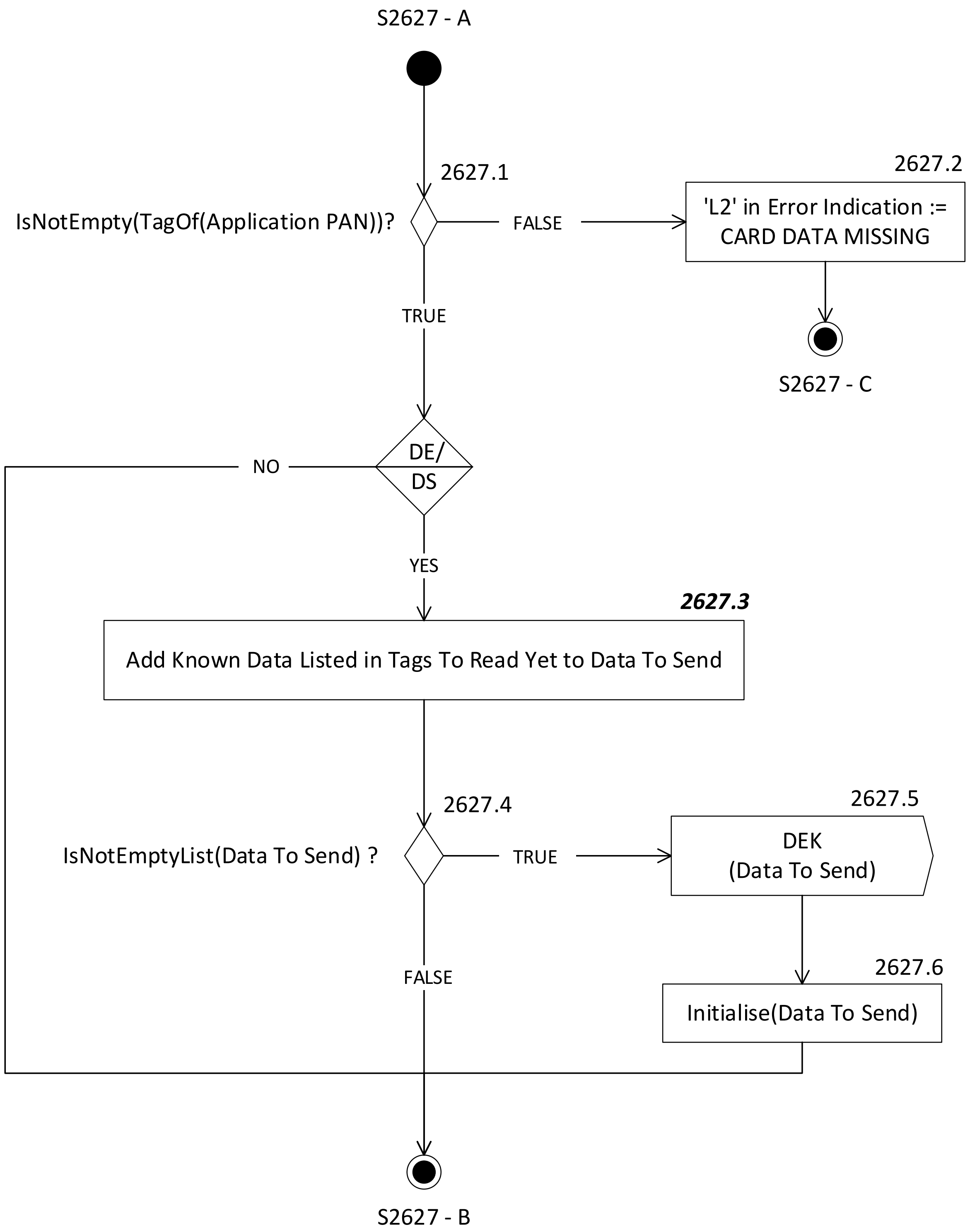
Figure 6.15—State 27 Flow Diagram

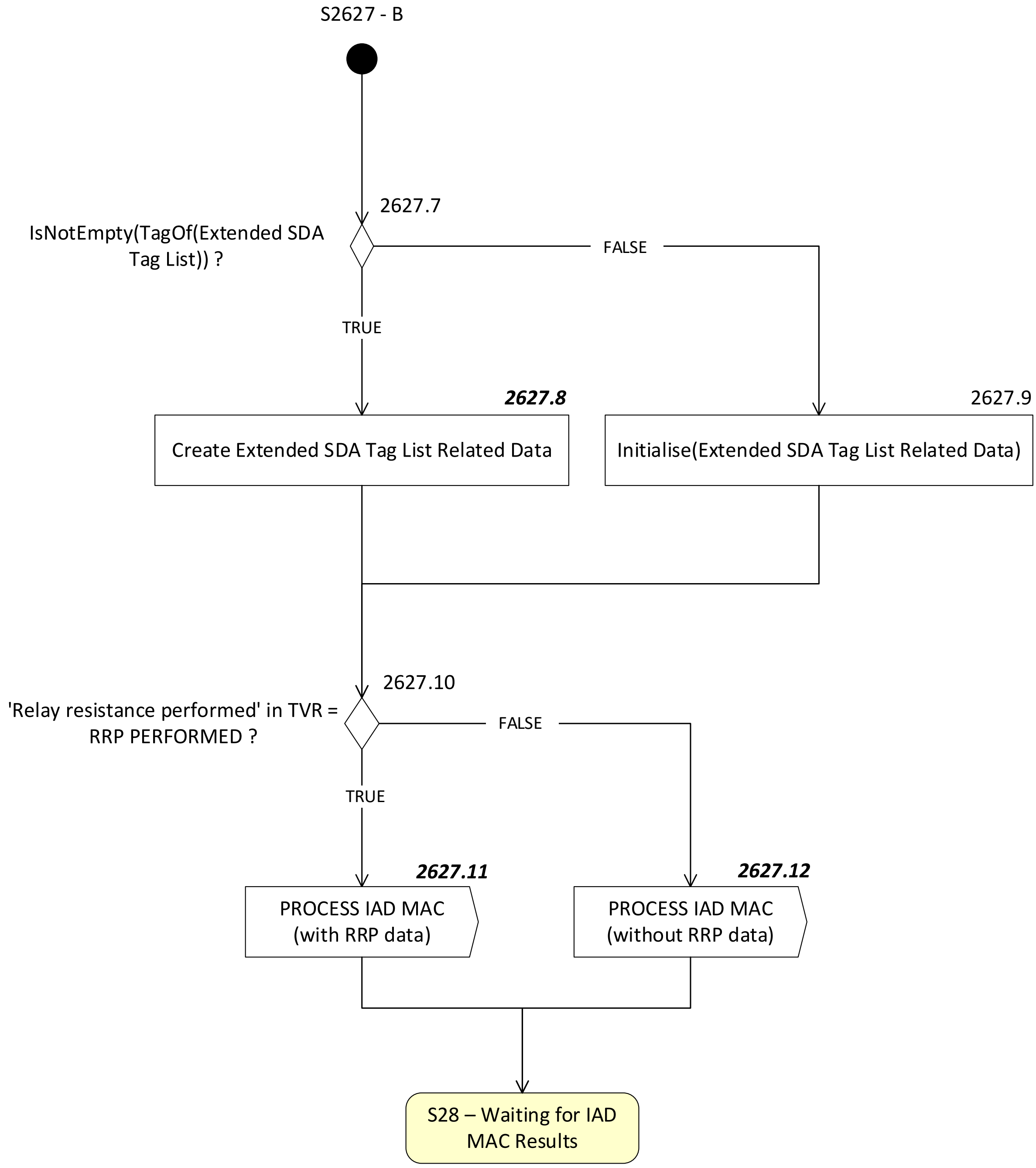


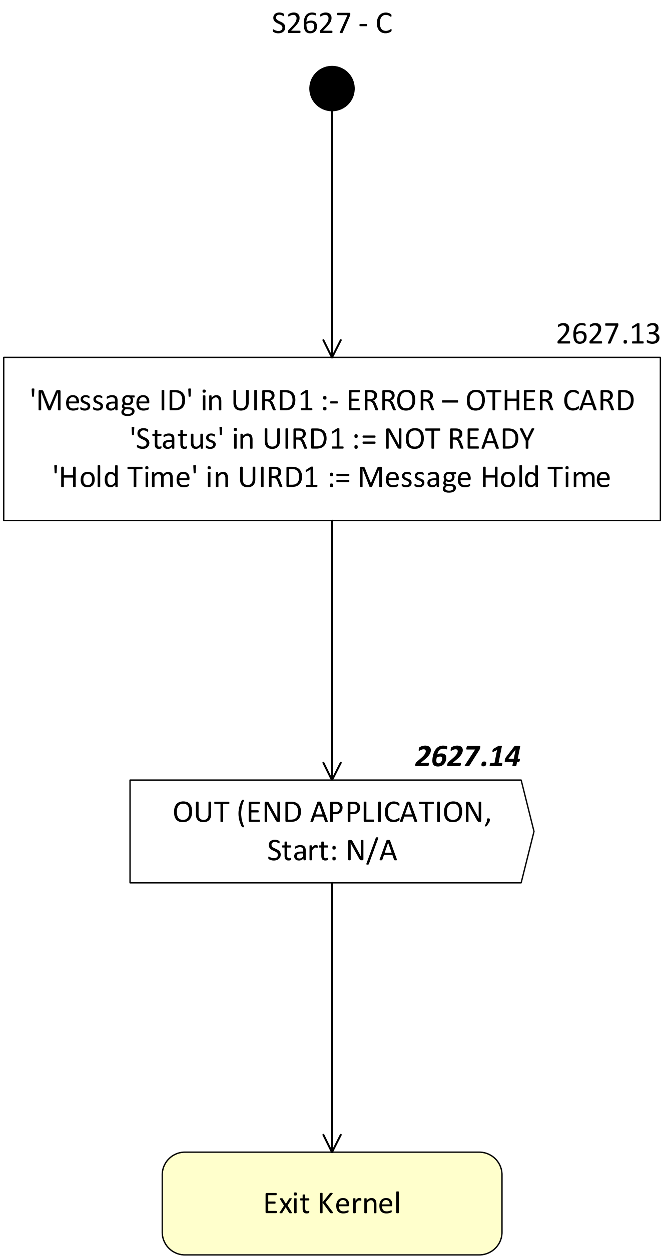
6.3.15 States 26 and 27 – Common Processing

Figure 6.16 shows the flow diagram for common processing between states 26 and 27. Symbols in this diagram are labelled 2627.X.

Figure 6.16—States 26 and 27 – Common Processing – Flow Diagram




S2627 - B
2627.7
IsNotEmpty(TagOf(Extended SDA Tag List)) ?
FALSE
TRUE
2627.8
Create Extended SDA Tag List Related Data
2627.9
Initialise(Extended SDA Tag List Related Data)
2627.10
'Relay resistance performed' in TVR = RRP PERFORMED ?
FALSE
TRUE
2627.11
PROCESS IAD MAC (with RRP data)
2627.12
PROCESS IAD MAC (without RRP data)
S28 – Waiting for IAD MAC Results


S2627 - C
2627.13
'Message ID' in UIRD1 :- ERROR – OTHER CARD
'Status' in UIRD1 := NOT READY
'Hold Time' in UIRD1 := Message Hold Time
2627.14
OUT (END APPLICATION,
Start: N/A
Exit Kernel

Note that the symbols 2627.3, 2627.4, 2627.5, and 2627.6 are only implemented for the DE/DS Implementation Option.

2627.3

```
FOR every T in Tags To Read Yet
{
    IF      [IsNotEmpty(T)]
    THEN
            AddToList(GetTLV(T), Data To Send)
            RemoveFromList(T, Tags To Read Yet)
    ENDIF
}
```

2627.8

Initialise Extended SDA Tag List Related Data with an empty string

```
FOR every T in Extended SDA Tag List
{
    IF      [IsNotEmpty(T)]
    THEN
            Extended SDA Tag List Related Data := Extended SDA Tag List Related Data
            || GetTLV(T)
    ELSE
            Extended SDA Tag List Related Data := Extended SDA Tag List Related Data
            || T || '00'
    ENDIF
}
```

2627.11

Modify GENERATE AC Response Message Data Field as follows:

- Maintain the order of the data objects as returned in the response to the GENERATE AC
- Remove tag '77' and length
- Remove Application Cryptogram, Enhanced Data Authentication MAC
- If 'Version' in Card Qualifier = VERSION 1, remove Issuer Application Data

Send PROCESS IAD MAC(

PDOL Values || CDOL1 Related Data || Terminal Relay Resistance Entropy || Last ERRD Response || modified GENERATE AC Response Message Data Field,

Application Interchange Profile,

Extended SDA Tag List Related Data) Signal

2627.12

Modify GENERATE AC Response Message Data Field as follows:

- Maintain the order of the data objects as returned in the response to the GENERATE AC
- Remove tag '77' and length
- Remove Application Cryptogram, Enhanced Data Authentication MAC
- If 'Version' in Card Qualifier = VERSION 1, remove Issuer Application Data

Send PROCESS IAD MAC(

PDOL Values || CDOL1 Related Data || modified GENERATE AC Response Message Data Field,

Application Interchange Profile,

Extended SDA Tag List Related Data) Signal

2627.14

CreateDiscretionaryData ()

'Status' in Outcome Parameter Set := END APPLICATION

SET 'UI Request on Outcome Present' in Outcome Parameter Set

Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
GetTLV(TagOf(Discretionary Data)),
GetTLV(TagOf(User Interface Request Data 1))) Signal

6.3.16 State 28 – Waiting for IAD MAC Results

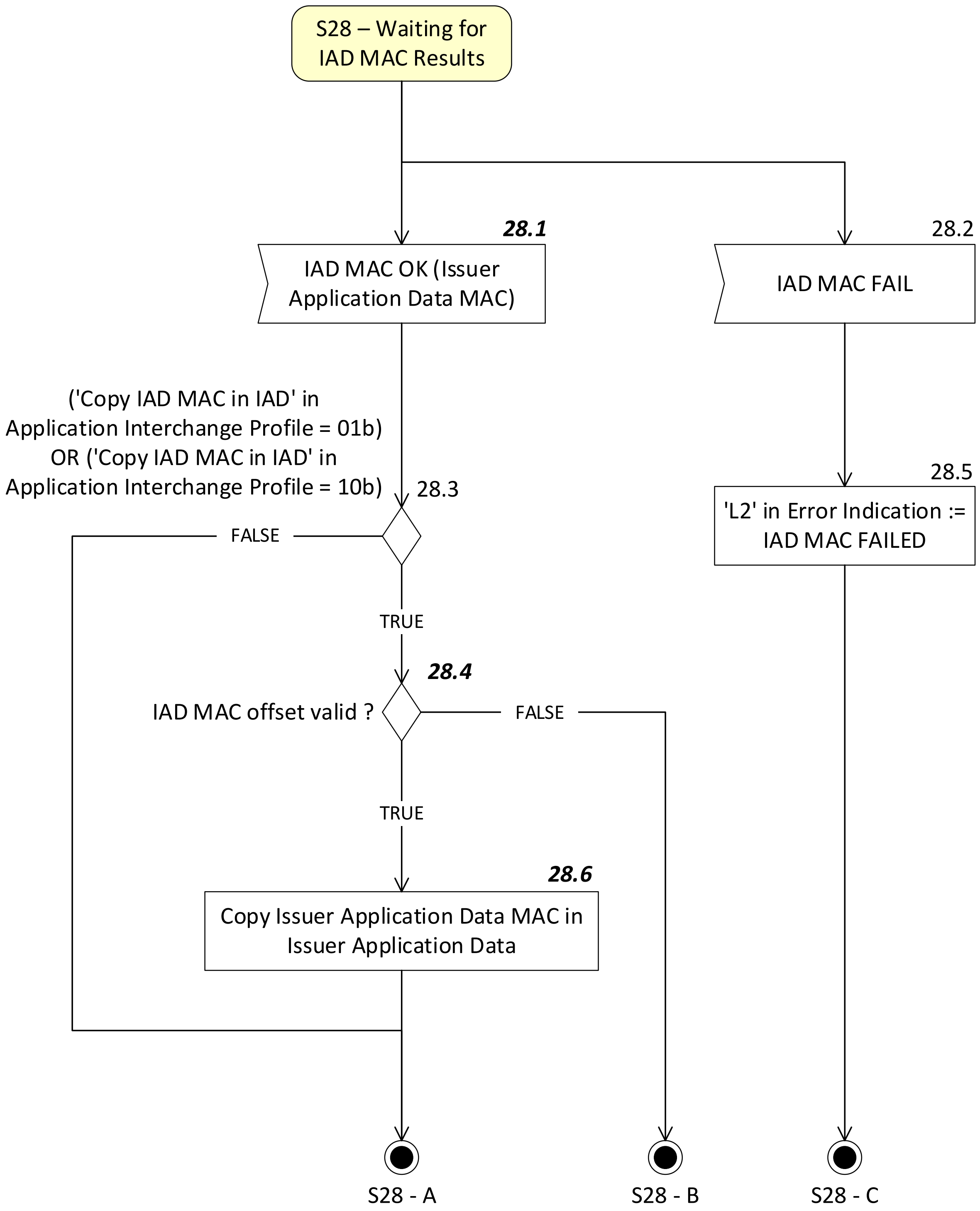
Table 6.11 shows the local variables used in `S28 – Waiting for IAD MAC Results`.

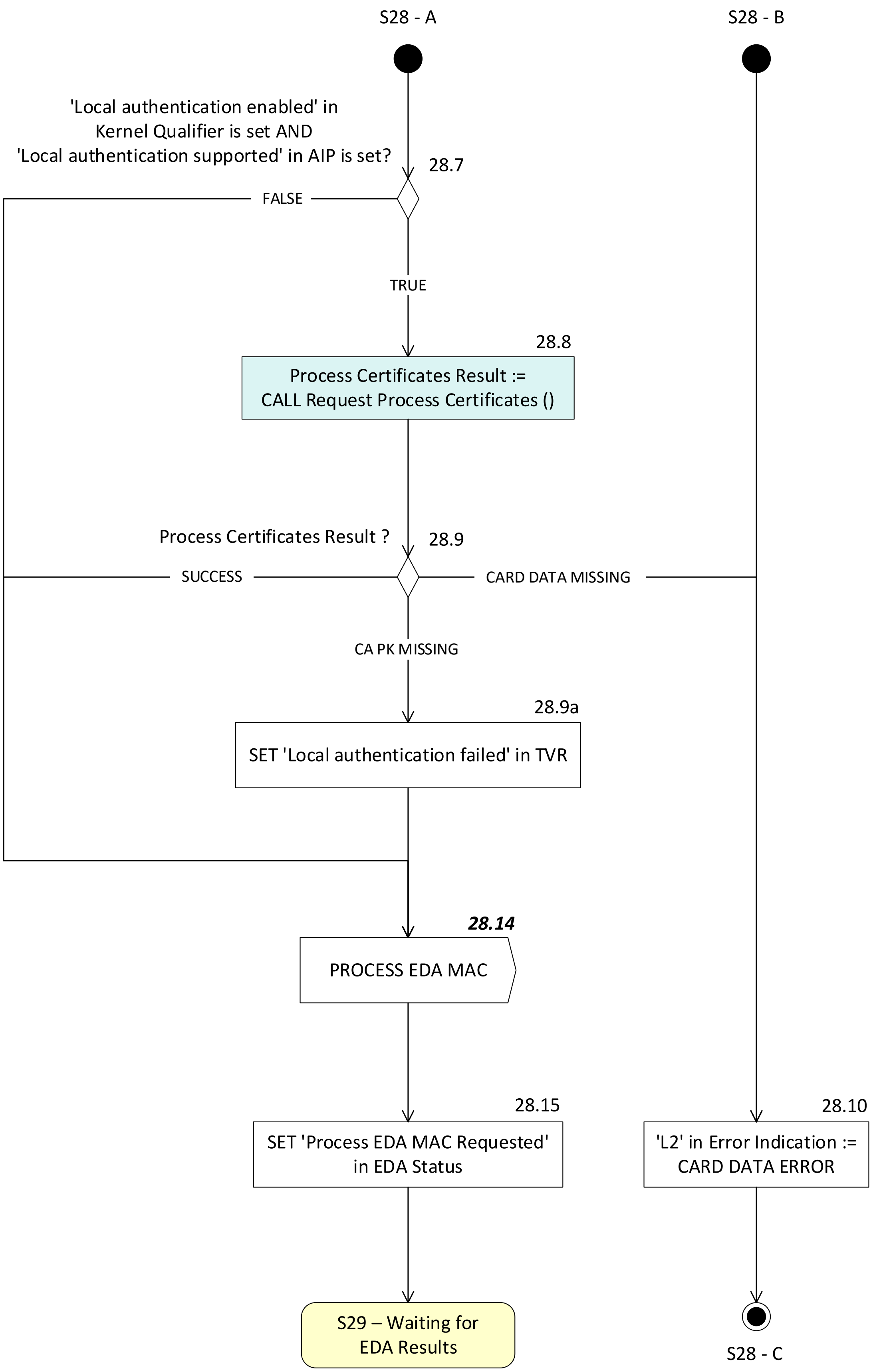
Table 6.11—State 28 Local Variables

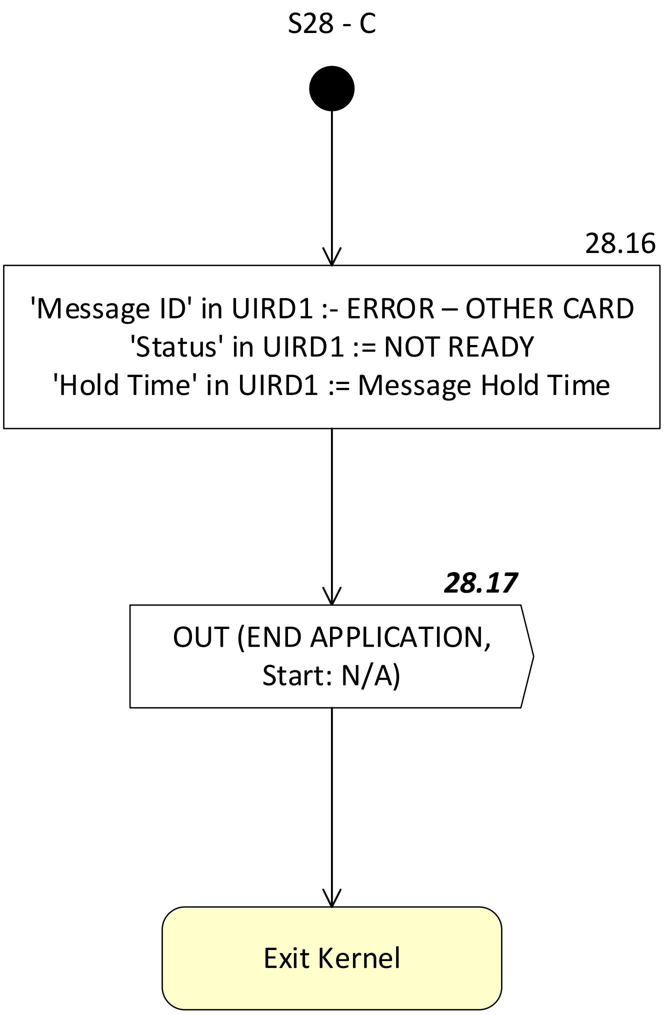
Name	Length	Format	Description
Process Certificates Result	1	b	Boolean used to store result of Request Process Certificates procedure

Figure 6.17 shows the flow diagram of `S28 – Waiting for IAD MAC Results`. Symbols in this diagram are labelled 28.X.

Figure 6.17—State 28 Flow Diagram




S28 - A
S28 - B
'Local authentication enabled' in Kernel Qualifier is set AND 'Local authentication supported' in AIP is set?
28.7
FALSE
TRUE
28.8
Process Certificates Result := CALL Request Process Certificates ()
Process Certificates Result ?
28.9
SUCCESS
CARD DATA MISSING
CA PK MISSING
28.9a
SET 'Local authentication failed' in TVR
28.14
PROCESS EDA MAC
28.15
SET 'Process EDA MAC Requested' in EDA Status
28.10
'L2' in Error Indication := CARD DATA ERROR
S29 – Waiting for EDA Results
S28 - C


S28 - C
28.16
'Message ID' in UIRD1 :- ERROR – OTHER CARD
'Status' in UIRD1 := NOT READY
'Hold Time' in UIRD1 := Message Hold Time
28.17
OUT (END APPLICATION,
Start: N/A)
Exit Kernel

28.1

Store Issuer Application Data MAC in TLV Database

28.4

IAD MAC offset valid ?:

[
 (('Copy IAD MAC in IAD' in Application Interchange Profile = 01b)
   AND (Default IAD MAC Offset + 8 ≤ GetLength(TagOf(Issuer Application Data))))
 OR
 (('Copy IAD MAC in IAD' in Application Interchange Profile = 10b)
   AND IsNotEmpty(IAD MAC Offset)
   AND (IAD MAC Offset + 8 ≤ GetLength(TagOf(Issuer Application Data))))
]

28.6

IF ['Copy IAD MAC in IAD' in Application Interchange Profile = 01b]
THEN
 Issuer Application Data[Default IAD MAC Offset+1 : Default IAD MAC Offset+8] := Issuer Application Data MAC
ELSE IF ['Copy IAD MAC in IAD' in Application Interchange Profile = 10b]
THEN
 Issuer Application Data[IAD MAC Offset+1 : IAD MAC Offset+8] := Issuer Application Data MAC
ENDIF

28.14

IF ['Version' in Card Qualifier = VERSION 1]
THEN
 Send PROCESS EDA MAC(
  Enhanced Data Authentication MAC,
  Application Cryptogram || Issuer Application Data) Signal
ELSE
 Send PROCESS EDA MAC(
  Enhanced Data Authentication MAC,
  Application Cryptogram || Issuer Application Data MAC) Signal
ENDIF

28.17

CreateDiscretionaryData ()

'Status' in Outcome Parameter Set := END APPLICATION

SET 'UI Request on Outcome Present' in Outcome Parameter Set

Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
GetTLV(TagOf(Discretionary Data)),
GetTLV(TagOf(User Interface Request Data 1))) Signal

6.3.17 State 29 – Waiting for EDA Results

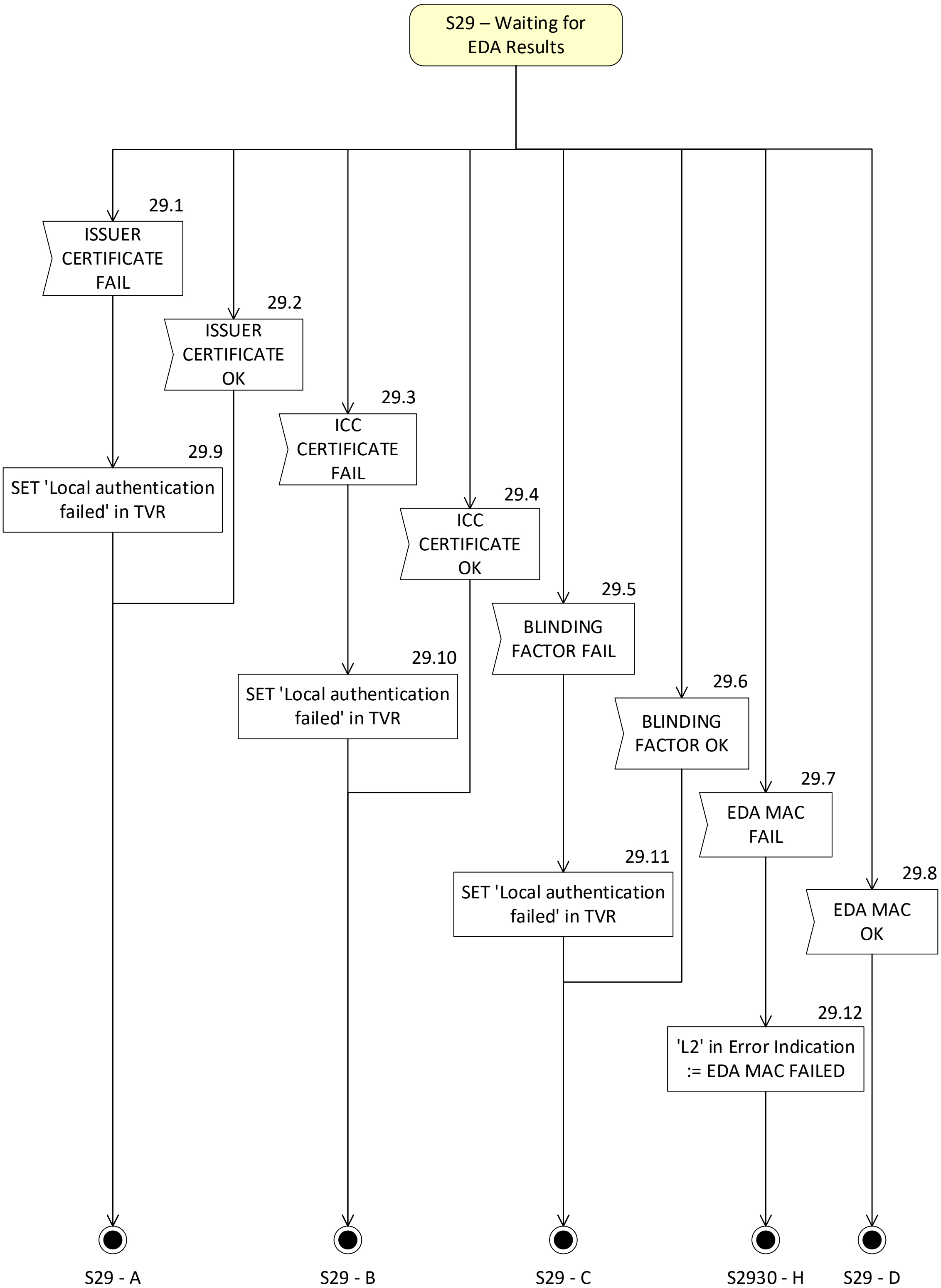
Table 6.12 shows the local variables used in `S29 - Waiting for EDA Results`.

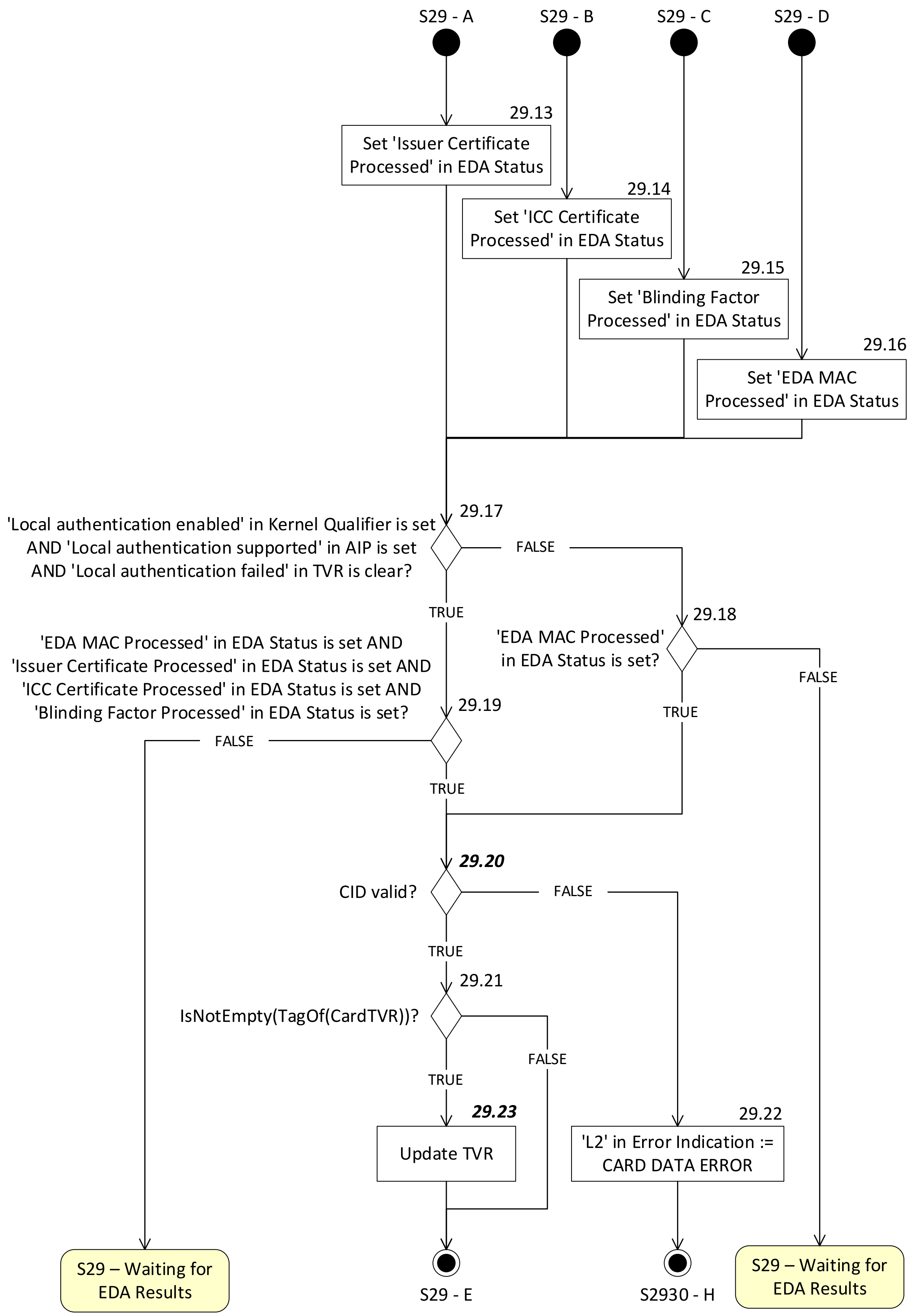
Table 6.12—State 29 Local Variables

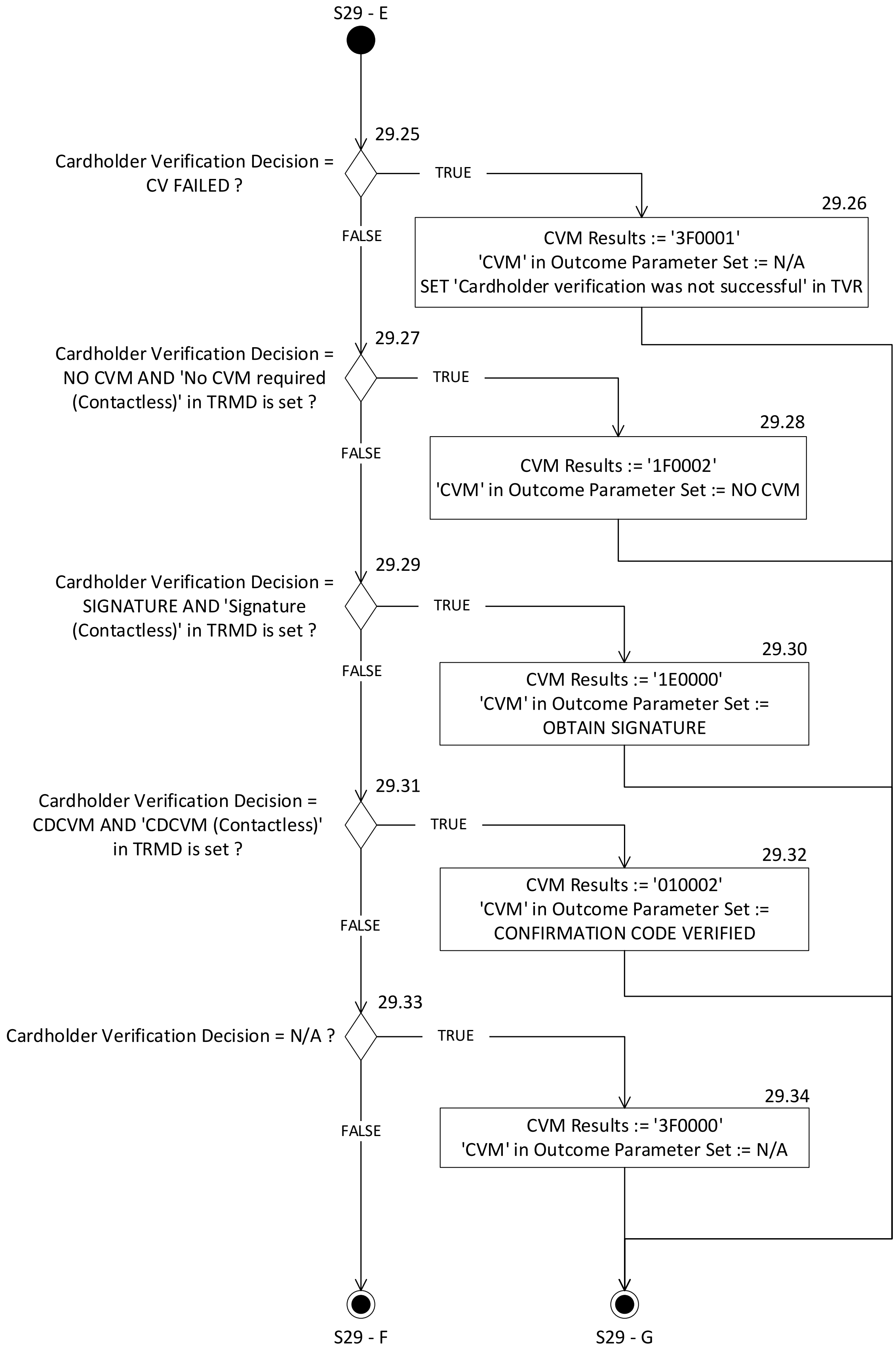
Name	Length	Format	Description
Restart Message Allowed	1	b	Boolean used to indicate if 'Message Identifier' in Restart Indicator is included in Message Identifiers On Restart

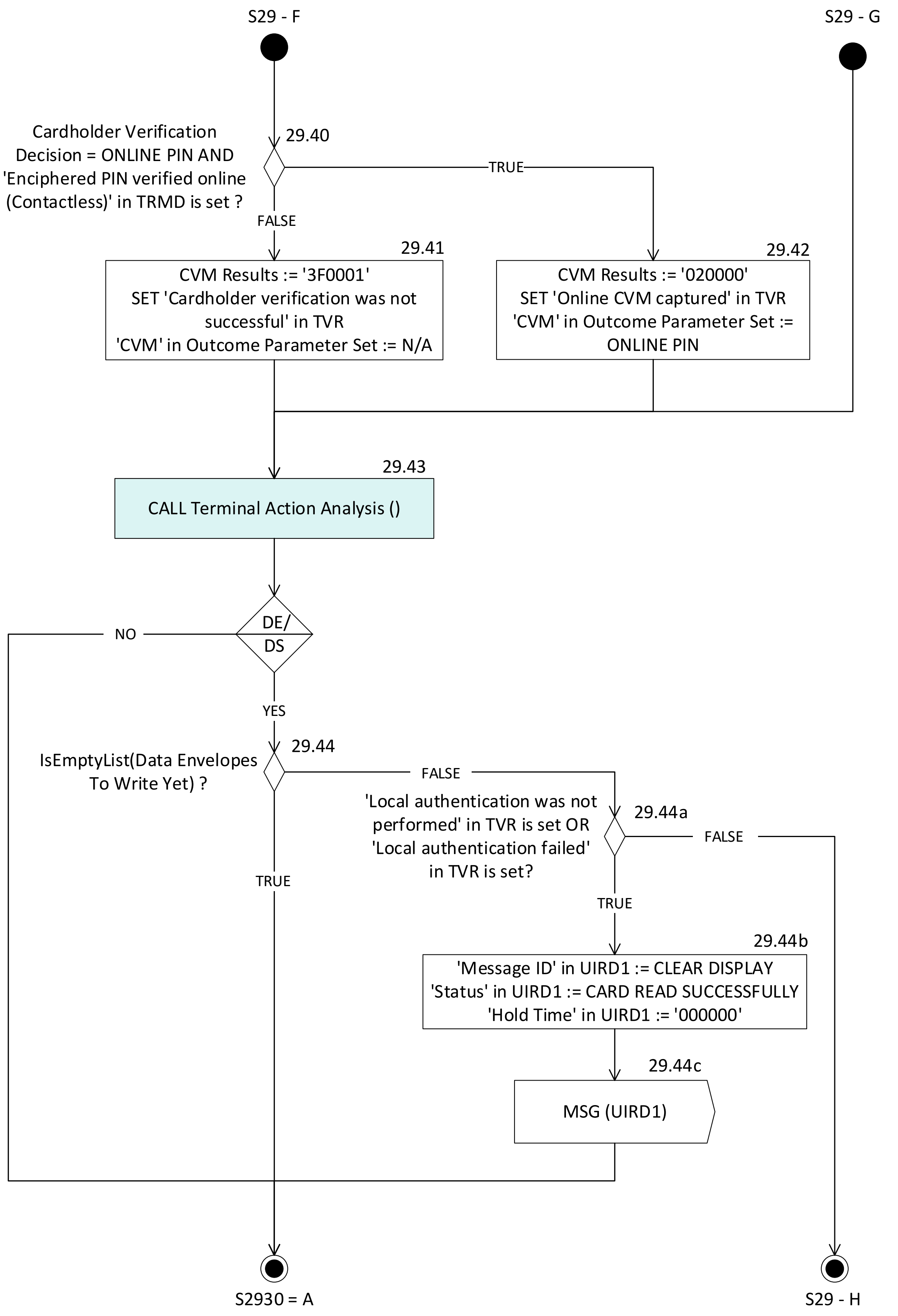
Figure 6.18 shows the flow diagram of `S29 - Waiting for EDA Results`. Symbols in this diagram are labelled 29.X.

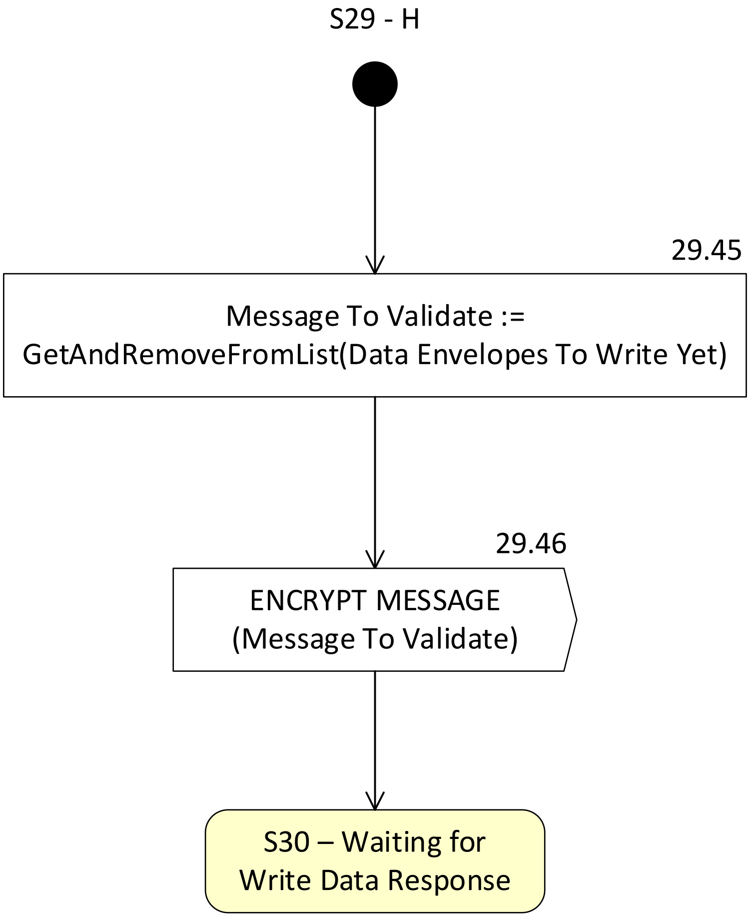
Figure 6.18—State 29 Flow Diagram




S29 - A
S29 - B
S29 - C
S29 - D
29.13
Set 'Issuer Certificate Processed' in EDA Status
29.14
Set 'ICC Certificate Processed' in EDA Status
29.15
Set 'Blinding Factor Processed' in EDA Status
29.16
Set 'EDA MAC Processed' in EDA Status
29.17
'Local authentication enabled' in Kernel Qualifier is set AND 'Local authentication supported' in AIP is set AND 'Local authentication failed' in TVR is clear?
FALSE
TRUE
29.18
'EDA MAC Processed' in EDA Status is set?
FALSE
TRUE
'EDA MAC Processed' in EDA Status is set AND 'Issuer Certificate Processed' in EDA Status is set AND 'ICC Certificate Processed' in EDA Status is set AND 'Blinding Factor Processed' in EDA Status is set?
29.19
FALSE
TRUE
29.20
CID valid?
FALSE
TRUE
29.21
IsNotEmpty(TagOf(CardTVR))?
FALSE
TRUE
29.23
Update TVR
29.22
'L2' in Error Indication := CARD DATA ERROR
S29 – Waiting for EDA Results
S29 - E
S2930 - H
S29 – Waiting for EDA Results


S29 - E
29.25
Cardholder Verification Decision = CV FAILED ?
TRUE
FALSE
29.26
CVM Results := '3F0001'
'CVM' in Outcome Parameter Set := N/A
SET 'Cardholder verification was not successful' in TVR
29.27
Cardholder Verification Decision = NO CVM AND 'No CVM required (Contactless)' in TRMD is set ?
TRUE
FALSE
29.28
CVM Results := '1F0002'
'CVM' in Outcome Parameter Set := NO CVM
29.29
Cardholder Verification Decision = SIGNATURE AND 'Signature (Contactless)' in TRMD is set ?
TRUE
FALSE
29.30
CVM Results := '1E0000'
'CVM' in Outcome Parameter Set := OBTAIN SIGNATURE
29.31
Cardholder Verification Decision = CDCVM AND 'CDCVM (Contactless)' in TRMD is set ?
TRUE
FALSE
29.32
CVM Results := '010002'
'CVM' in Outcome Parameter Set := CONFIRMATION CODE VERIFIED
29.33
Cardholder Verification Decision = N/A ?
TRUE
FALSE
29.34
CVM Results := '3F0000'
'CVM' in Outcome Parameter Set := N/A
S29 - F
S29 - G


S29 - F
S29 - G
29.40
Cardholder Verification Decision = ONLINE PIN AND 'Enciphered PIN verified online (Contactless)' in TRMD is set ?
TRUE
FALSE
29.41
CVM Results := '3F0001'
SET 'Cardholder verification was not successful' in TVR
'CVM' in Outcome Parameter Set := N/A
29.42
CVM Results := '020000'
SET 'Online CVM captured' in TVR
'CVM' in Outcome Parameter Set := ONLINE PIN
29.43
CALL Terminal Action Analysis ()
DE/ DS
NO
YES
29.44
IsEmptyList(Data Envelopes To Write Yet) ?
FALSE
TRUE
29.44a
'Local authentication was not performed' in TVR is set OR 'Local authentication failed' in TVR is set?
FALSE
TRUE
29.44b
'Message ID' in UIRD1 := CLEAR DISPLAY
'Status' in UIRD1 := CARD READ SUCCESSFULLY
'Hold Time' in UIRD1 := '000000'
29.44c
MSG (UIRD1)
S2930 = A
S29 - H


S29 - H
29.45
Message To Validate :=
GetAndRemoveFromList(Data Envelopes To Write Yet)
29.46
ENCRYPT MESSAGE
(Message To Validate)
S30 – Waiting for
Write Data Response

Note that the symbols 29.44, 29.44a, 29.44b, 29.44c, 29.45, and 29.46 are only implemented for the DE/DS Implementation Option.

29.20

CID valid ?:

[(('AC type' in Cryptogram Information Data = TC)
AND ('AC type' in Reference Control Parameter = TC))
OR
(('AC type' in Cryptogram Information Data = ARQC)
AND
(('AC type' in Reference Control Parameter = TC)
OR ('AC type' in Reference Control Parameter = ARQC)))
OR
('AC type' in Cryptogram Information Data = AAC)]

29.23

Terminal Verification Results := (Terminal Verification Results AND Kernel Reserved TVR Mask) OR (Card TVR AND NOT(Kernel Reserved TVR Mask))

6.3.18 State 30 – Waiting for Write Data Response

State 30 is a state specific to data storage processing and is only implemented if DE/DS Implementation Option is used.

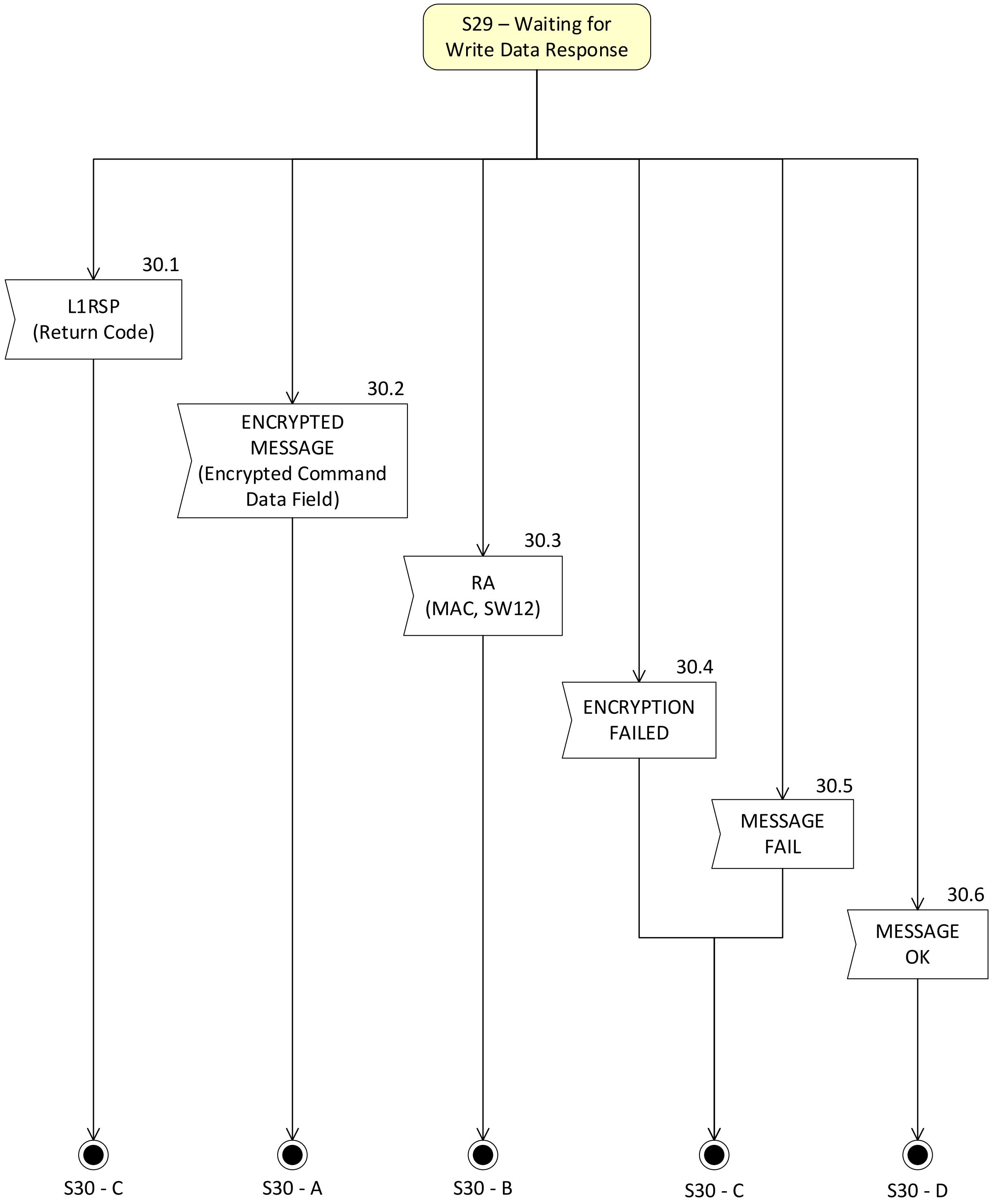
Table 6.13 shows the local variables used in `S30 – Waiting for Write Data Response`.

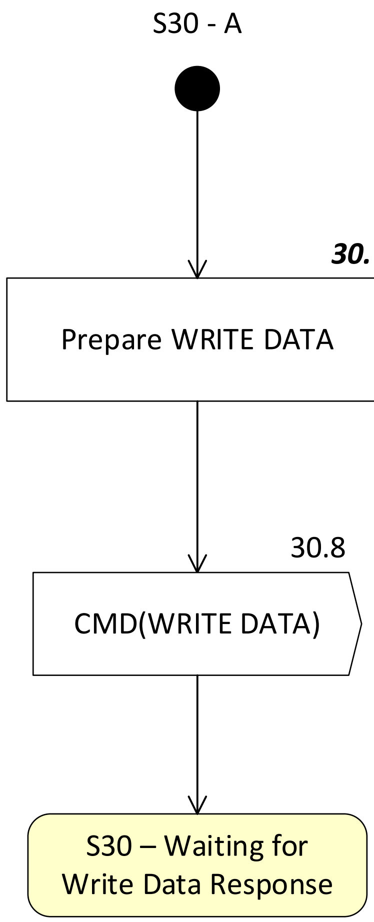
Table 6.13—State 30 Local Variables

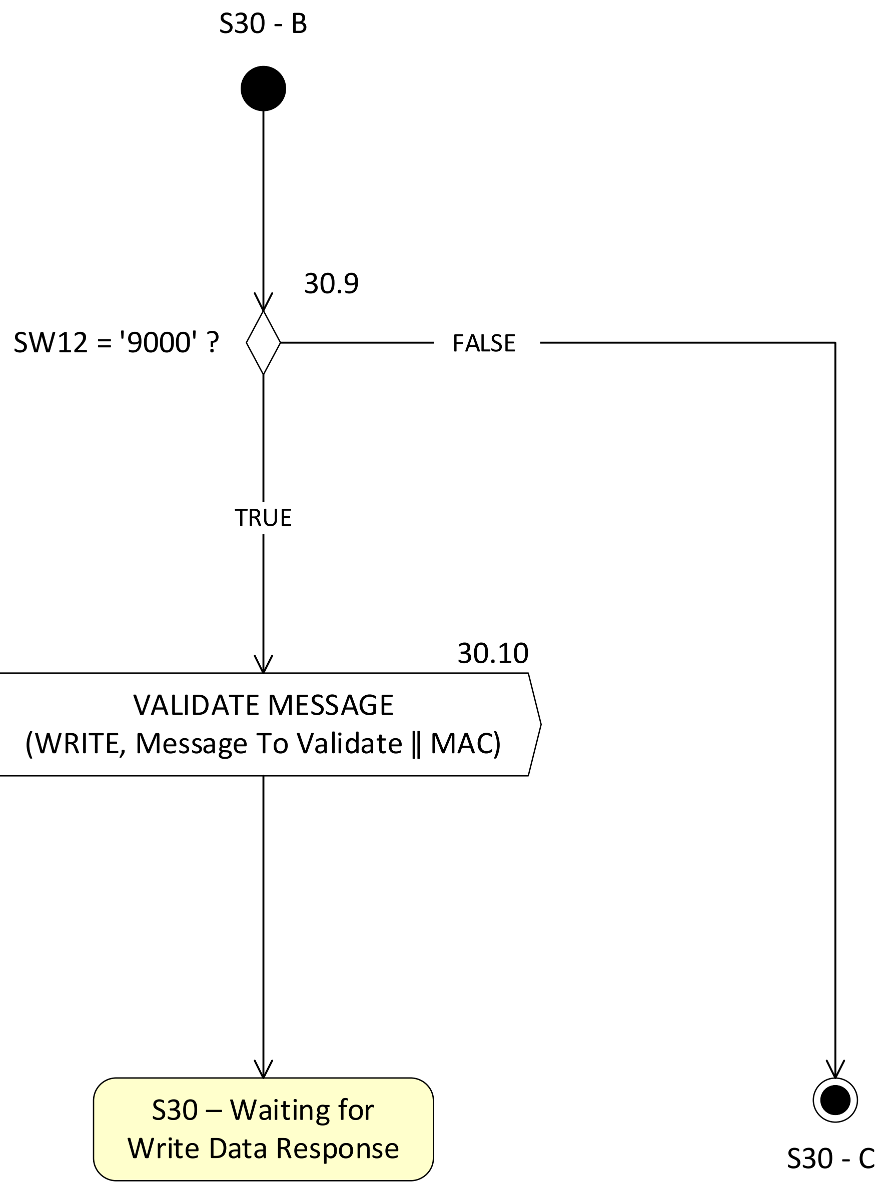
Name	Length	Format	Description
SW12	2	b	Status bytes
Return Code	1	b	Value returned with L1RSP Signal (TIMEOUT ERROR, PROTOCOL ERROR, TRANSMISSION ERROR)
MAC	8	b	8-byte MAC returned in WRITE DATA response.
Encrypted Command Data Field	var. up to 248	b	Encrypted WRITE DATA command data field returned by ENCRYPTED MESSAGE Signal
Restart Message Allowed	1	b	Boolean used to indicate if 'Message Identifier' in Restart Indicator is included in Message Identifiers On Restart

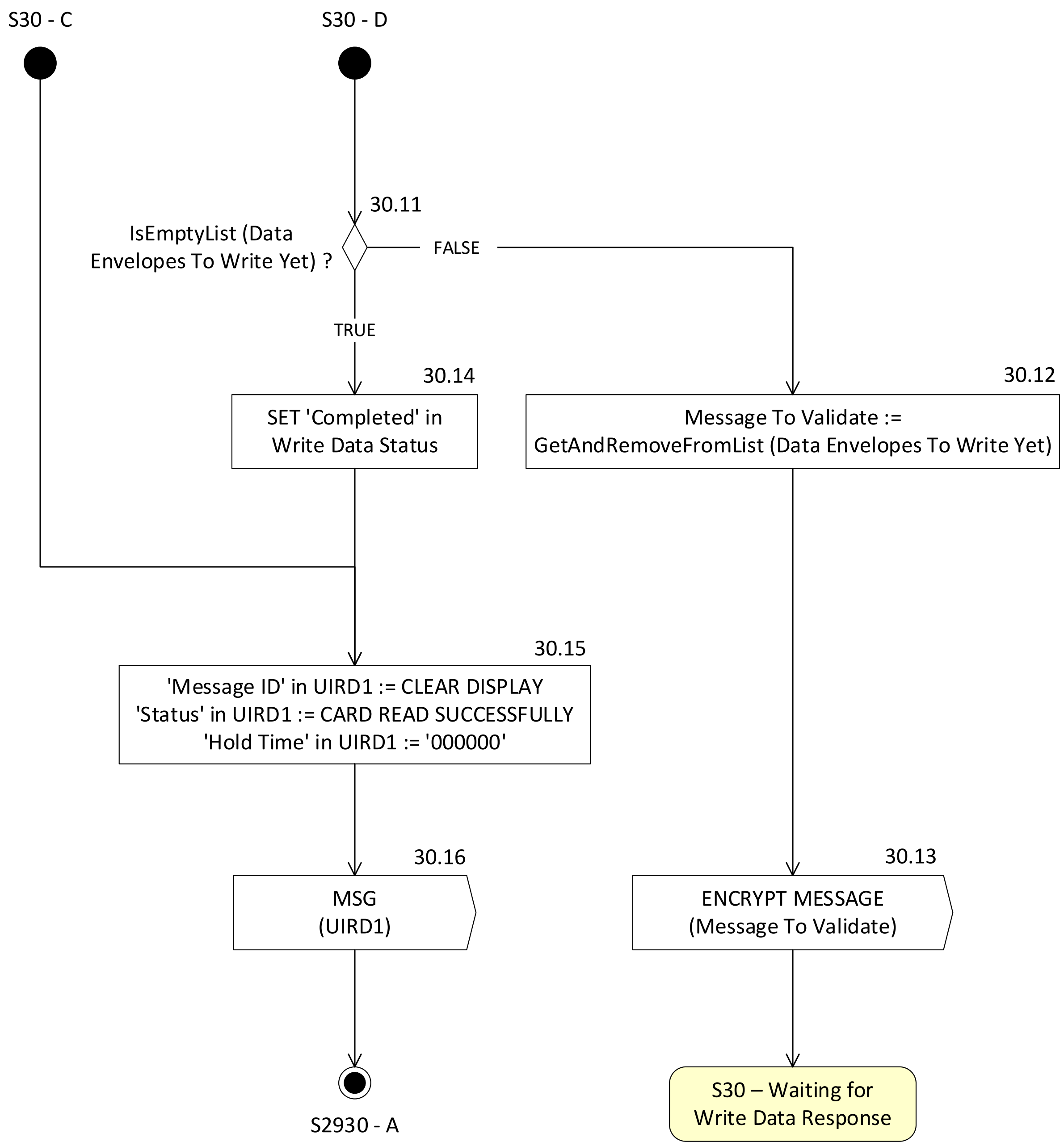
Figure 6.19 shows the flow diagram of `S30 – Waiting for Write Data Response`. Symbols in this diagram are labelled 30.X.

Figure 6.19—State 30 Flow Diagram




S30 - A
30.7
Prepare WRITE DATA
30.8
CMD(WRITE DATA)
S30 – Waiting for Write Data Response


S30 - B
30.9
SW12 = '9000' ?
FALSE
TRUE
30.10
VALIDATE MESSAGE
(WRITE, Message To Validate || MAC)
S30 – Waiting for
Write Data Response
S30 - C


S30 - C
S30 - D
30.11
IsEmptyList (Data Envelopes To Write Yet) ?
FALSE
TRUE
30.14
SET 'Completed' in Write Data Status
30.12
Message To Validate := GetAndRemoveFromList (Data Envelopes To Write Yet)
30.15
'Message ID' in UIRD1 := CLEAR DISPLAY
'Status' in UIRD1 := CARD READ SUCCESSFULLY
'Hold Time' in UIRD1 := '000000'
30.16
MSG (UIRD1)
30.13
ENCRYPT MESSAGE (Message To Validate)
S2930 - A
S30 – Waiting for Write Data Response

30.7

IF [IsNotEmptyList(Data Envelopes To Write Yet)]

THEN

Prepare WRITE DATA command for Encrypted Command Data Field with P1P2 is '0080' as specified in section 5.7

ELSE

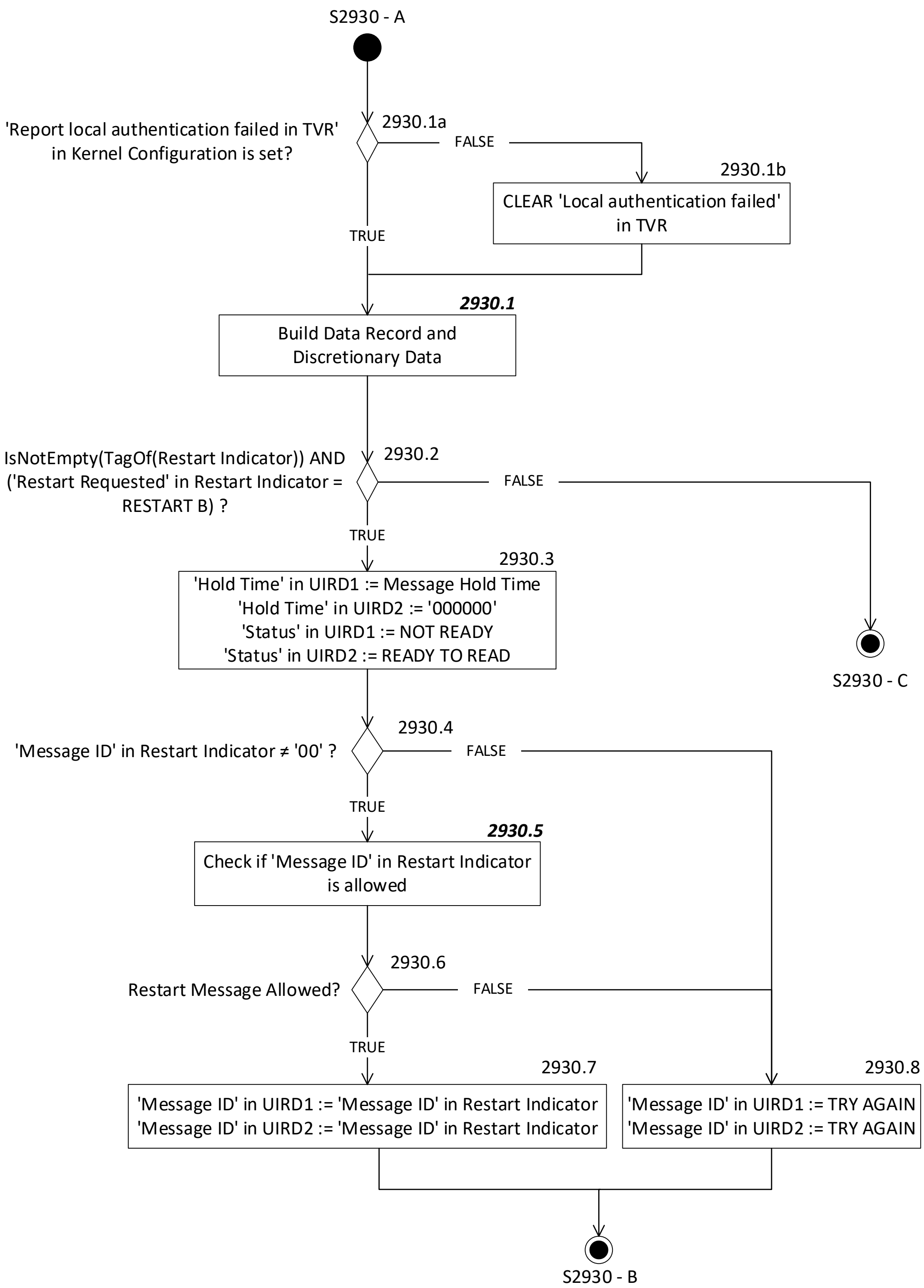
Prepare WRITE DATA command for Encrypted Command Data Field with P1P2 is '0000' as specified in section 5.7

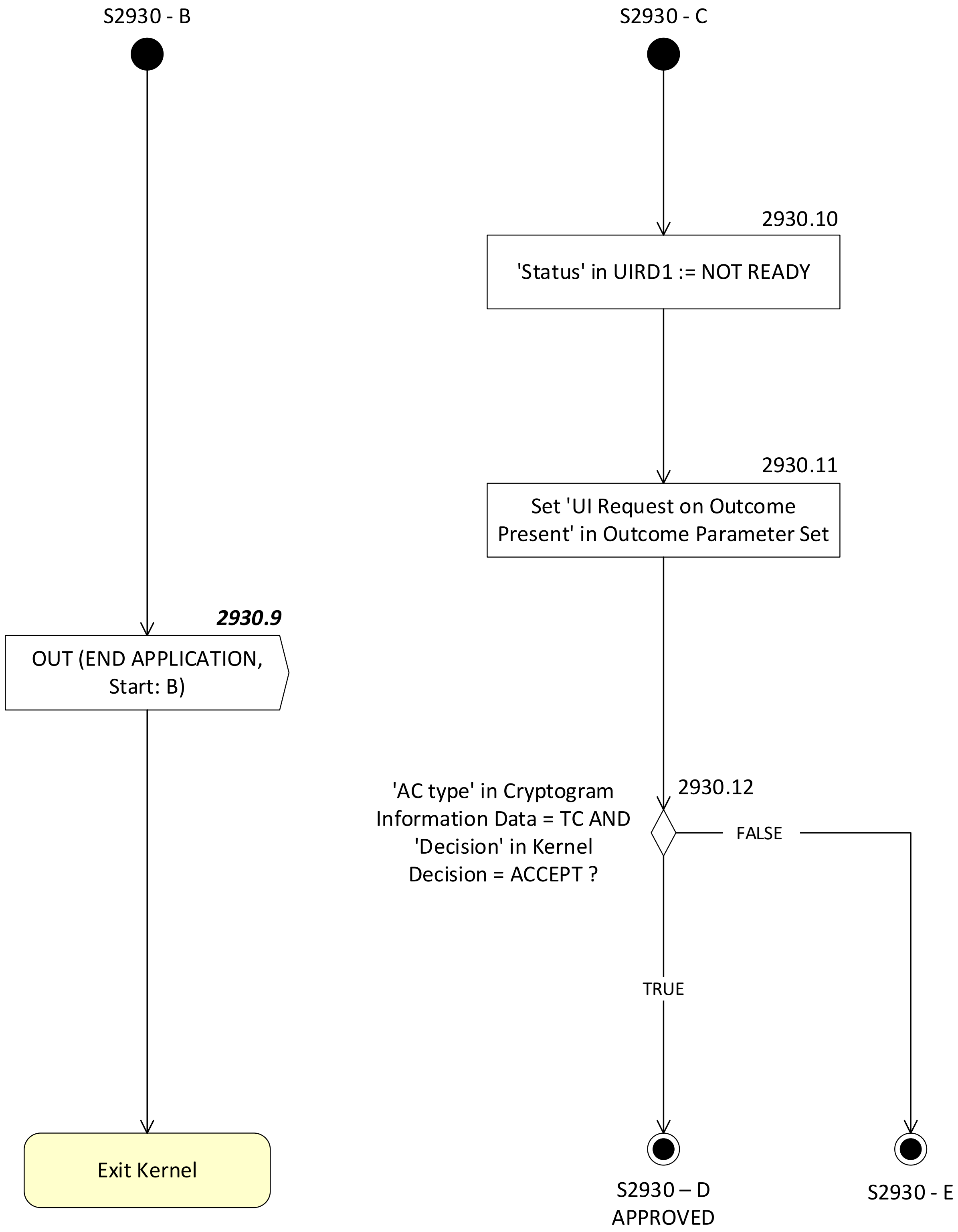
ENDIF

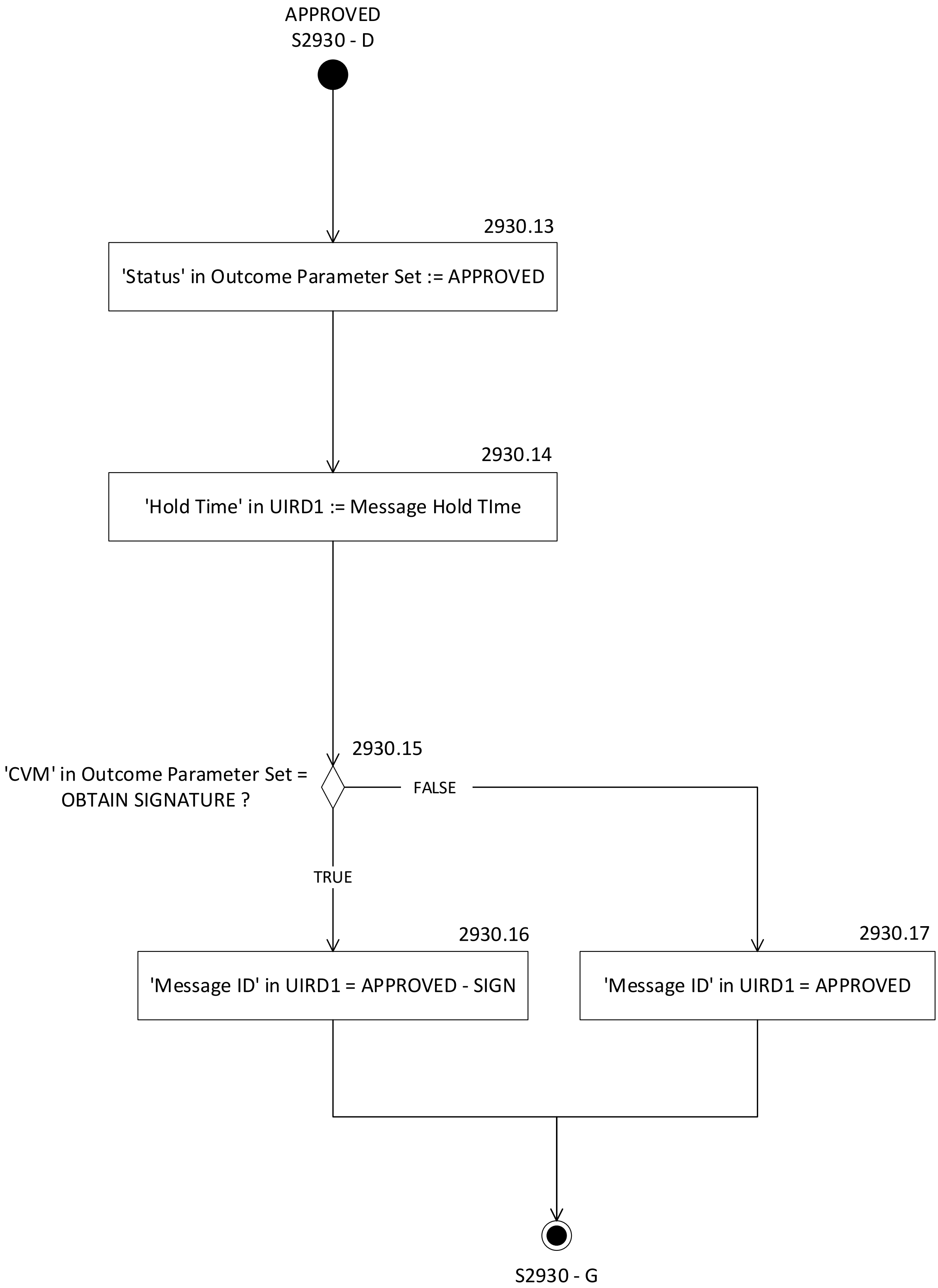
6.3.19 States 29 and 30 – Common Processing

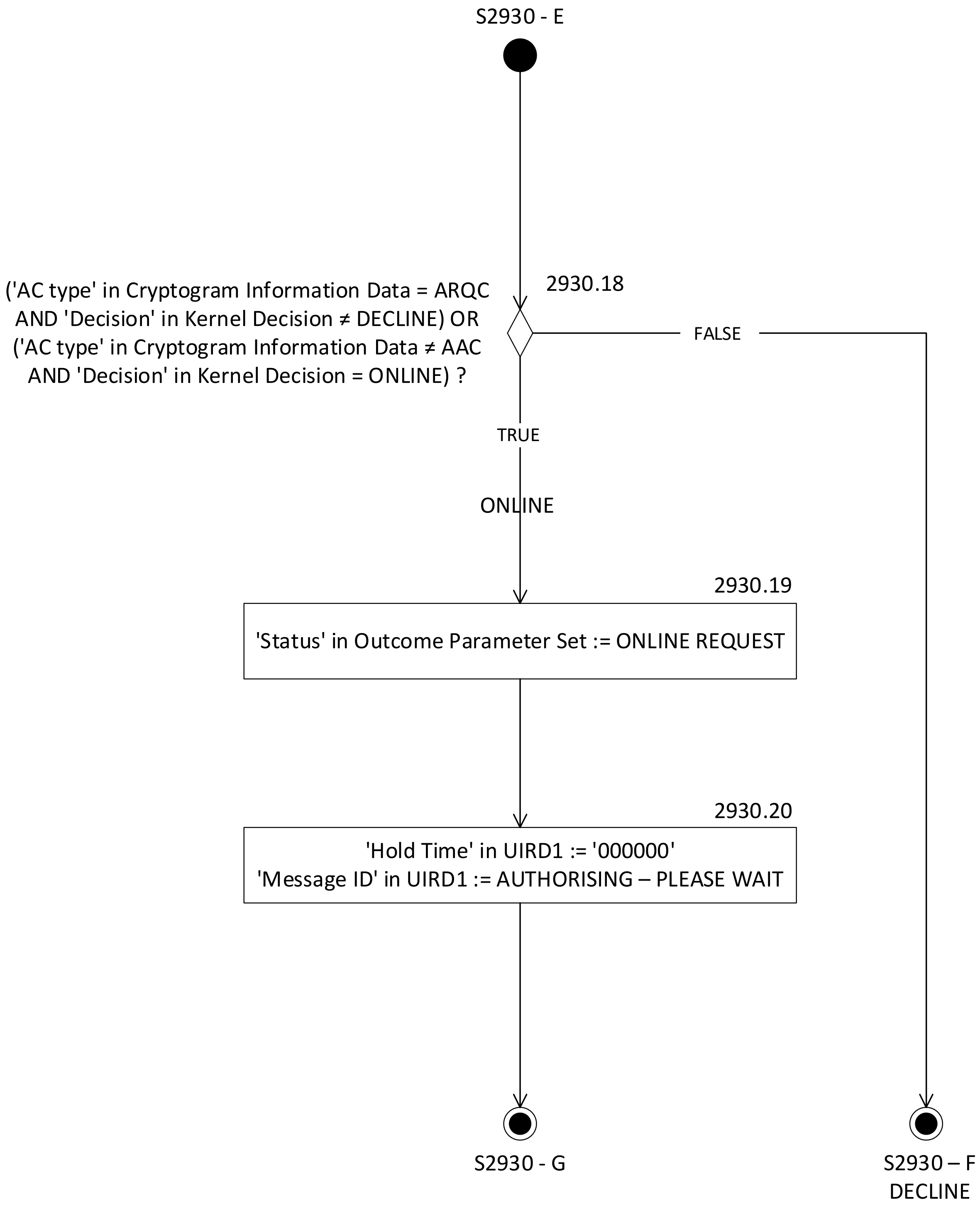
Figure 6.20 shows the flow diagram for common processing between states 29 and 30. Symbols in this diagram are labelled 2930.X.

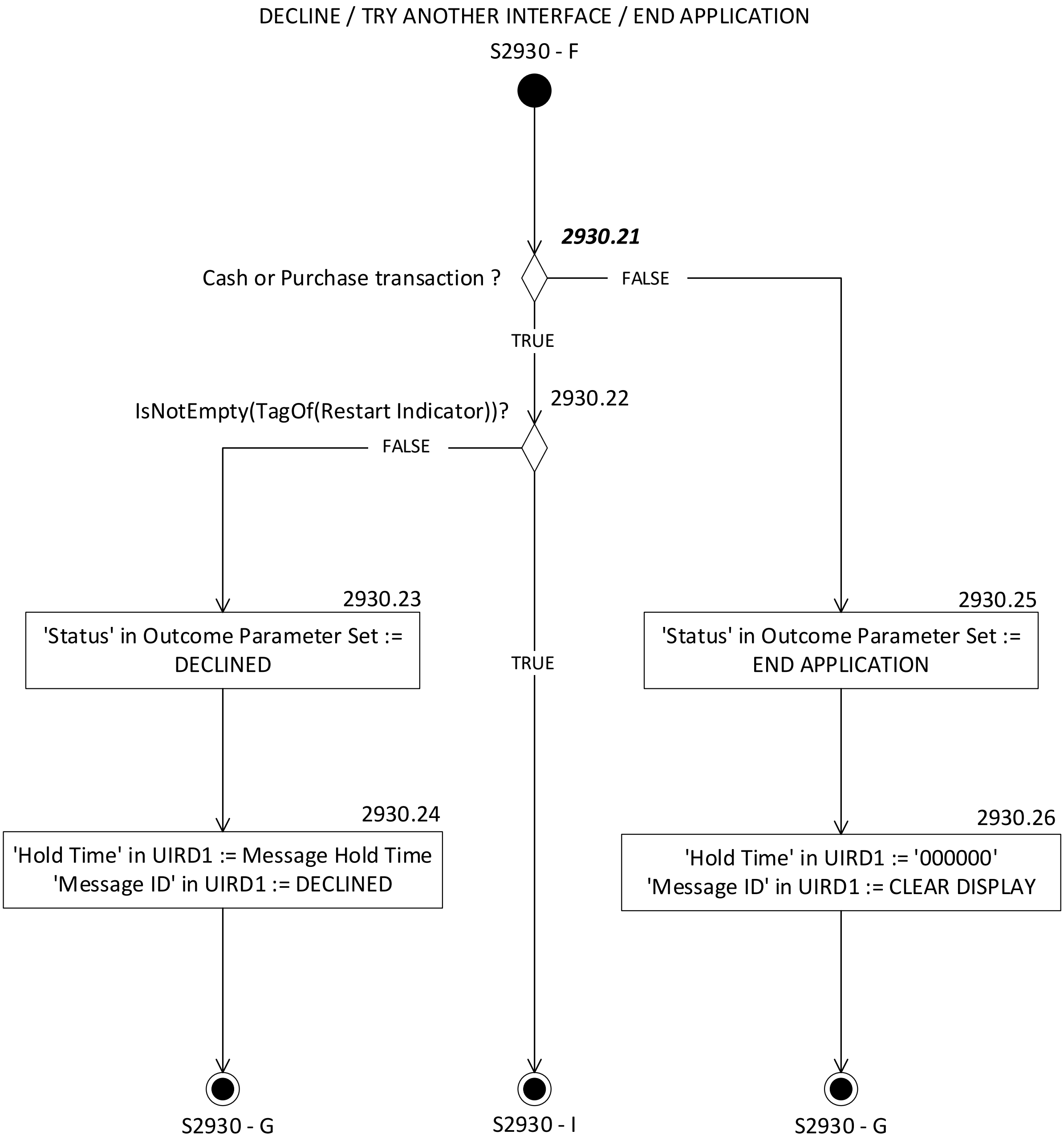
Figure 6.20—States 29 and 30 – Common Processing – Flow Diagram

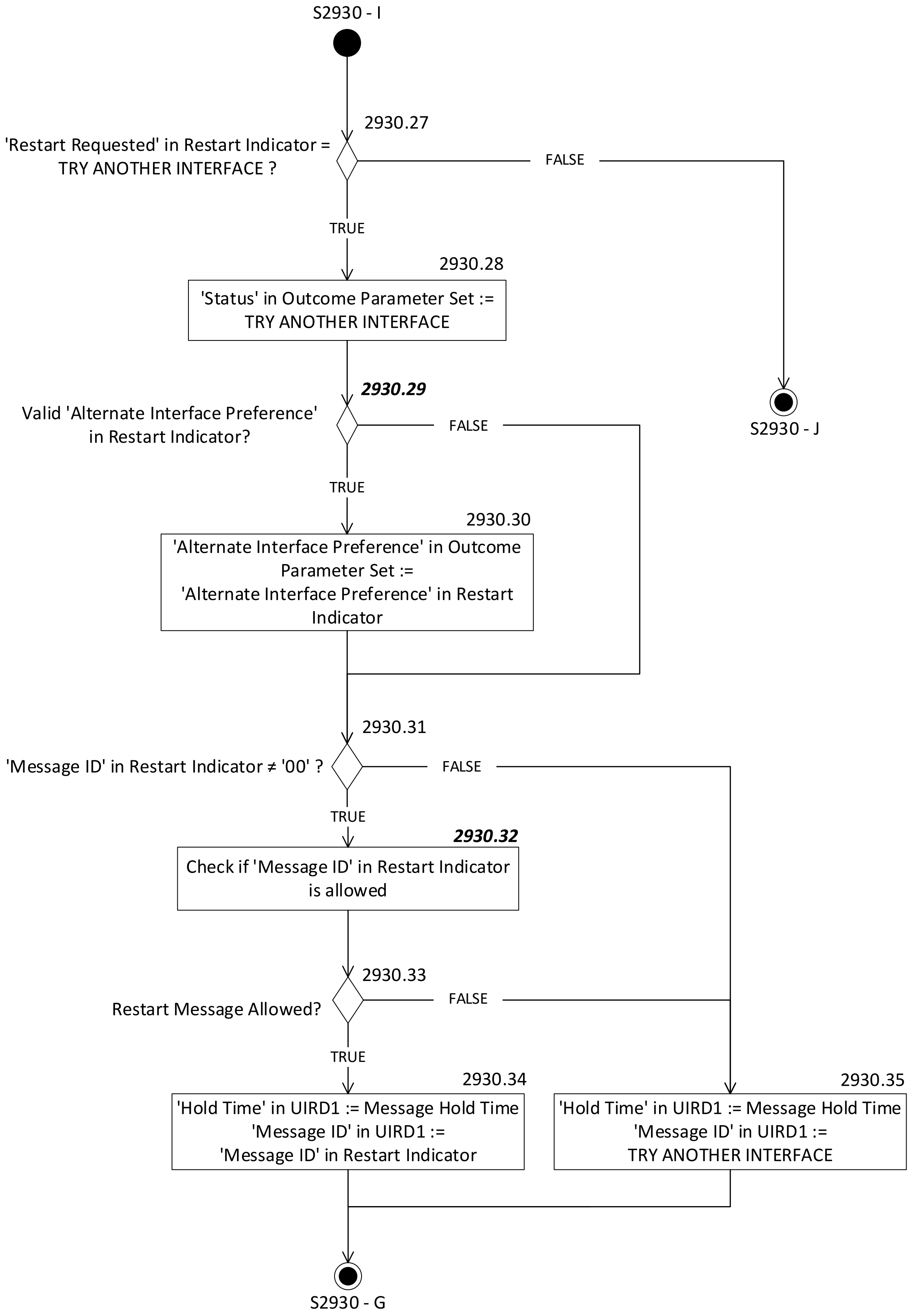


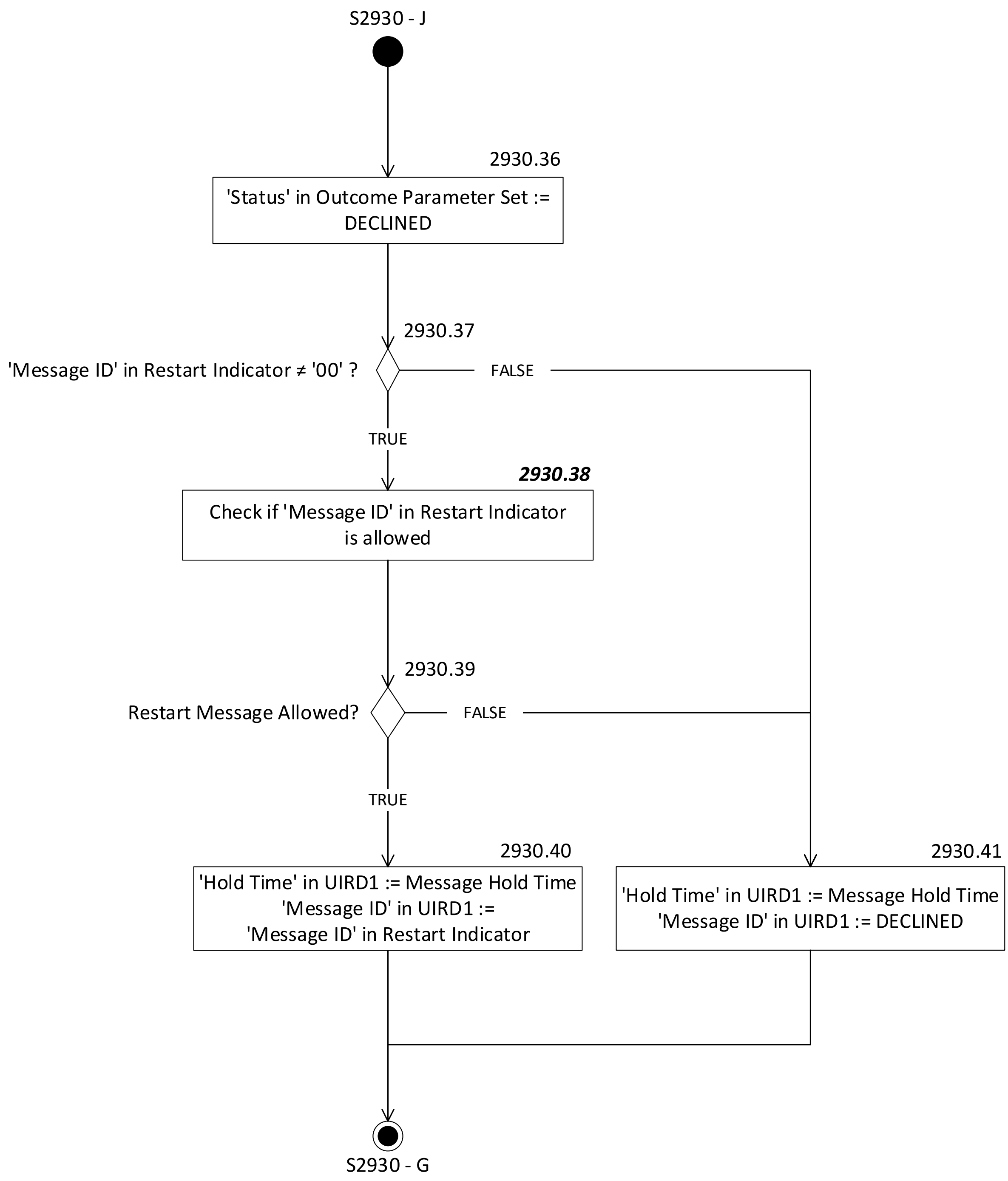

S2930 - B
S2930 - C
2930.10
'Status' in UIRD1 := NOT READY
2930.11
Set 'UI Request on Outcome Present' in Outcome Parameter Set
2930.9
OUT (END APPLICATION, Start: B)
'AC type' in Cryptogram Information Data = TC AND 'Decision' in Kernel Decision = ACCEPT ?
2930.12
FALSE
TRUE
Exit Kernel
S2930 – D APPROVED
S2930 - E

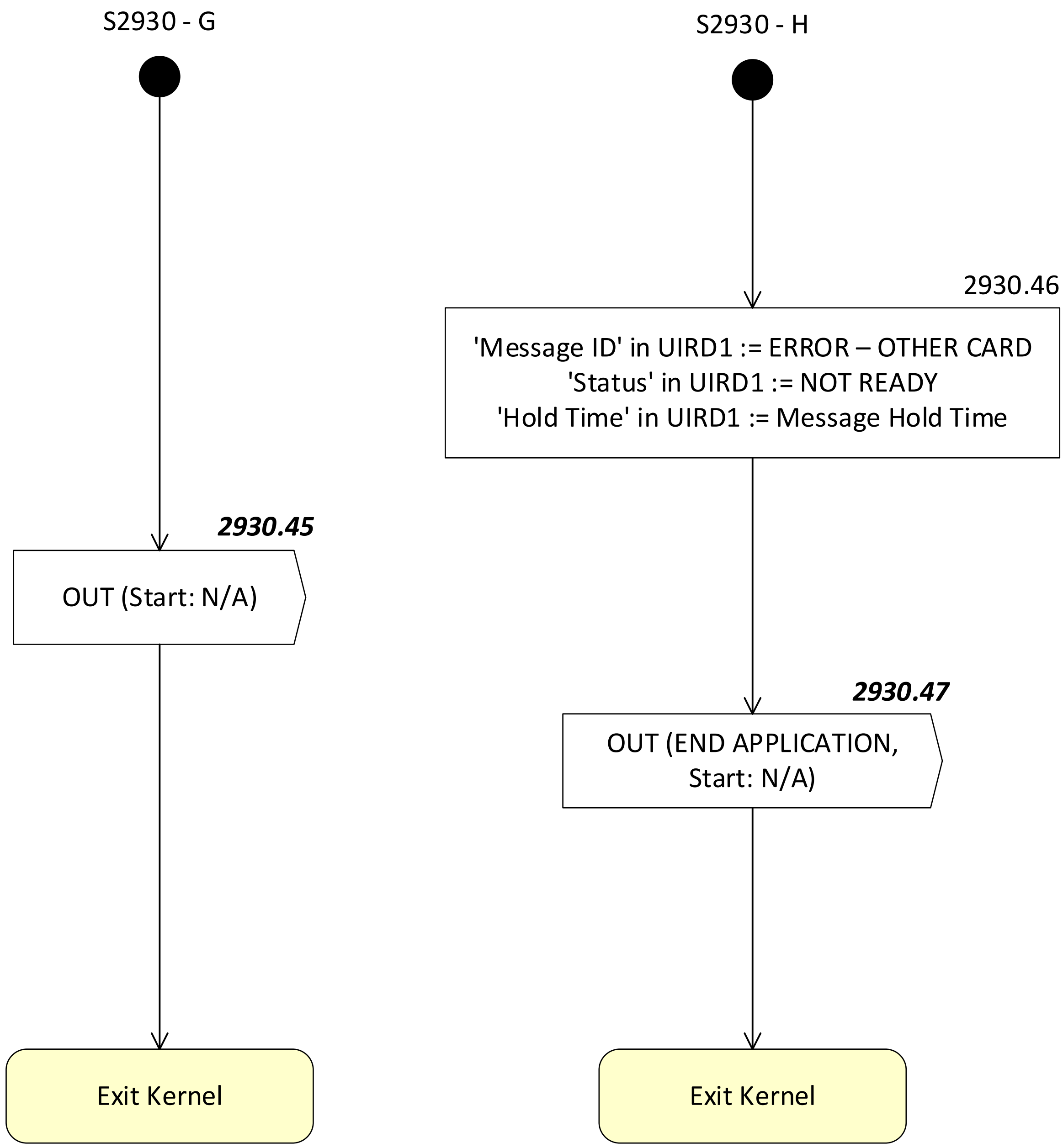

APPROVED
S2930 - D
2930.13
'Status' in Outcome Parameter Set := APPROVED
2930.14
'Hold Time' in UIRD1 := Message Hold TIme
2930.15
'CVM' in Outcome Parameter Set = OBTAIN SIGNATURE ?
FALSE
TRUE
2930.16
'Message ID' in UIRD1 = APPROVED - SIGN
2930.17
'Message ID' in UIRD1 = APPROVED
S2930 - G


S2930 - E
2930.18
('AC type' in Cryptogram Information Data = ARQC AND 'Decision' in Kernel Decision ≠ DECLINE) OR ('AC type' in Cryptogram Information Data ≠ AAC AND 'Decision' in Kernel Decision = ONLINE) ?
FALSE
TRUE
ONLINE
2930.19
'Status' in Outcome Parameter Set := ONLINE REQUEST
2930.20
'Hold Time' in UIRD1 := '000000'
'Message ID' in UIRD1 := AUTHORISING – PLEASE WAIT
S2930 - G
S2930 – F
DECLINE


DECLINE / TRY ANOTHER INTERFACE / END APPLICATION
S2930 - F
2930.21
Cash or Purchase transaction ?
FALSE
TRUE
2930.22
IsNotEmpty(TagOf(Restart Indicator))?
FALSE
TRUE
2930.23
'Status' in Outcome Parameter Set := DECLINED
2930.24
'Hold Time' in UIRD1 := Message Hold Time
'Message ID' in UIRD1 := DECLINED
2930.25
'Status' in Outcome Parameter Set := END APPLICATION
2930.26
'Hold Time' in UIRD1 := '000000'
'Message ID' in UIRD1 := CLEAR DISPLAY
S2930 - G
S2930 - I
S2930 - G


S2930 - I
2930.27
'Restart Requested' in Restart Indicator = TRY ANOTHER INTERFACE ?
FALSE
TRUE
2930.28
'Status' in Outcome Parameter Set := TRY ANOTHER INTERFACE
S2930 - J
2930.29
Valid 'Alternate Interface Preference' in Restart Indicator?
FALSE
TRUE
2930.30
'Alternate Interface Preference' in Outcome Parameter Set := 'Alternate Interface Preference' in Restart Indicator
2930.31
'Message ID' in Restart Indicator ≠ '00' ?
FALSE
TRUE
2930.32
Check if 'Message ID' in Restart Indicator is allowed
2930.33
Restart Message Allowed?
FALSE
TRUE
2930.34
'Hold Time' in UIRD1 := Message Hold Time
'Message ID' in UIRD1 := 'Message ID' in Restart Indicator
2930.35
'Hold Time' in UIRD1 := Message Hold Time
'Message ID' in UIRD1 := TRY ANOTHER INTERFACE
S2930 - G


S2930 - J
2930.36
'Status' in Outcome Parameter Set := DECLINED
2930.37
'Message ID' in Restart Indicator ≠ '00' ?
FALSE
TRUE
2930.38
Check if 'Message ID' in Restart Indicator is allowed
2930.39
Restart Message Allowed?
FALSE
TRUE
2930.40
'Hold Time' in UIRD1 := Message Hold Time
'Message ID' in UIRD1 := 'Message ID' in Restart Indicator
2930.41
'Hold Time' in UIRD1 := Message Hold Time
'Message ID' in UIRD1 := DECLINED
S2930 - G


S2930 - G
S2930 - H
2930.46
'Message ID' in UIRD1 := ERROR – OTHER CARD
'Status' in UIRD1 := NOT READY
'Hold Time' in UIRD1 := Message Hold Time
2930.45
OUT (Start: N/A)
2930.47
OUT (END APPLICATION, Start: N/A)
Exit Kernel
Exit Kernel

2930.1

SET 'Data Record Present' in Outcome Parameter Set
'Msg On Error' in Error Indication := N/A
CreateDataRecord ()
CreateDiscretionaryData ()

2930.5

```
Restart Message Allowed := FALSE
FOR i := 1 TO GetLength(TagOf(Message Identifiers On Restart))
{
    IF      [Message Identifiers On Restart[i] = 'Message Identifier' in Restart Indicator]
    THEN
            Restart Message Allowed := TRUE
    ENDIF
}
```

2930.9

```
'Status' in Outcome Parameter Set := END APPLICATION
'Start' in Outcome Parameter Set := B
SET 'UI Request on Restart Present' in Outcome Parameter Set
SET 'UI Request on Outcome Present' in Outcome Parameter Set
Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
         GetTLV(TagOf(Data Record)),
         GetTLV(TagOf(Discretionary Data)),
         GetTLV(TagOf(User Interface Request Data 1)),
         GetTLV(TagOf(User Interface Request Data 2))) Signal
```

2930.21

Check if Transaction Type indicates a cash transaction (cash withdrawal or cash disbursement) or a purchase transaction (purchase or purchase with cashback):

[Transaction Type = '01' OR Transaction Type = '17' OR
Transaction Type = '00' OR Transaction Type = '09']

2930.29

Check if Card returns a valid 'Alternate Interface Preference' in Restart Indicator:

['Alternate Interface Preference' in Restart Indicator is not an RFU value of 'Alternate Interface Preference' in Outcome Parameter Set?]

2930.32 and **2930.38**

```
Restart Message Allowed := FALSE
FOR i := 1 TO GetLength(TagOf(Message Identifiers On Restart))
{
    IF      [Message Identifiers On Restart[i] = 'Message Identifier' in Restart Indicator]
    THEN
            Restart Message Allowed := TRUE
            EXIT loop
    ENDIF
}
```

2930.45

```
Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
         GetTLV(TagOf(Data Record)),
         GetTLV(TagOf(Discretionary Data)),
         GetTLV(TagOf(User Interface Request Data 1))) Signal
```

2930.47

```
CreateDiscretionaryData ()
'Status' in Outcome Parameter Set := END APPLICATION
SET 'UI Request on Outcome Present' in Outcome Parameter Set
Send OUT(GetTLV(TagOf(Outcome Parameter Set)),
         GetTLV(TagOf(Discretionary Data)),
         GetTLV(TagOf(User Interface Request Data 1))) Signal
```

6.4 Procedures

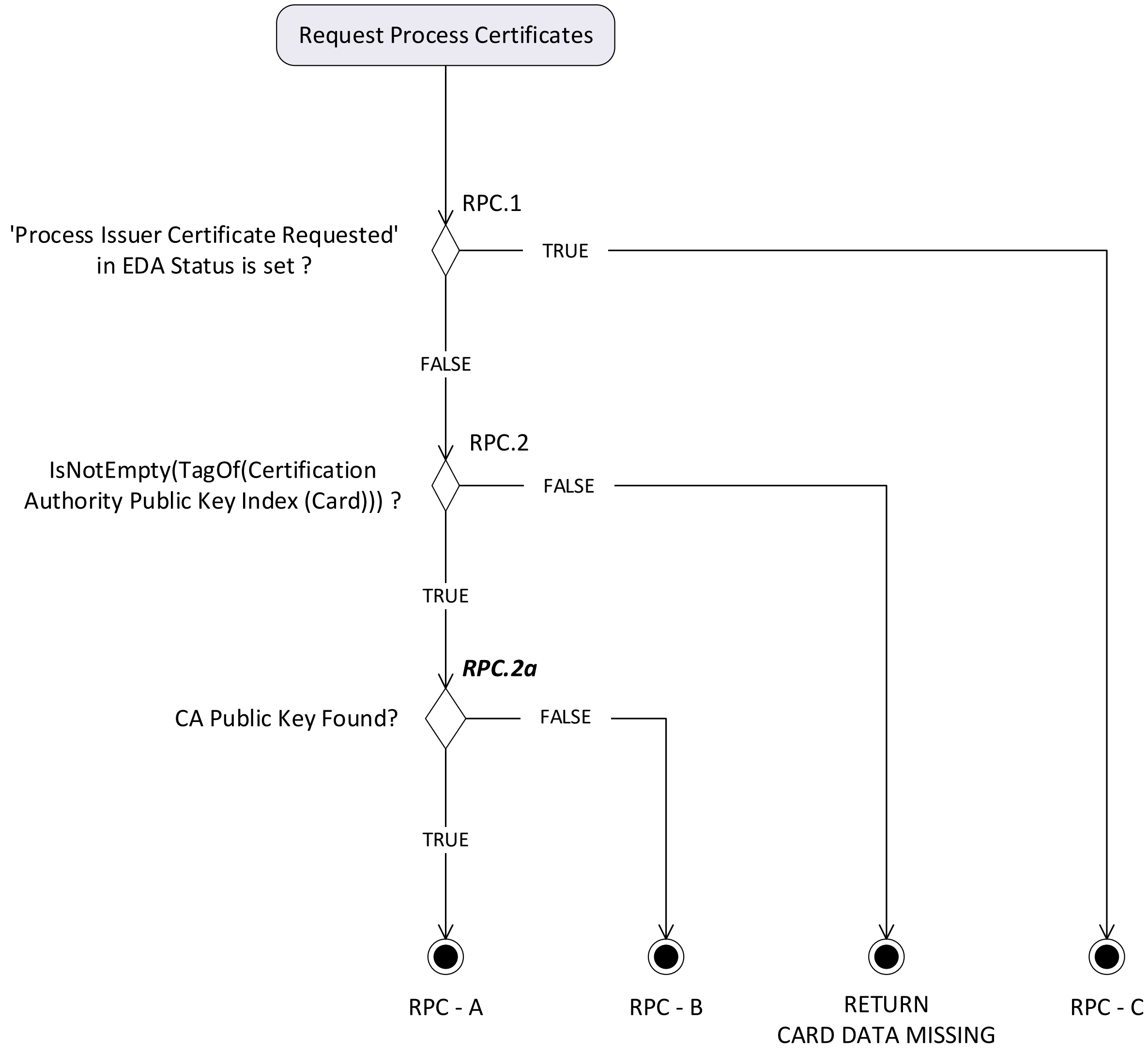
This section describes the procedures used by the state transition flow diagrams of section 6.3.

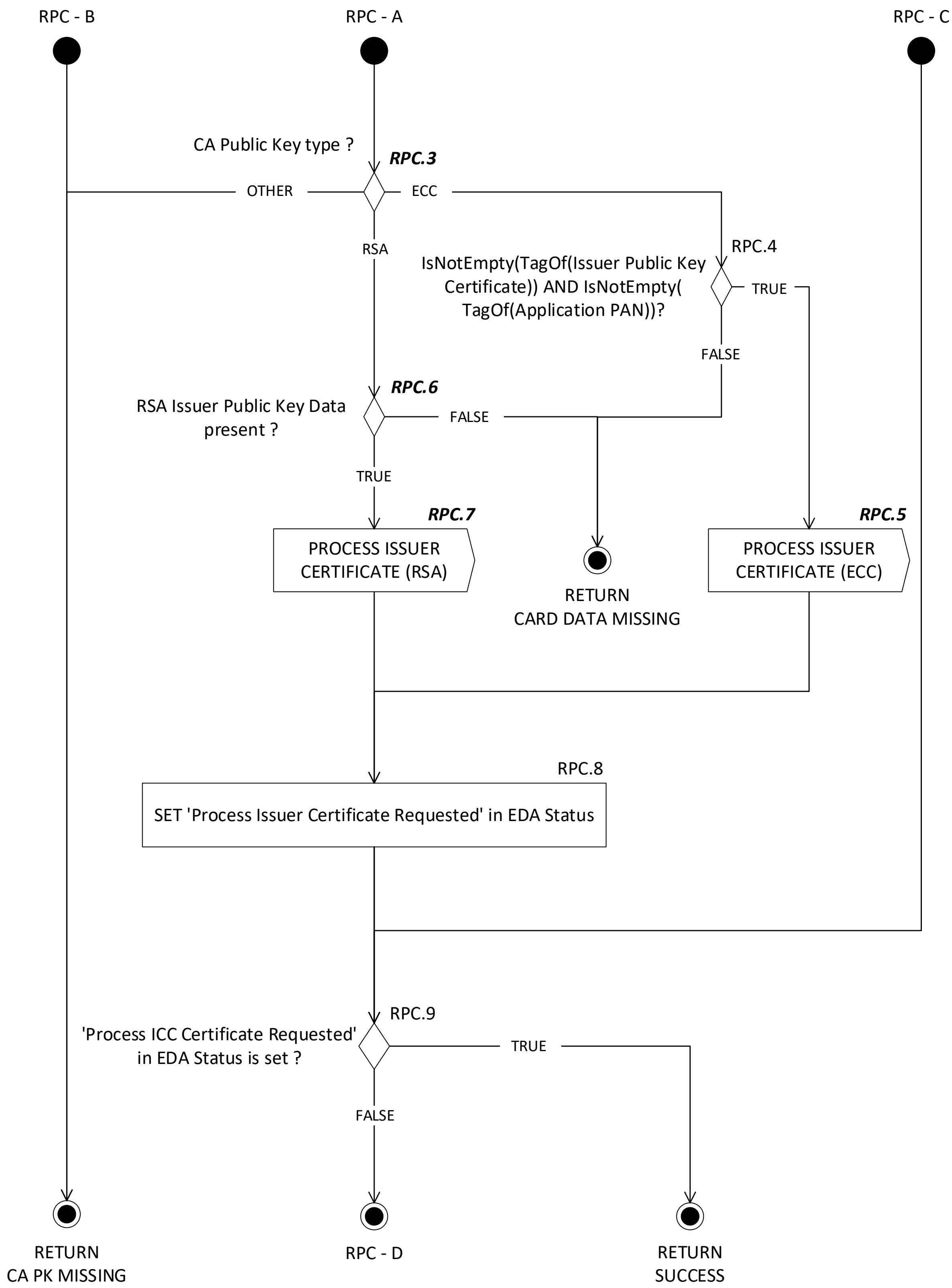
6.4.1 Request Process Certificates

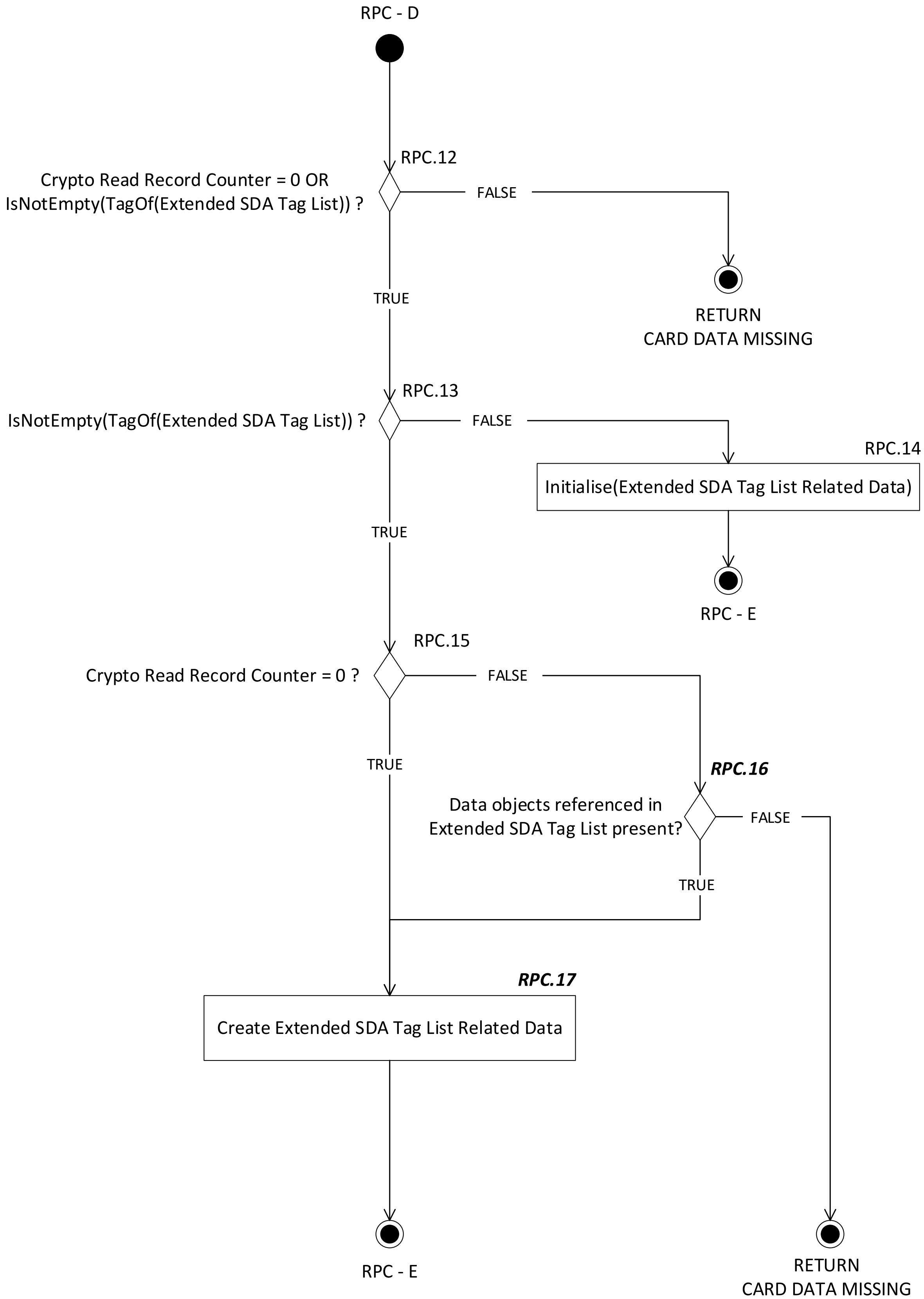
This procedure verifies if the data objects necessary to process the issuer and ICC certificates are present in the TLV Database and CA Public Key Database. If the data objects are present, then the procedure sends the PROCESS ISSUER CERTIFICATE and PROCESS ICC CERTIFICATE Signals. It returns SUCCESS if both requests are sent, it returns CARD DATA MISSING if a necessary Card data object is missing, and it returns CA PK MISSING if the Certification Authority Public Key is not present in the CA Public Key Database.

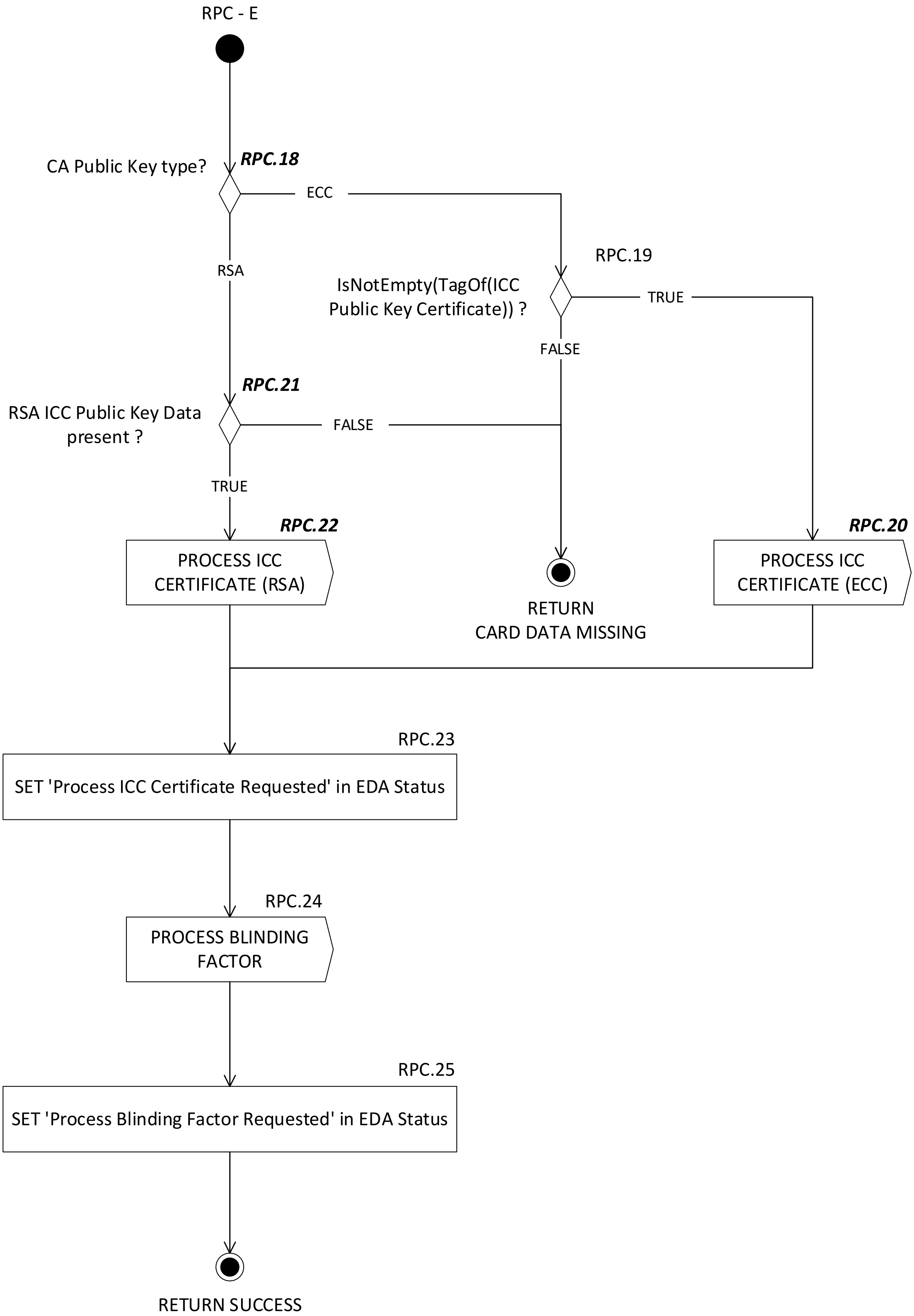
Figure 6.21 shows the flow diagram of the Request Process Certificates procedure. Symbols in this diagram are labelled RPC.X.

Figure 6.21—Request Process Certificates Flow Diagram




RPC - B
RPC - A
RPC - C
CA Public Key type ?
RPC.3
OTHER
ECC
RSA
RPC.4
IsNotEmpty(TagOf(Issuer Public Key Certificate)) AND IsNotEmpty(TagOf(Application PAN))?
TRUE
FALSE
RPC.6
RSA Issuer Public Key Data present ?
FALSE
TRUE
RPC.7
PROCESS ISSUER CERTIFICATE (RSA)
RETURN CARD DATA MISSING
RPC.5
PROCESS ISSUER CERTIFICATE (ECC)
RPC.8
SET 'Process Issuer Certificate Requested' in EDA Status
RPC.9
'Process ICC Certificate Requested' in EDA Status is set ?
TRUE
FALSE
RETURN CA PK MISSING
RPC - D
RETURN SUCCESS


RPC - D
RPC.12
Crypto Read Record Counter = 0 OR
IsNotEmpty(TagOf(Extended SDA Tag List)) ?
FALSE
TRUE
RETURN
CARD DATA MISSING
RPC.13
IsNotEmpty(TagOf(Extended SDA Tag List)) ?
FALSE
RPC.14
Initialise(Extended SDA Tag List Related Data)
TRUE
RPC - E
RPC.15
Crypto Read Record Counter = 0 ?
FALSE
TRUE
RPC.16
Data objects referenced in
Extended SDA Tag List present?
FALSE
TRUE
RPC.17
Create Extended SDA Tag List Related Data
RPC - E
RETURN
CARD DATA MISSING


RPC - E
CA Public Key type?
RPC.18
ECC
RSA
RPC.19
IsNotEmpty(TagOf(ICC Public Key Certificate)) ?
TRUE
FALSE
RPC.21
RSA ICC Public Key Data present ?
FALSE
TRUE
RPC.22
PROCESS ICC CERTIFICATE (RSA)
RETURN CARD DATA MISSING
RPC.20
PROCESS ICC CERTIFICATE (ECC)
RPC.23
SET 'Process ICC Certificate Requested' in EDA Status
RPC.24
PROCESS BLINDING FACTOR
RPC.25
SET 'Process Blinding Factor Requested' in EDA Status
RETURN SUCCESS

RPC.2a

CA Public Key found?:

[The Certification Authority Public Key referenced by Certification Authority Public Key Index (Card) and RID (DF Name[1:5]) is present in CA Public Key Database]

RPC.3

CA Public Key type?:

[Verify if the Certification Authority Public Key Algorithm (Suite) Indicator of the entry identified by Certification Authority Public Key Index (Card) and RID (DF Name[1:5]) in the CA Public Key Database defines an RSA, ECC, or other certificate algorithm suite (see [EMV Book E] section 8.4)]

RPC.5

Send PROCESS ISSUER CERTIFICATE(
    Issuer Public Key Certificate,
    Application PAN,
    Certification Authority Public Key Index (Card),
    DF Name) Signal

RPC.6

RSA Issuer Public Key Data present ?:

[IsNotEmpty(TagOf(Issuer Public Key Certificate))
AND IsNotEmpty(TagOf(Issuer RSA Public Key Exponent))
AND ((Crypto Read Record Counter = 0) OR IsNotEmpty(TagOf(Issuer RSA Public Key Remainder)))
AND IsNotEmpty(TagOf(Application PAN))]

RPC.7

```
IF      [IsNotEmpty(TagOf(Issuer RSA Public Key Remainder))]
THEN
        Send PROCESS ISSUER CERTIFICATE(
                Issuer Public Key Certificate,
                Issuer RSA Public Key Exponent,
                Issuer RSA Public Key Remainder,
                Application PAN,
                Certification Authority Public Key Index (Card),
                DF Name) Signal
ELSE
        Send PROCESS ISSUER CERTIFICATE(
                Issuer Public Key Certificate,
                Issuer RSA Public Key Exponent,
                Application PAN,
                Certification Authority Public Key Index (Card),
                DF Name) Signal
ENDIF
```

RPC.16

Data objects referenced in Extended SDA Tag List present ?:

```
Data objects referenced in Extended SDA Tag List present := TRUE
FOR every T in Extended SDA Tag List
{
        IF      [IsNotPresent(T) OR IsEmpty(T)]
        THEN
                Data objects referenced in Extended SDA Tag List present := FALSE
                EXIT loop
        ENDIF
}
```

RPC.17

Initialise Extended SDA Tag List Related Data with an empty string

```
FOR every T in Extended SDA Tag List
{
        IF      [IsNotEmpty(T)]
        THEN
                Extended SDA Tag List Related Data := Extended SDA Tag List Related Data
                || GetTLV(T)
        ELSE
                Extended SDA Tag List Related Data := Extended SDA Tag List Related Data
                || T || '00'
        ENDIF
}
```

RPC.18

CA Public Key type?:

[Verify if the Certification Authority Public Key Algorithm (Suite) Indicator of the entry identified by Certification Authority Public Key Index (Card) and RID (DF Name[1:5]) in the CA Public Key Database defines an RSA or ECC certificate algorithm suite (see [EMV Book E] section 8.4)]

RPC.20

```
Send PROCESS ICC CERTIFICATE(
        ICC Public Key Certificate,
        Application Interchange Profile,
        Extended SDA Tag List Related Data) Signal
```

RPC.21

RSA ICC Public Key Data present ?:

[IsNotEmpty(TagOf(ICC Public Key Certificate))
AND IsNotEmpty(TagOf(ICC RSA Public Key Exponent))
AND ((Crypto Read Record Counter = 0) OR IsNotEmpty(TagOf(ICC RSA Public Key Remainder)))
AND IsNotEmpty(TagOf(Application PAN))]

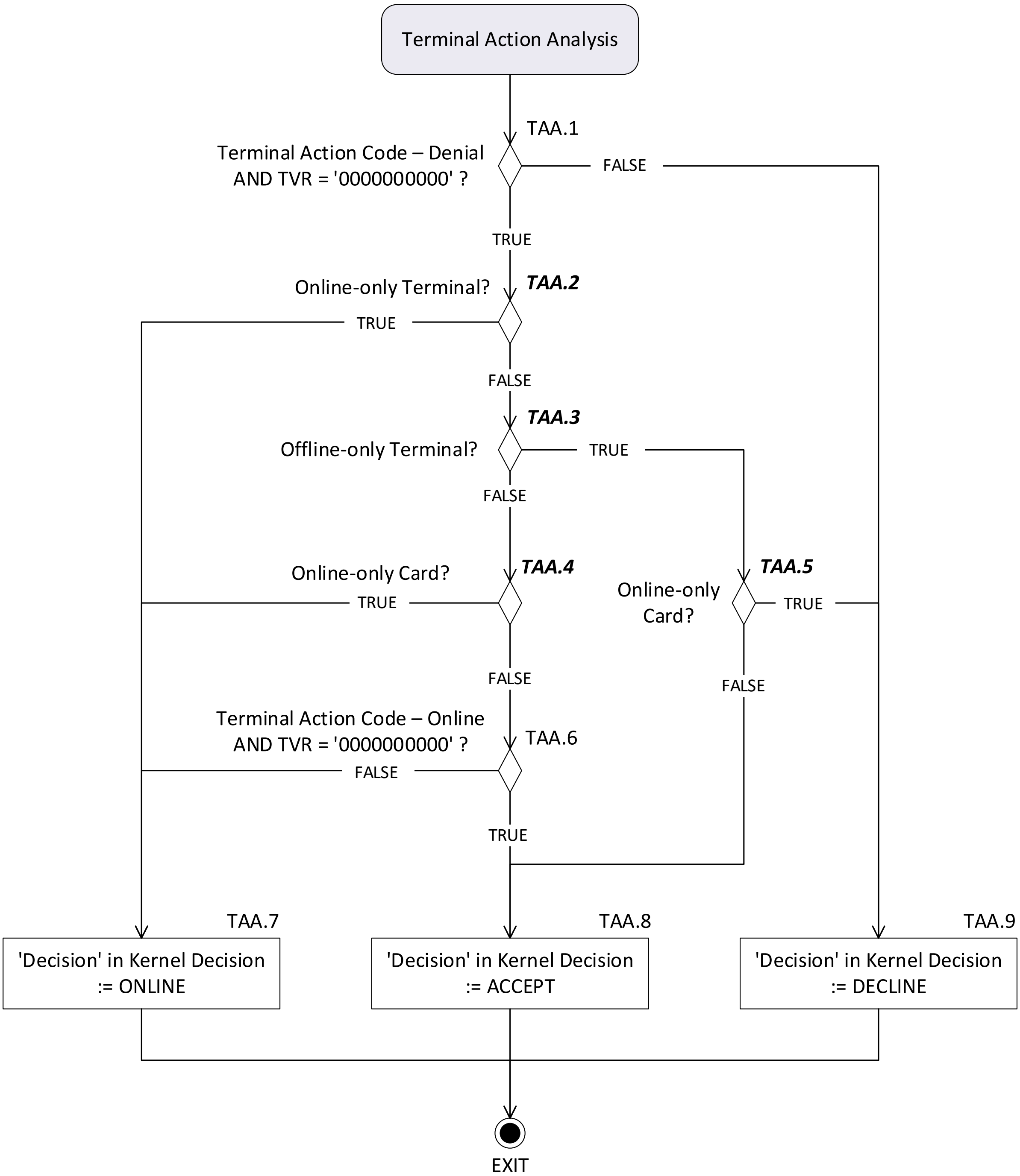
RPC.22

IF [IsNotEmpty(TagOf(ICC RSA Public Key Remainder))]
THEN
    Send PROCESS ICC CERTIFICATE(
        ICC Public Key Certificate,
        ICC RSA Public Key Exponent,
        ICC RSA Public Key Remainder,
        Application PAN,
        Application Interchange Profile,
        Extended SDA Tag List Related Data) Signal
ELSE
    Send PROCESS ICC CERTIFICATE(
        ICC Public Key Certificate,
        ICC RSA Public Key Exponent,
        Application PAN,
        Application Interchange Profile,
        Extended SDA Tag List Related Data) Signal
ENDIF

6.4.2 Terminal Action Analysis

Figure 6.22 shows the flow diagram of the Terminal Action Analysis procedure. Symbols in this diagram are labelled TAA.X.

Figure 6.22—Terminal Action Analysis Flow Diagram



TAA.2

Online-only Terminal?:

[(Terminal Type = '11') OR (Terminal Type = '21') OR (Terminal Type = '14') OR (Terminal Type = '24') OR (Terminal Type = '34')]

TAA.3

Offline-only Terminal?:

[(Terminal Type = '23') OR (Terminal Type = '26') OR (Terminal Type = '36') OR (Terminal Type = '13') OR (Terminal Type = '16')]

TAA.4

Online-only Card?:

['Offline capable' in Application Interchange Profile is not set]

TAA.5

Online-only Card?:

['Offline capable' in Application Interchange Profile is not set]

7 Process C — Cryptographic Processing

This chapter describes the cryptographic Process C.

Process C is specified in a consistent manner to Process K but it has just one state: `Idle`. When it receives a Signal, it processes it. State is maintained by a number of Boolean flags. Process C receives Signals only from Process K and it sends Signals only to Process K. Signals received by Process K from Process C are handled on a separate Queue to other messages received by Process K. When Process K terminates, Process C also terminates and any Signals on the Queue are discarded.

Process C does not have access to the TLV Database of Process K. Process C maintains its own datastore. Data is transferred between Process C and Process K through the Signals and this specification assumes that this is done by value. Process C has however direct access to the CA Public Key Database and Certification Revocation List.

7.1 Internal Data Store

The data objects listed in Table 7.1 are used by Process C during the processing of the Signals as described in this chapter. These data objects are not retained in memory when Process C terminates.

Table 7.1—Process C Internal Data Store

Data Object	Description
Blinded Public Key	A copy of the blinded public key recovered from the Card Key Data received with the GPO PRIVACY Signal in (*x*, *y*) format. The internal representation of Elliptic Curve points is implementation specific.
Card Message Counter	2-byte counter used in processing VALIDATE MESSAGE Signals and in decrypting records. It is initialised to '8000' and incremented after the GPO PRIVACY Signal, each RECORD RECEIVED Signal with an encrypted record, and each VALIDATE MESSAGE Signal. It forms the first 2 bytes of the counter block (SV) used for the CTR mode decryption. It forms the first 2 bytes of the MAC calculation for READ DATA and WRITE DATA processing. Note: For Enhanced Data Authentication MAC validation, a fixed 2-byte value of '0000' is used instead so that there is no chance of collision.
C ASI List	List containing the certificate ASIs supported by Process C. See [EMV Book E] section 8.4.
ICC ECC Public Key	The ICC ECC public key recovered as part of processing the ICC certificate in (*x*, *y*) format. The internal representation of Elliptic Curve points is implementation specific.
ICC ECC Public Key Recovered	Boolean indicating whether an ICC ECC public key has been recovered.
Issuer RSA Public Key	If RSA certificates are being used, the issuer RSA public key recovered from the issuer certificate.

Data Object	Description
Issuer ECC Public Key	If ECC certificates are being used, the issuer ECC public key recovered from the issuer certificate in (*x*, *y*) format. The internal representation of Elliptic Curve points is implementation specific.
Kernel Message Counter	2-byte counter used in processing ENCRYPT MESSAGE Signals. It is initialised to '0000' and incremented after each subsequent ENCRYPT MESSAGE Signal. It forms the first 2 bytes of the counter block (SV) used for the CTR mode encryption.
Kernel ECC Private Key	The ECC private key used for Elliptic Curve Diffie-Hellman establishment of the Session Keys for Privacy and Integrity protection.
Kernel ECC Public Key	The ECC public key used for Elliptic Curve Diffie-Hellman establishment of the Session Keys for Privacy and Integrity protection. It consists of the N_{FIELD} byte *x*-coordinate followed by the N_{FIELD} byte *y*-coordinate.
Key Type	Enumeration data object indicating what type of issuer key is being used; set to one of “NONE”, “ECC”, or “RSA”.
MAC	Temporary storage for MAC to be validated.
Recovered Blinding Factor	The plaintext version of the blinding factor recovered from the encrypted blinding factor included in Card Key Data.
Result OK	Temporary storage for Boolean flag indicating if processing was successful or not.
RSA Certificates Enabled	Boolean indicating whether RSA certificates are enabled.
SC ASI	The secure channel ASI that will be used for this transaction. See [EMV Book E] section 8.4.
SC ASI List	List containing the secure channel ASIs supported by Process C. For this version of the specification, SC ASI List = '00'. See [EMV Book E] section 8.4.
SDA Hash	Temporary storage for the hash over static data.

Data Object	Description
Session Key for Confidentiality	AES session key derived during processing of the GPO PRIVACY Signal and used for encrypting and decrypting messages.
Session Key for Integrity	AES session key derived during processing of the GPO PRIVACY Signal and used for validating AES-CMACs.
Session Keys Recovered	Boolean indicating whether the Session Keys for Confidentiality and Integrity have (TRUE) or have not (FALSE) been recovered as a result of processing the GPO PRIVACY Signal.
Signed Records	A string of signed records
Static Data To Be Authenticated	A string of the static data that needs to be hashed to obtain SDA Hash.
TLV	Temporary storage to store TLV-coded byte array.
Temp Data	Temporary storage when processing data such as a record or Issuer Application Data MAC.

7.2 Flow Diagrams

This section describes the flow diagrams that process the Signals sent by Process K to Process C. Symbols in the flow diagrams are labelled C.X.

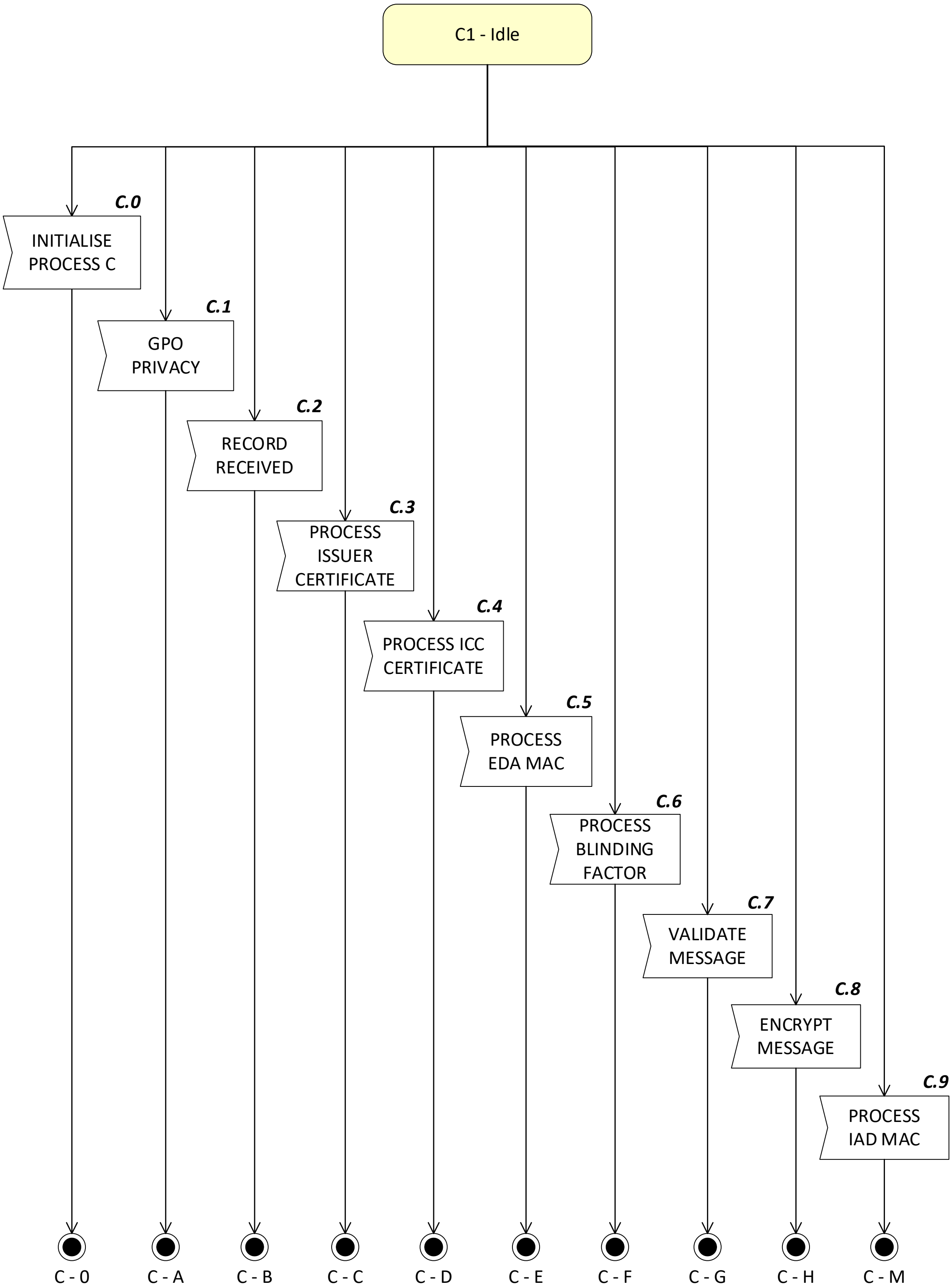
The Signals that Process K sends to Process C are:

- INITIALISE PROCESS C
- GPO PRIVACY
- RECORD RECEIVED
- PROCESS ISSUER CERTIFICATE
- PROCESS ICC CERTIFICATE
- PROCESS EDA MAC
- PROCESS BLINDING FACTOR
- VALIDATE MESSAGE
- ENCRYPT MESSAGE
- PROCESS IAD MAC

7.2.1 Process Incoming Signals

Figure 7.1 shows the flow diagram of the processing of the incoming Signals.

Figure 7.1—Process Incoming Signals



C.0

Receive INITIALISE PROCESS C Signal with the following:

- P_Card_SC_ASI_List (list of secure channel ASIs supported by the Card)
- P_RSA_Certificates_Enabled_Flag (Boolean, TRUE if RSA certificates are enabled)

C.1

Receive GPO PRIVACY Signal with the following:

- P_Card_Key_Data

C.2

Receive RECORD RECEIVED Signal with the following:

- P_Record (either '70' or 'DA' template)
- P_Signed_Record_Flag (Boolean, TRUE if record is signed)

C.3

Receive PROCESS ISSUER CERTIFICATE Signal with the following:

- P_Issuer_Public_Key_Certificate (containing an RSA or ECC certificate)
- P_Issuer_RSA_Public_Key_Exponent (for RSA only)
- P_Issuer_RSA_Public_Key_Remainder (for RSA only and only if present)
- P_Application_PAN (from which the Issuer Identifier will be recovered for checking)
- P_CA_Public_Key_Index_Card
- P_DF_Name (from which the RID is recovered for checking)

C.4

Receive PROCESS ICC CERTIFICATE Signal with the following:

- P_ICC_Public_Key_Certificate (containing an RSA or ECC certificate)
- P_ICC_RSA_Public_Key_Exponent (for RSA only)
- P_ICC_RSA_Public_Key_Remainder (for RSA only and only if present)
- P_Application_PAN (for RSA only)
- P_AIP
- P_Extended_SDA_Tag_List_Related_Data (this may be an empty string if no data)

C.5

Receive PROCESS EDA MAC Signal with the following:

- P_Enhanced_Data_Authentication_MAC
- P_Enhanced_Data_Message_To_Be_Validated

C.6

Receive PROCESS BLINDING FACTOR Signal

C.7

Receive VALIDATE MESSAGE Signal with the following:

- P_Message_Type (WRITE or READ)
- P_Message_To_Be_Validated (including the MAC at the end of the message)

C.8

Receive ENCRYPT MESSAGE Signal with the following:

- P_Message_To_Be_Encrypted

C.9

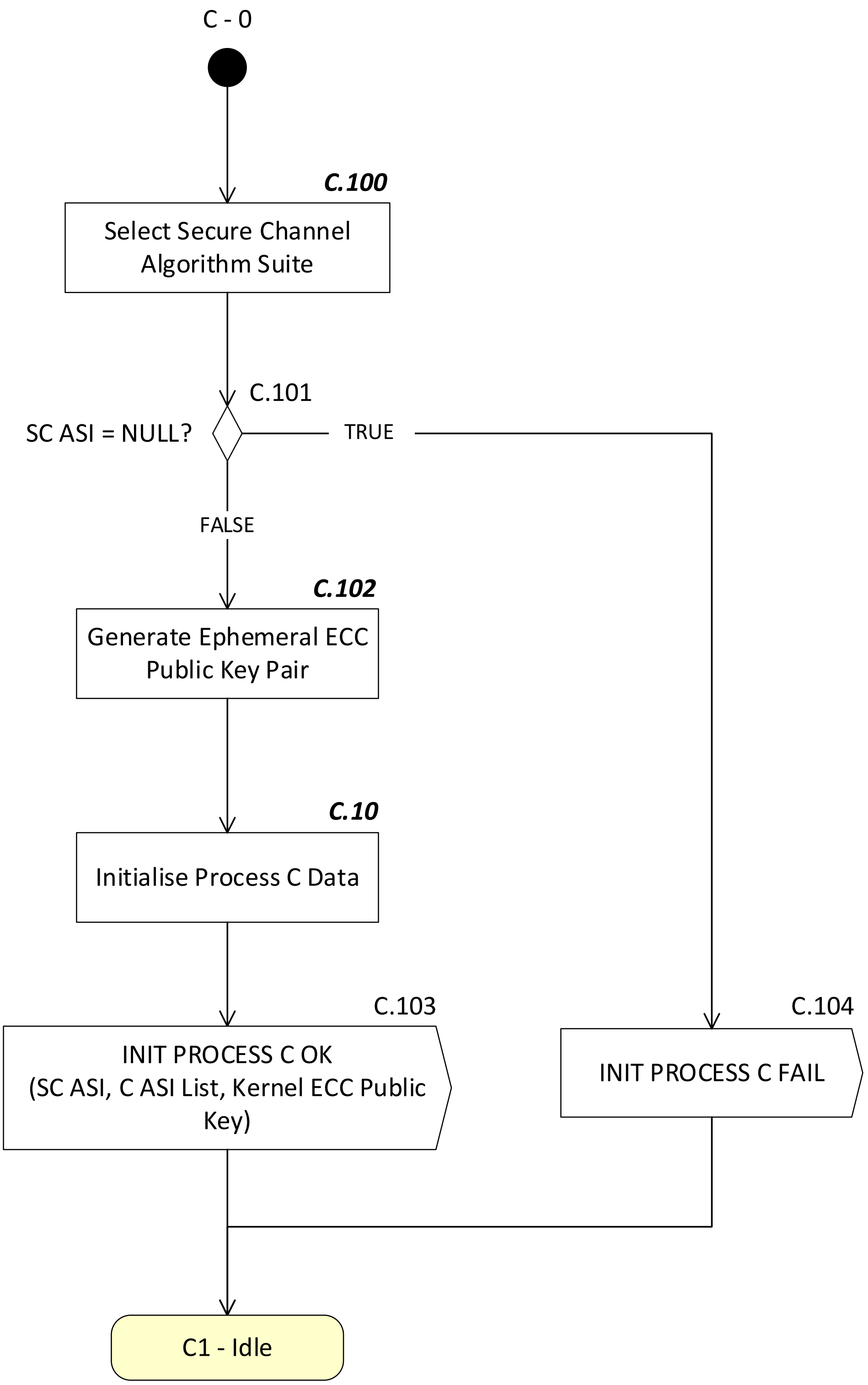
Receive PROCESS IAD MAC Signal with the following:

- P_Message_To_Be_Processed
- P_AIP
- P_Extended_SDA_Tag_List_Related_Data (this may be an empty string if no data)

7.2.2 Initialise Process C

Figure 7.2 shows the flow diagram of the processing of the INITIALISE PROCESS C Signal.

Figure 7.2—Initialise Process C



C.100

```
SC ASI := NULL
FOR every sc_asi in P_Card_SC_ASI_List
{
        IF   [SC ASI List includes sc_asi]
        THEN
                SC ASI := sc_asi
                EXIT loop
        ENDIF
}
```

C.102

Generate Kernel ECC Private Key and Kernel ECC Public Key for the selected SC ASI as specified in [EMV Book E] section 7.1).

Note:

> Whilst the generation of the key pair can be found here, it is not a requirement that it is generated at this point. The implementation is free to choose a more optimised way to generate the key pair. The requirement is merely that by the time this point is passed, a key pair exists ready to be used and that a fresh key pair is used for each transaction.

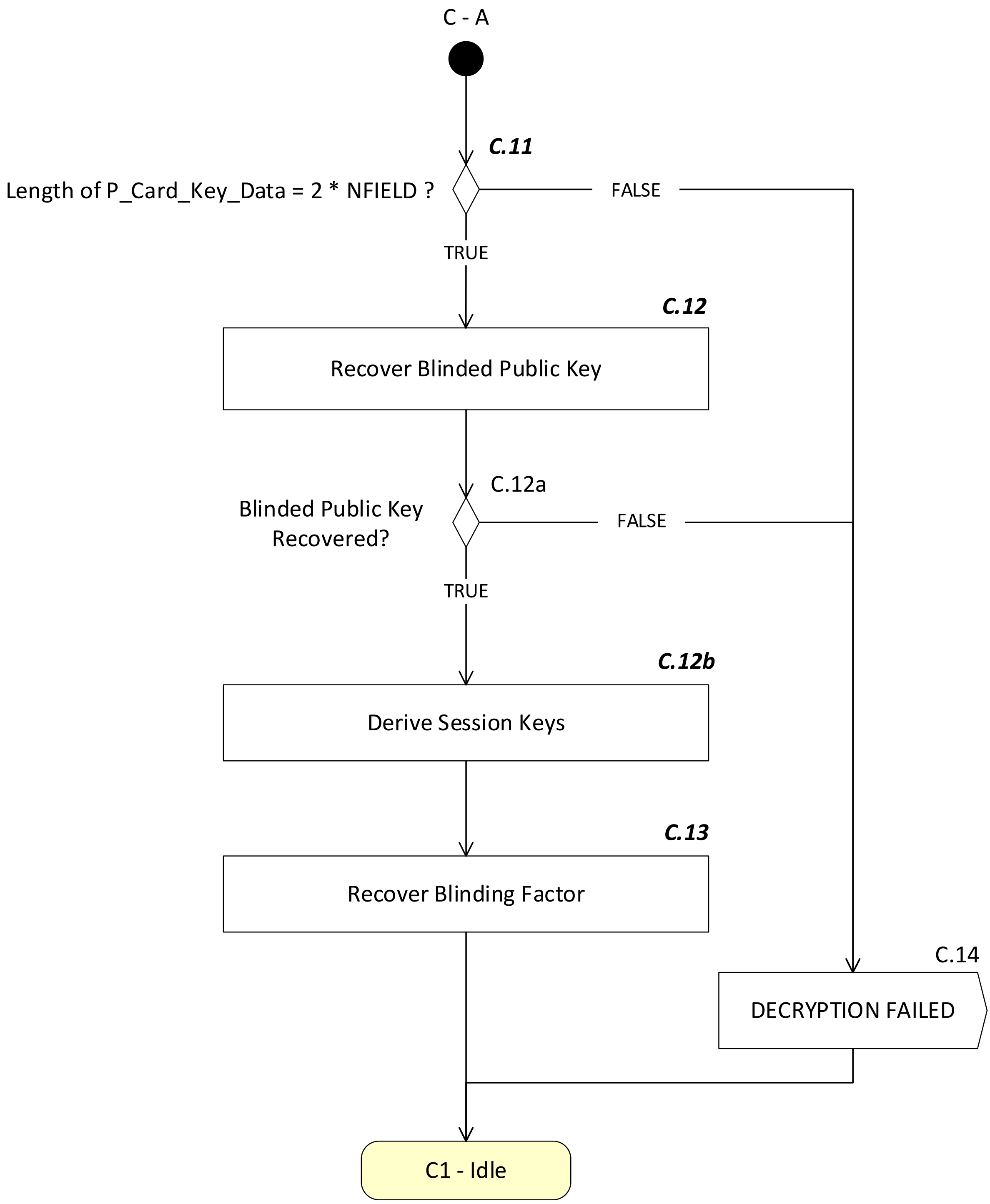
C.10

```
Session Keys Recovered := FALSE
Kernel Message Counter := '0000'
Card Message Counter := '8000'
ICC ECC Public Key Recovered := FALSE
Key Type := NONE
Static Data To Be Authenticated := empty string
Signed Records := empty string
RSA Certificates Enabled := P_RSA_Certificates_Enabled_Flag
IF      [RSA Certificates Enabled]
THEN
        C ASI List := '0110FF'
ELSE
        C ASI List := '10FFFF'
ENDIF
```

7.2.3 GPO Privacy

Figure 7.3 shows the flow diagram of the processing of the GPO PRIVACY Signal.

Figure 7.3—GPO Privacy



C.11

Note: N_{FIELD} is the size of the field element for the Elliptic Curve in question (e.g. for P-256, this is 32 bytes) and is the size of the *x*-coordinate. It is also the size of the blinding factor, hence the concatenation of the public key point and encrypted blinding factor is 2 * N_{FIELD}.

C.12

Recover the Blinded Public Key from P_Card_Key_Data[1:N_{FIELD}] according to [EMV Book E] section 8.8.4.

C.12b

Derive Session Key for Confidentiality and Session Key for Integrity using Kernel ECC Private Key and Blinded Public Key according to [EMV Book E] section 7.2.

Session Keys Recovered := TRUE

C.13

Recovered Blinding Factor := AES-CTR(Session Key for Confidentiality)[Card Message Counter, P_Card_Key_Data[N_{FIELD}+1:2*N_{FIELD}]]

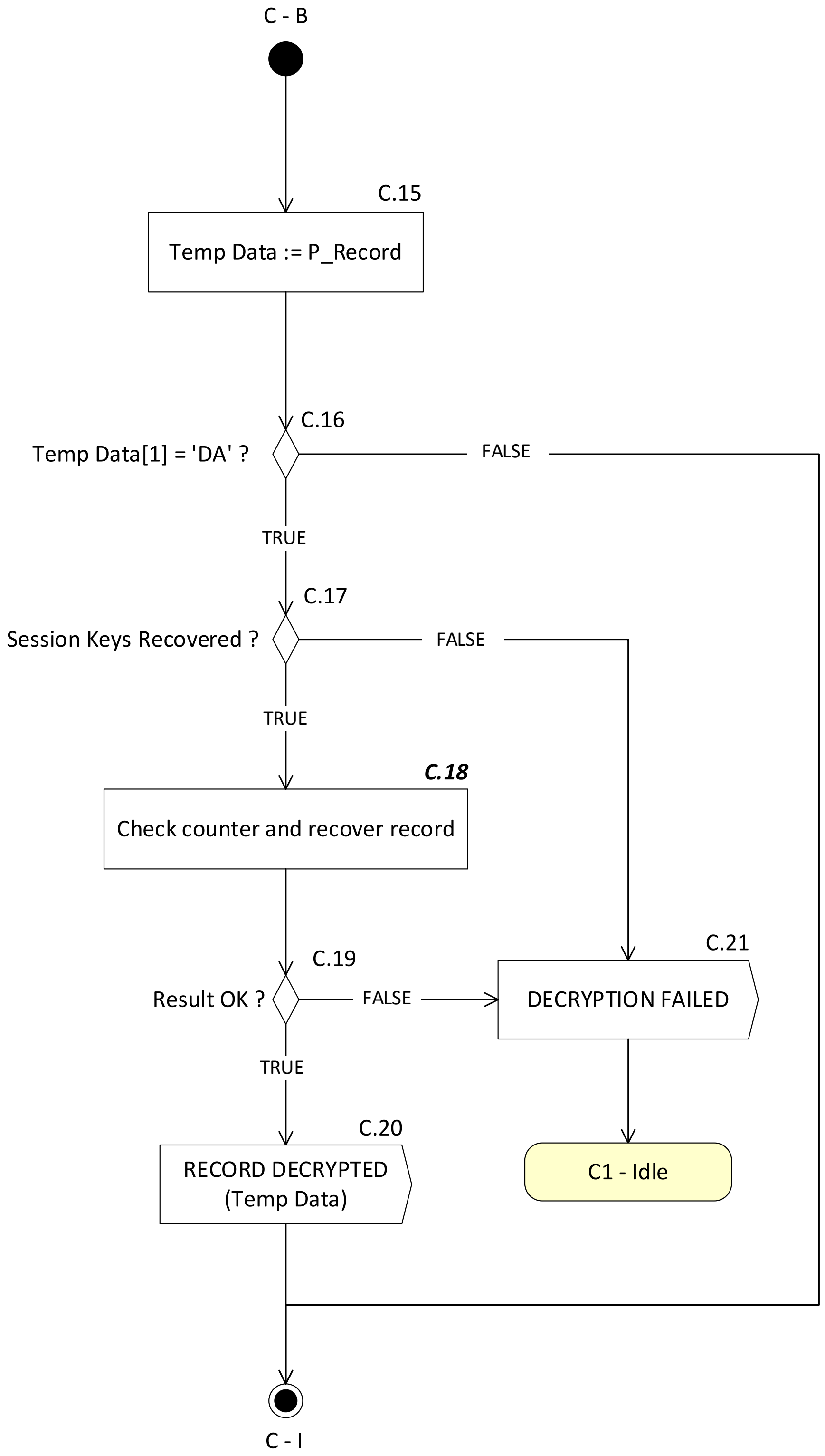
Card Message Counter := Card Message Counter + 1

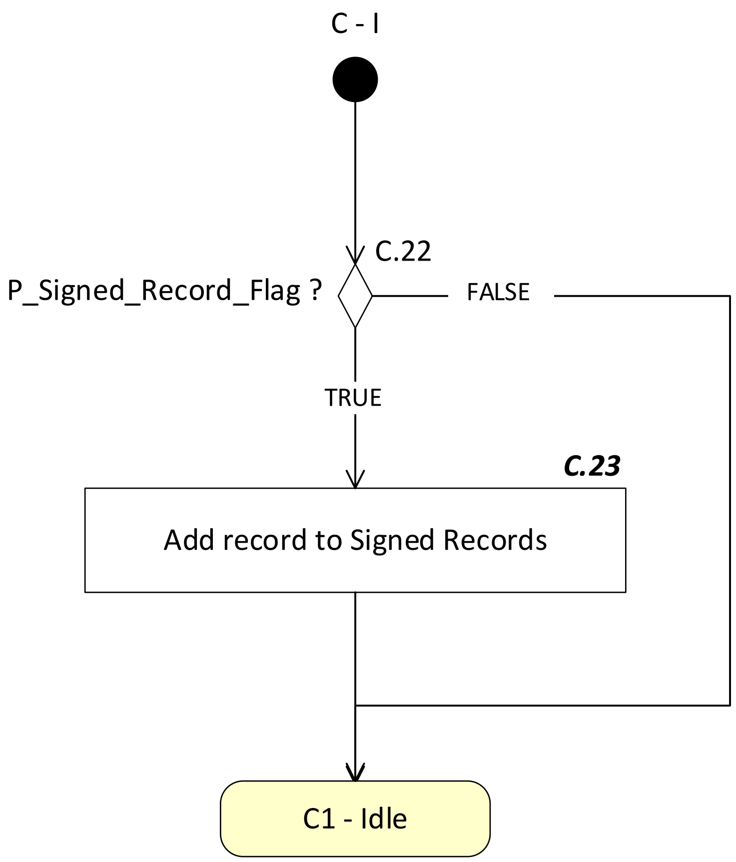
Refer to [EMV Book E] section 8.6.2 for the definition of AES-CTR.

7.2.4 Record Received

Figure 7.4 shows the flow diagram of the processing of the RECORD RECEIVED Signal.

Figure 7.4—Record Received





C.18

IF [Card Message Counter < 'FFFF']

THEN

Value field of Temp Data := AES-CTR(Session Key for Confidentiality)[Card Message Counter, Value field of Temp Data]

Card Message Counter := Card Message Counter + 1

Temp Data[1] := '70'

Result OK := TRUE

ELSE

Result OK := FALSE

ENDIF

Refer to [EMV Book E] section 8.6.2 for the definition of AES-CTR.

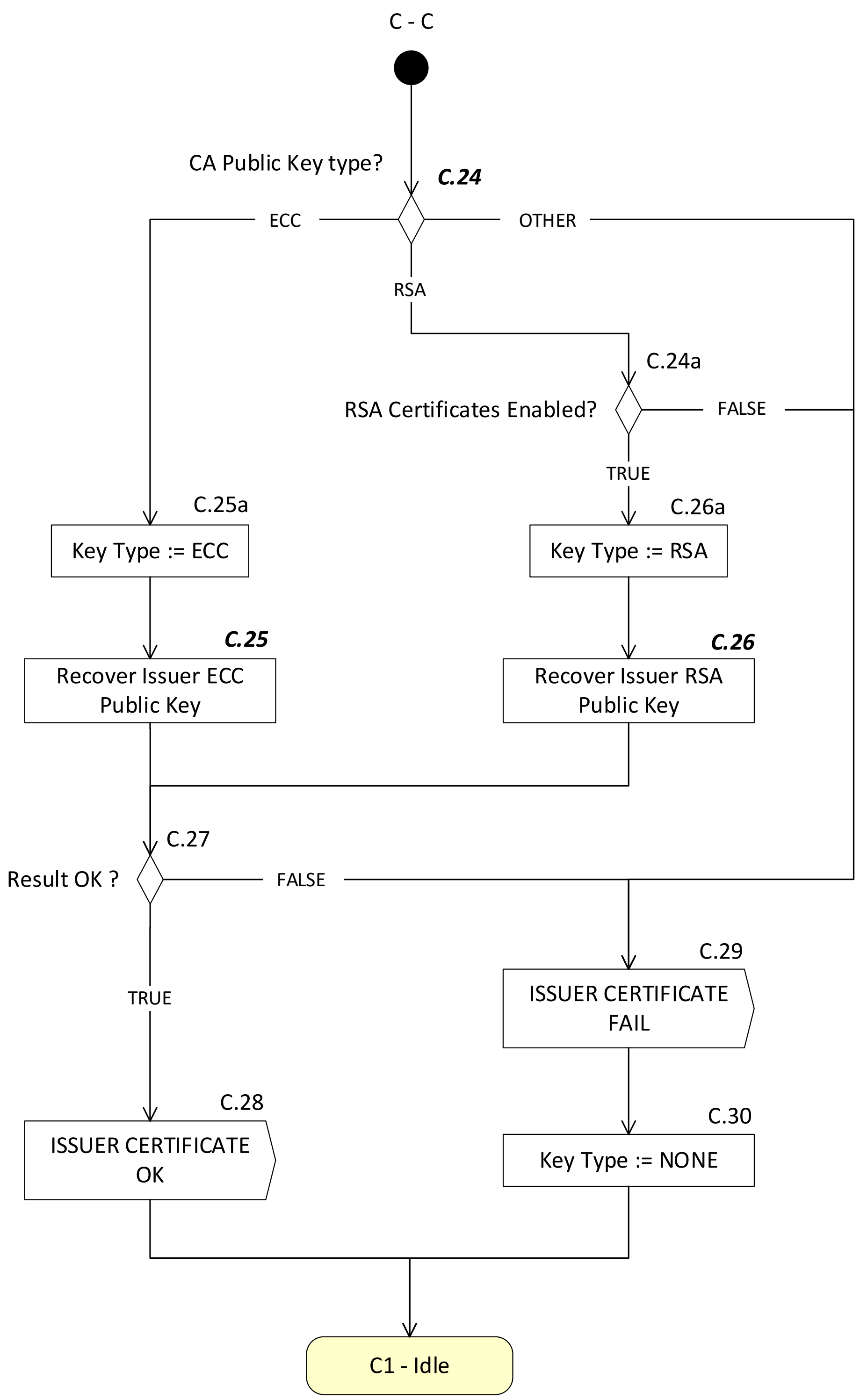
C.23

Append the value field of Temp Data (i.e. excluding tag '70' and length) at the end of Signed Records

7.2.5 Process Issuer Certificate

Figure 7.5 shows the flow diagram of the processing of the PROCESS ISSUER CERTIFICATE Signal.

Figure 7.5—Process Issuer Certificate



C.24

CA Public Key type?:

[Verify if the Certification Authority Public Key Algorithm (Suite) Indicator of the entry identified by P_CA_Public_Key_Index_Card and RID (P_DF_Name[1:5]) in the CA Public Key Database defines an RSA, ECC, or other certificate algorithm suite (see [EMV Book E] section 8.4)]

C.25

Recover Issuer ECC Public Key from P_Issuer_Public_Key_Certificate using the Certification Authority Public Key located in C.24, the Issuer Identifier from the P_Application_PAN, and the RID from the P_DF_Name according to [EMV Book E] section 5.2.

Result OK := TRUE if recovery successful, FALSE if not

C.26

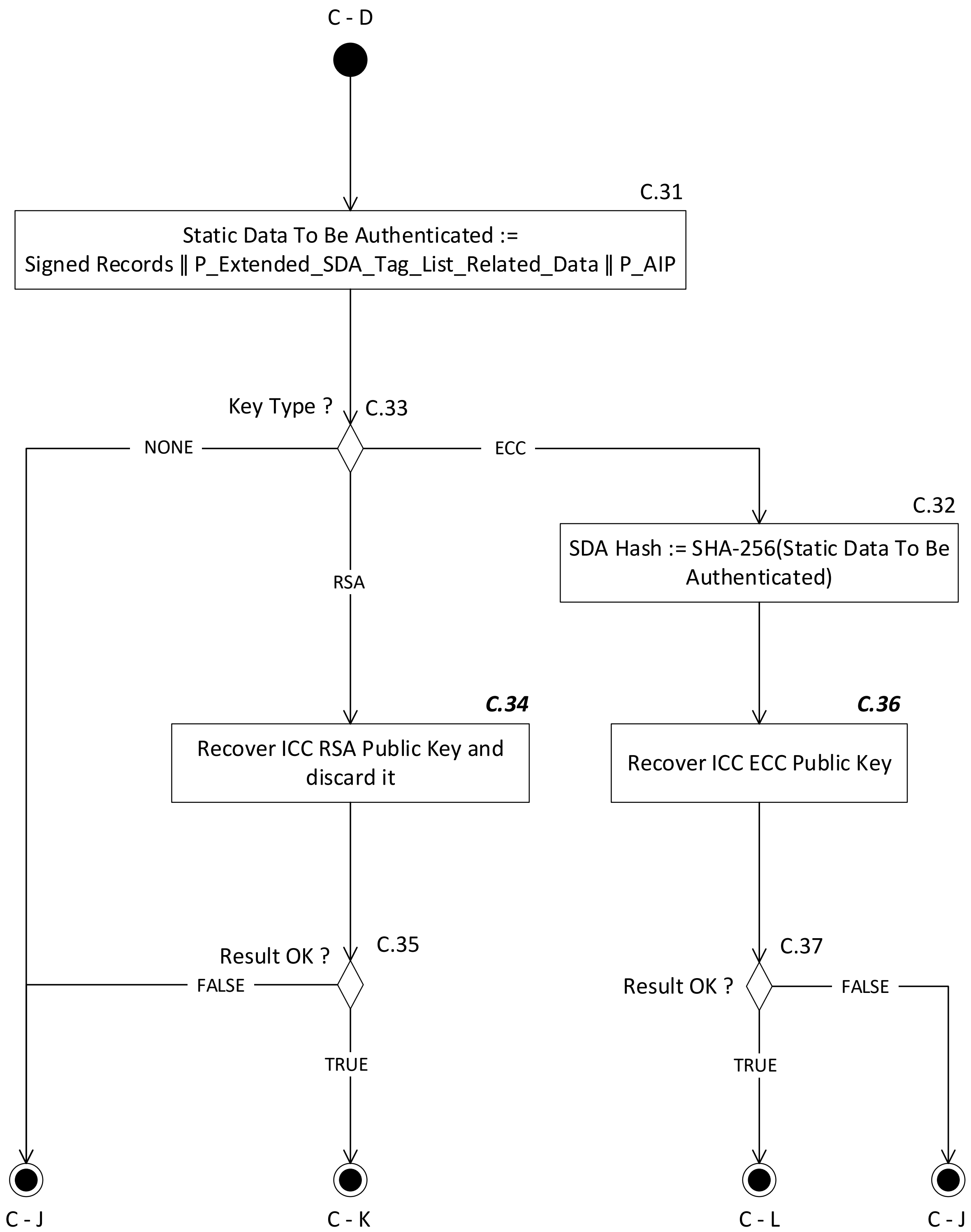
Recover Issuer RSA Public Key from P_Issuer_Public_Key_Certificate using the Certification Authority Public Key located in C.24, the Issuer Identifier from the P_Application_PAN, P_Issuer_RSA_Public_Key_Exponent and, if present, P_Issuer_RSA_Public_Key_Remainder according to [EMV Book E] section 6.2.

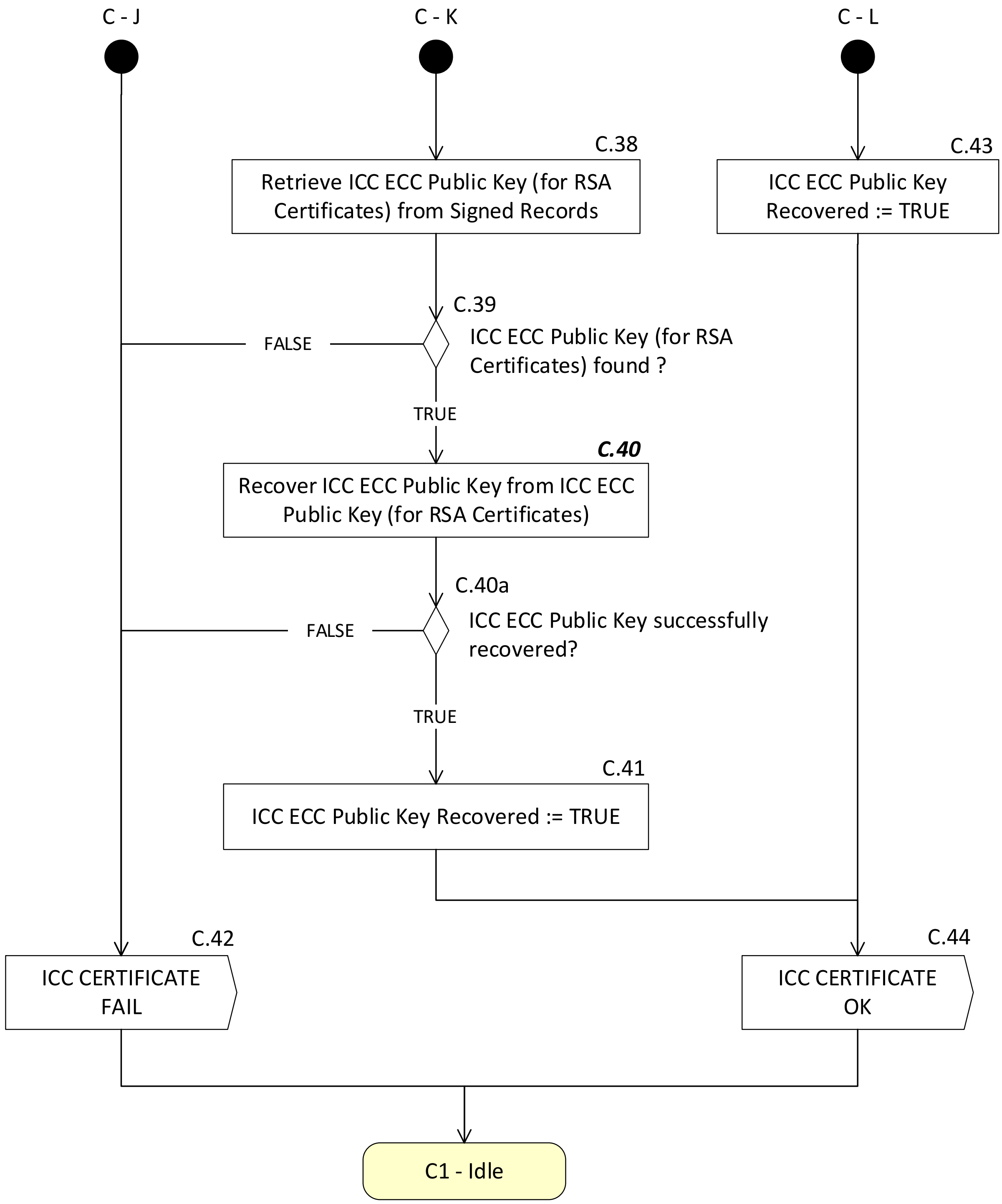
Result OK := TRUE if recovery successful, FALSE if not

7.2.6 Process ICC Certificate

Figure 7.6 shows the flow diagram of the processing of the PROCESS ICC CERTIFICATE Signal.

Figure 7.6—Process ICC Certificate




C - J
C - K
C - L
C.38
Retrieve ICC ECC Public Key (for RSA Certificates) from Signed Records
C.43
ICC ECC Public Key Recovered := TRUE
C.39
ICC ECC Public Key (for RSA Certificates) found ?
FALSE
TRUE
C.40
Recover ICC ECC Public Key from ICC ECC Public Key (for RSA Certificates)
C.40a
ICC ECC Public Key successfully recovered?
FALSE
TRUE
C.41
ICC ECC Public Key Recovered := TRUE
C.42
ICC CERTIFICATE FAIL
C.44
ICC CERTIFICATE OK
C1 - Idle

C.34

Recover the ICC RSA Public Key from the P_ICC_Public_Key_Certificate using P_ICC_RSA_Public_Key_Remainder, P_ICC_RSA_Public_Key_Exponent, P_Application_PAN, and Static Data To Be Authenticated according to [EMV Book E] section 6.4.

Result OK := TRUE if recovery successful, FALSE if not

C.36

Recover and store the ICC ECC Public Key from the P_ICC_Public_Key_Certificate using SDA Hash according to [EMV Book E] section 5.4.

Result OK := TRUE if recovery successful, FALSE if not

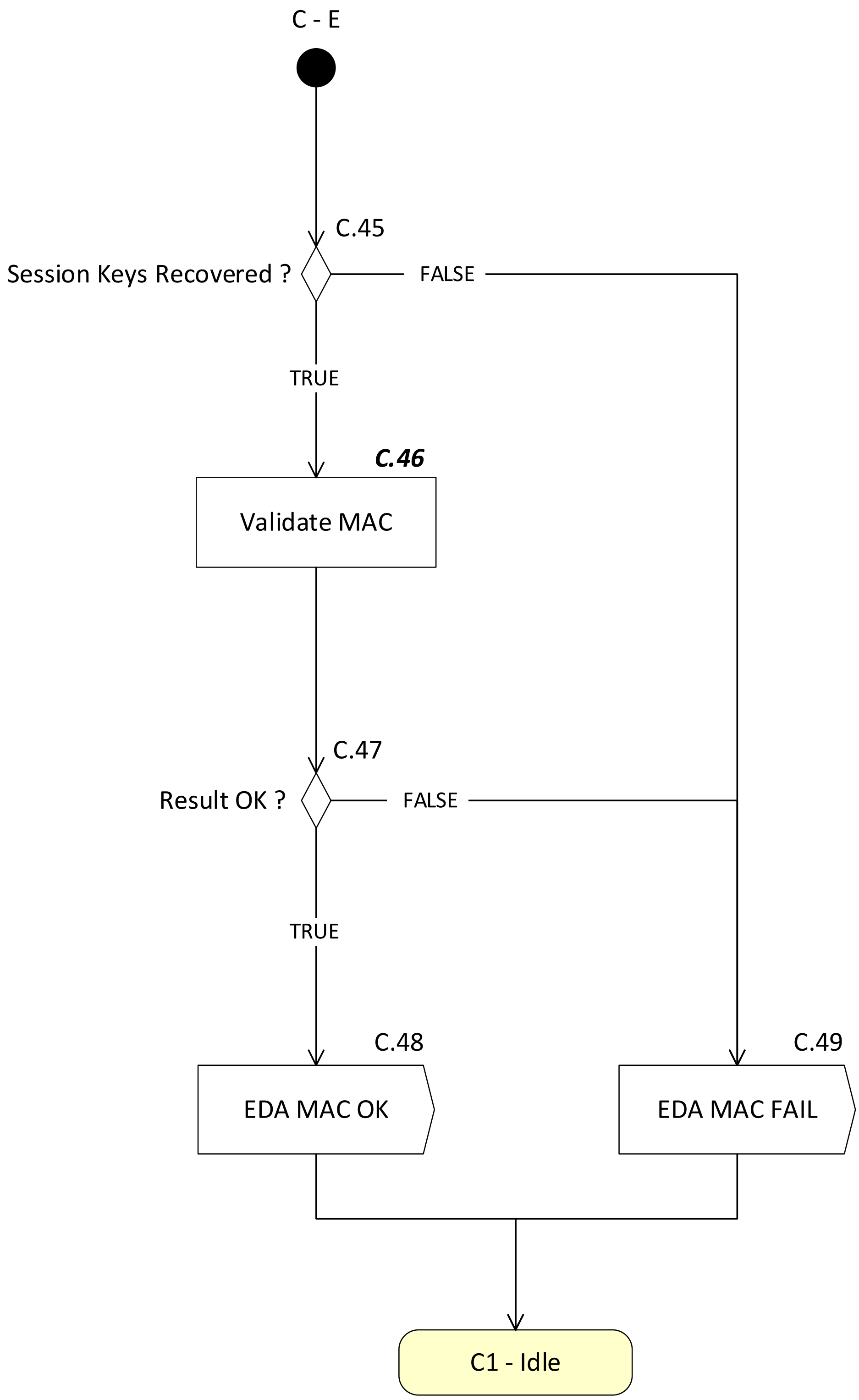
C.40

Convert ICC ECC Public Key (for RSA Certificates) into (*x, y*) format according to [EMV Book E] section 8.8.4 and store in ICC ECC Public Key.

7.2.7 Process EDA MAC

Figure 7.7 shows the flow diagram of the processing of the PROCESS EDA MAC Signal.

Figure 7.7—Process EDA MAC



C.46

Temp Data := leftmost 8-bytes of AES-CMAC(Session Key for Integrity) ['0000' || P_Enhanced_Data_Message_To_Be_Validated]

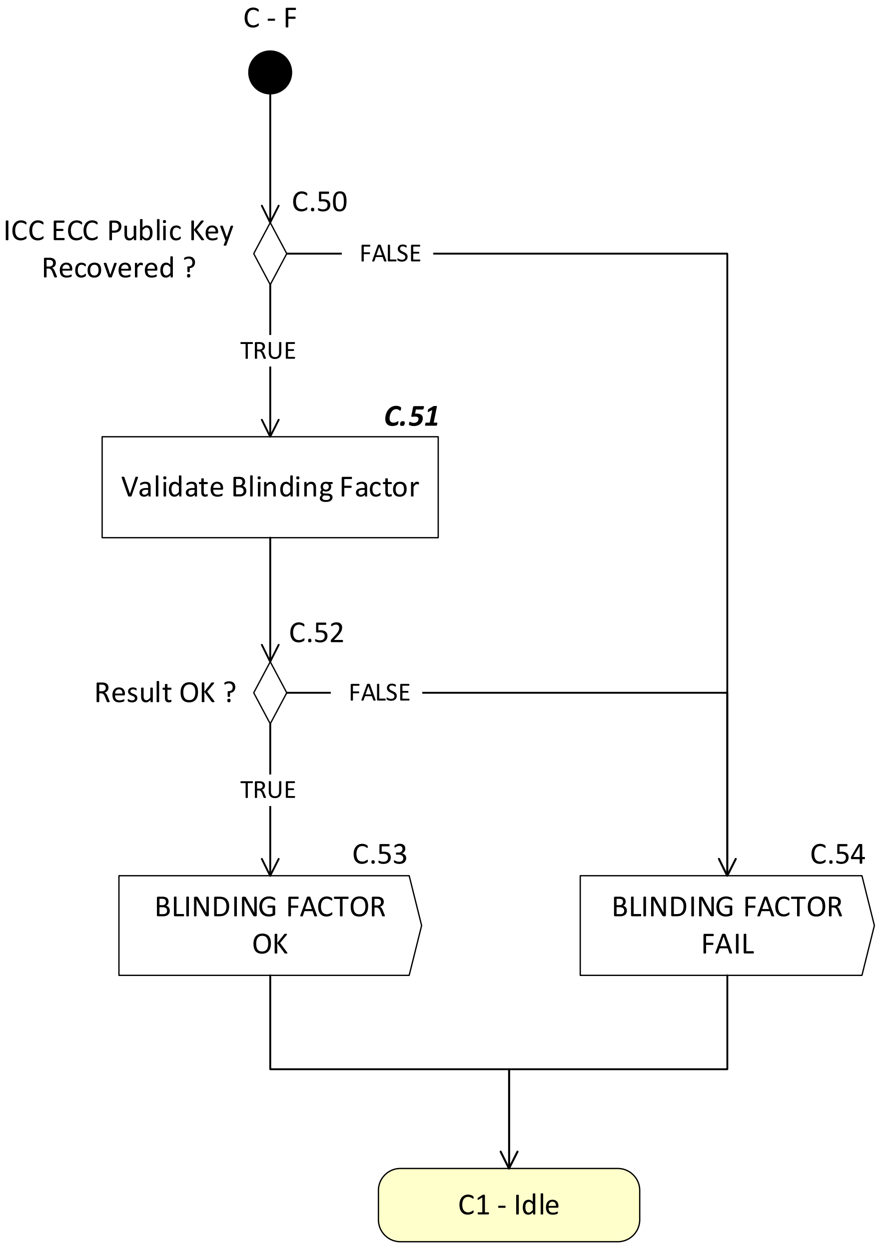
IF [Temp Data = P_Enhanced_Data_Authentication_MAC]
THEN Result OK := TRUE
ELSE Result OK := FALSE

Refer to [EMV Book E] section 8.6.4 for the definition of AES-CMAC.

7.2.8 Process Blinding Factor

Figure 7.8 shows the flow diagram of the processing of the PROCESS BLINDING FACTOR Signal.

Figure 7.8—Process Blinding Factor



C.51

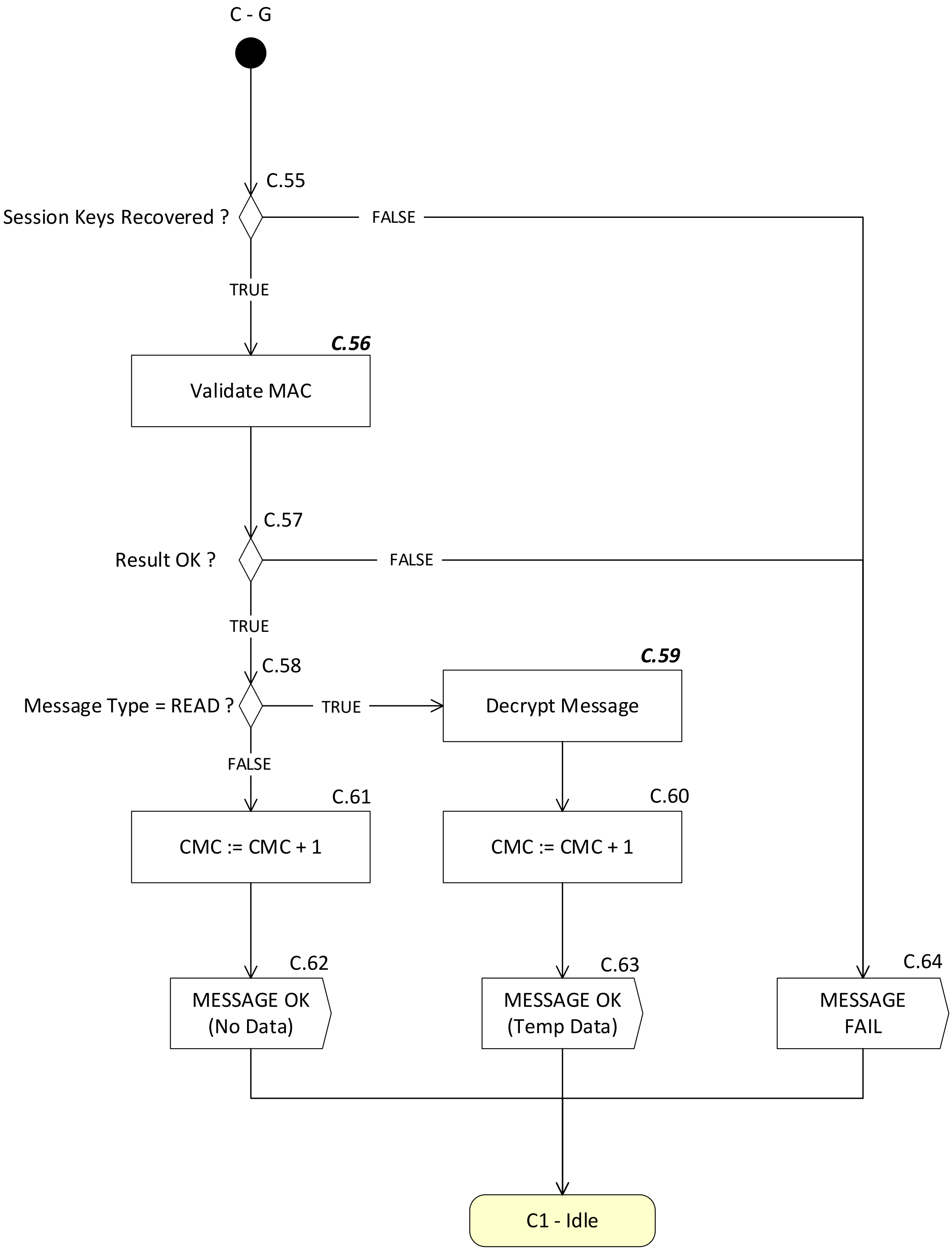
Validate the blinding factor using the Blinded Public Key, ICC ECC Public Key, and Recovered Blinding Factor as specified in [EMV Book E] section 7.3.

Result OK := TRUE if validation succeeds, FALSE if validation fails

7.2.9 Validate Message

Figure 7.9 shows the flow diagram of the processing of the VALIDATE MESSAGE Signal.

Figure 7.9—Validate Message



C.56

```
IF      [Length of P_Message_To_Be_Validated < 8]
THEN
        Result OK := FALSE
ELSE
        MAC := Least significant 8 bytes of P_Message_To_Be_Validated
        Temp Data := P_Message_To_Be_Validated apart from least significant 8 bytes
        IF      [Card Message Counter < 'FFFF']
        THEN
                Temp Data := Card Message Counter || Temp Data
                Temp Data := leftmost 8-bytes of AES-CMAC(Session Key for
                Integrity)[Temp Data]
                Result OK := Temp Data = MAC
                Refer to [EMV Book E] section 8.6.4 for the definition of AES-CMAC.
        ELSE
                Result OK := FALSE
        ENDIF
ENDIF
```

C.59

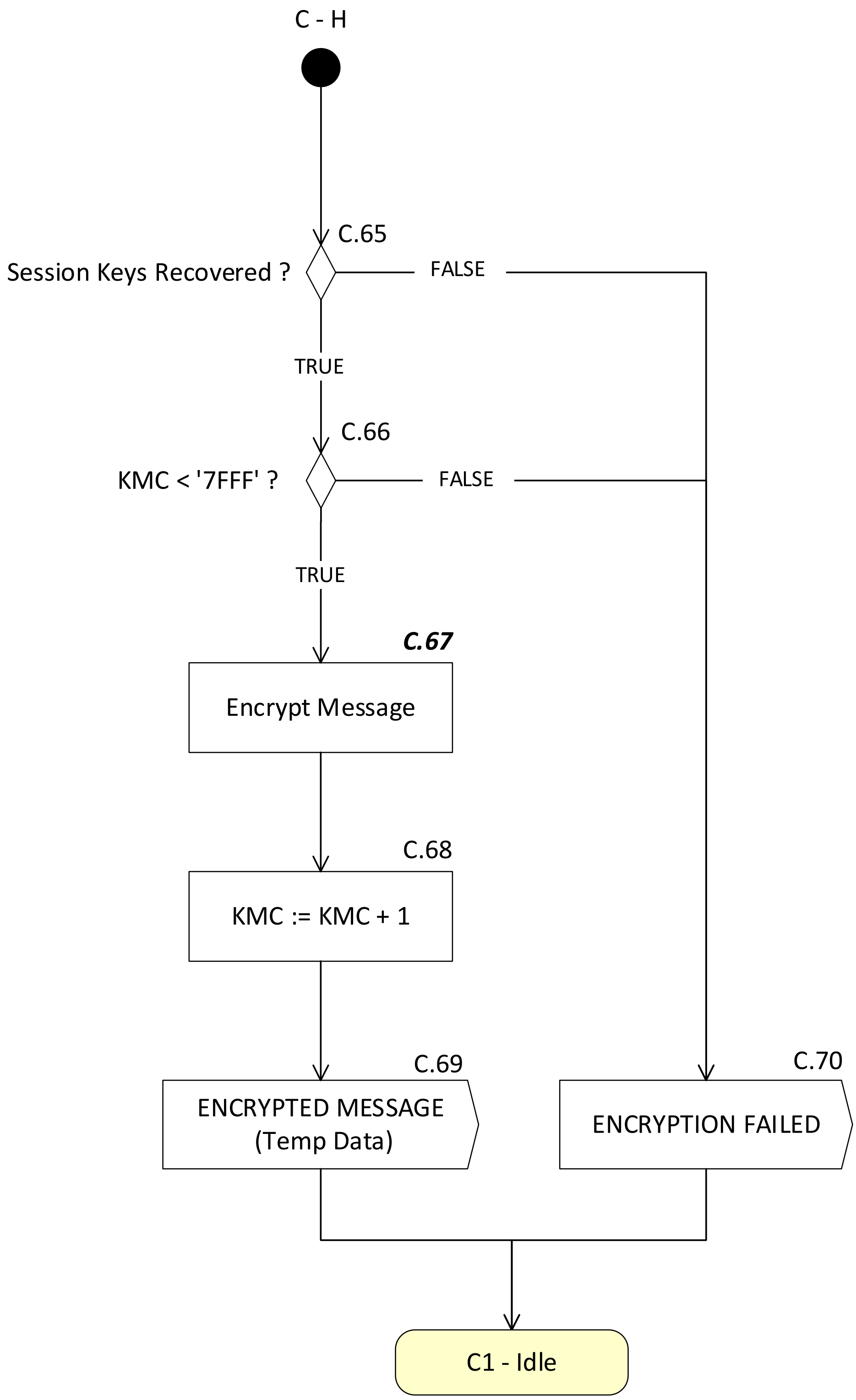
Temp Data := AES-CTR(Session Key for Confidentiality)[Card Message Counter, Temp Data]

Refer to [EMV Book E] section 8.6.2 for the definition of AES-CTR.

7.2.10 Process Encrypt Message

Figure 7.10 shows the flow diagram of the processing of the ENCRYPT MESSAGE Signal.

Figure 7.10—Encrypt Message



C.67

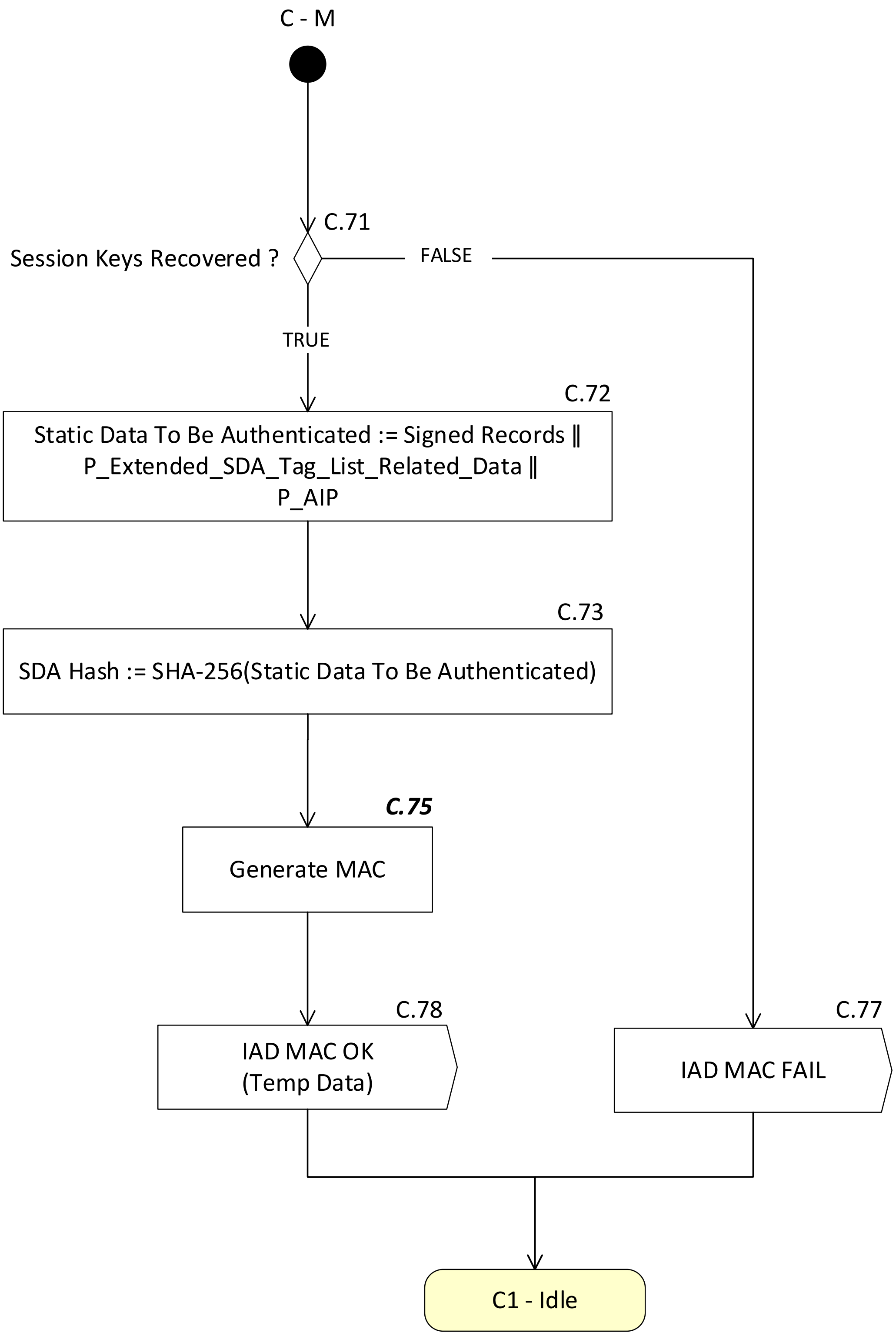
Temp Data := AES-CTR(Session Key for Confidentiality)[Kernel Message Counter, P_Message_To_Be_Encrypted]

Refer to [EMV Book E] section 8.6.2 for the definition of AES-CTR.

7.2.11 Process IAD MAC

Figure 7.11 shows the flow diagram of the processing of the PROCESS IAD MAC Signal.

Figure 7.11—Process IAD MAC



C.75

Temp Data := leftmost 8-bytes of AES-CMAC+ (Session Key for Integrity)['0000' || P_Message_To_Be_Processed || SDA Hash]

Refer to [EMV Book E] section 8.6.5 for the definition of AES-CMAC+.

8 (formerly Security Algorithms)

The content of this chapter has been moved to [EMV Book E].

Annex A Data Dictionary

This annex includes the data dictionary of the data objects used by Kernel 8.

A.1 Data Objects by Name

A.1.1 Account Type

Tag:	'5F57'
Length:	1
Format:	n 2
Update:	K/ACT/DET
Description:	Indicates the type of account selected on the Terminal, coded as specified in [EMV Book 3] Annex G.

A.1.2 Acquirer Identifier

Tag:	'9F01'
Length:	6
Format:	n 6-11
Update:	K
Description:	Uniquely identifies the acquirer within each payment system.

A.1.3 Active AFL

Tag:	—
Length:	var. up to 248
Format:	b
Update:	K
Description:	Contains the AFL indicating the (remaining) terminal file records to be read from the Card. The Active AFL is initialised with the list items referenced in the Application File Locator that are in files with SFI ≤ 10. The Active AFL is updated after each successful READ RECORD.

A.1.4 Active Tag

Tag:	—
Length:	var. up to 2
Format:	b
Update:	K
Description:	Contains the tag requested by the READ DATA command.

A.1.5 Additional Terminal Capabilities

Tag:	'9F40'
Length:	5
Format:	b
Update:	K
Description:	Indicates the data input and output capabilities of the Terminal and Reader. Additional Terminal Capabilities is coded as specified in [EMV Book 4] section A3.

A.1.6 Amount, Authorised (Numeric)

Tag:	'9F02'
Length:	6
Format:	n 12
Update:	K/ACT/DET
Description:	Authorised amount of the transaction (excluding adjustments). This amount is expressed with implicit decimal point corresponding to the minor unit of currency as defined by [ISO 4217] (for example the six bytes '00 00 00 00 01 23' represent USD 1.23 when the currency code is '840').

A.1.7 Amount, Other (Numeric)

Tag:	'9F03'
Length:	6
Format:	n 12
Update:	K/ACT/DET
Description:	Secondary amount associated with the transaction representing a cash back amount. This amount is expressed with implicit decimal point corresponding to the minor unit of currency as defined by [ISO 4217] (for example the 6 bytes '00 00 00 00 01 23' represent GBP 1.23 when the currency code is '826').

A.1.8 Application Cryptogram

Tag:	'9F26'
Length:	8
Format:	b
Update:	K/RA
Description:	Cryptogram returned by the Card in response to the GENERATE AC command.

A.1.9 Application Currency Code

Tag:	'9F42'
Length:	2
Format:	n 3
Update:	K/RA
Description:	Indicates the currency in which the account is managed in accordance with [ISO 4217].

A.1.10 Application Currency Exponent

Tag:	'9F44'
Length:	1
Format:	n 1
Update:	K/RA
Description:	Indicates the implied position of the decimal point from the right of the amount represented in accordance with [ISO 4217].

A.1.11 Application Expiration Date

Tag:	'5F24'
Length:	3
Format:	n 6 (YYMMDD)
Update:	K/RA
Description:	Date after which application expires. The date is expressed in the YYMMDD format. Application Expiration Date is processed as per the requirements outlined in [EMV Book 4] section 6.7.3.

A.1.12 Application File Locator

Tag: '94'

Length: var. multiple of 4 between 4 and 248

Format: b

Update: K/RA

Description: Indicates the location (SFI range of records) of the Application Elementary Files associated with a particular AID and read by the Kernel during a transaction.

The Application File Locator is a list of entries of 4 bytes each. Each entry codes an SFI and a range of records as follows:

- The five most significant bits of the first byte indicate the SFI.
- The second byte indicates the first (or only) record number to be read for that SFI.
- The third byte indicates the last record number to be read for that SFI. When the third byte is greater than the second byte, all the records ranging from the record number in the second byte to and including the record number in the third byte must be read for that SFI. When the third byte is equal to the second byte, only the record number coded in the second byte must be read for that SFI.
- The fourth byte indicates the number of records involved in local authentication starting with the record number coded in the second byte. The fourth byte may range from zero to the value of the third byte less the value of the second byte plus 1.

A.1.13 Application Identifier (Configuration Data)

Tag:	'9F06'
Length:	5-16
Format:	b
Update:	K
Description:	The AID used by Process S to identify the application during application selection and to define the configuration dataset to be used for the transaction.

A.1.14 Application Interchange Profile

Tag:	'82'
Length:	2
Format:	b
Update:	K/RA
Description:	Indicates the capabilities of the Card to support specific functions in the application. The Application Interchange Profile is returned in the response message of the GET PROCESSING OPTIONS command.

Table A.1—Application Interchange Profile

Byte	Bit	Meaning
Byte 1	b8-2	Each bit RFU
	b1	Local authentication supported
Byte 2	b8-5	Each bit RFU
	b4	Offline capable
	b3-2	Copy IAD MAC in IAD 00b: Do not copy IAD MAC in IAD 01b: Copy IAD MAC in IAD using Default IAD MAC Offset 10b: Copy IAD MAC in IAD using IAD MAC Offset 11b: RFU
	b1	Relay resistance protocol is supported

A.1.15 Application Label

Tag:	'50'
Length:	var. up to 16
Format:	ans
Update:	K/RA
Description:	Name associated with the AID, in accordance with [ISO/IEC 7816-5].

A.1.16 Application PAN

Tag:	'5A'
Length:	var. up to 10
Format:	cn var. up to 19
Update:	K/RA
Description:	Valid cardholder account number.

A.1.17 Application PAN Sequence Number

Tag:	'5F34'
Length:	1
Format:	n 2
Update:	K/RA
Description:	Identifies and differentiates cards with the same Application PAN.

A.1.18 Application Preferred Name

Tag:	'9F12'
Length:	var. up to 16
Format:	ans
Update:	K/RA
Description:	Preferred name associated with the AID.

A.1.19 Application Priority Indicator

Tag:	'87'
Length:	1
Format:	b
Update:	K/RA
Description:	Indicates the priority of a given application or group of applications in a directory.

A.1.20 Application Selection Registered Proprietary Data

Tag:	'9F0A'
Length:	var.
Format:	b
Update:	K/RA
Description:	A variable length data object which is governed by EMVCo. The value field of the Application Selection Registered Proprietary Data data object follows the following format:

ID1, L1, V1, ID2, L2, V2,…

Where

- ID is a two byte proprietary data identifier 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 and format is proprietary.

A.1.21 Application Transaction Counter

Tag:	'9F36'
Length:	2
Format:	b
Update:	K/RA
Description:	Counter maintained by the application in the Card.

A.1.22 Application Usage Control

Tag: '9F07'

Length: 2

Format: b

Update: K/RA

Description: Indicates the issuer's specified restrictions on the geographic use and services allowed for the application.

Table A.2—Application Usage Control

Byte	Bit	Meaning
Byte 1	b8	Valid for domestic cash transactions
	b7	Valid for international cash transactions
	b6	Valid for domestic goods
	b5	Valid for international goods
	b4	Valid for domestic services
	b3	Valid for international services
	b2	Valid at ATMs
	b1	Valid at terminals other than ATMs
Byte 2	b8	Domestic cashback allowed
	b7	International cashback allowed
	b6-1	Each bit RFU

A.1.23 Application Version Number (Reader)

Tag: '9F09'

Length: 2

Format: b

Update: K

Description: Version number assigned by the payment system for the Kernel application.

A.1.24 Authenticated Application Data

Tag:	'9F8106'
Length:	var. up to 128
Format:	b
Update:	K/RA
Description:	Returned in the response of the GENERATE AC command and contains BER-TLV-coded data which may be communicated to the issuer.

A.1.25 Cardholder Verification Decision

Tag:	'9F8102'
Length:	1
Format:	b
Update:	K/RA
Description:	Indicates which cardholder verification to be performed.

Table A.3—Cardholder Verification Decision

Byte	Meaning	
Byte 1	Cardholder Verification Decision	
		'00': NO CVM
		'01': SIGNATURE
		'02': ONLINE PIN
		'03': CDCVM
		'0F': N/A
		'FF': CV FAILED
		Other values RFU

A.1.26 Card Capabilities Information

Tag: '9F810D'

Length: 2

Format: b

Update: K/RA

Description: Indicates the CVM and interface capabilities of the Card.

Table A.4—Card Capabilities Information

Byte	Bit	Meaning
Byte 1	b8	RFU
	b7	Magnetic stripe
	b6	Contact interface
	b5	Contactless interface
	b4-1	Each bit RFU
Byte 2	b8	Offline plaintext PIN
	b7	Enciphered PIN for online verification
	b6	Signature
	b5	Offline Enciphered PIN
	b4	NO CVM
	b3	CDCVM
	b2-1	Each bit RFU

A.1.27 Card Data Input Capability

Tag: '9F8206'

Length: 1

Format: b

Update: K

Description: Indicates the card data input capability of the Terminal and Reader.

Table A.5—Card Data Input Capability

Byte	Bit	Meaning
Byte 1	b8	Manual key entry
	b7	Magnetic stripe
	b6	IC with contacts
	b5-1	Each bit RFU

A.1.28 Card Key Data

Tag: '9F8103'

Length: 64 or 132

Format: b

Update: K/RA

Description: Includes the *x*-coordinate of the ECC blinded public key point (bytes 1 to N_{FIELD}) and the encrypted blinding factor (bytes N_{FIELD}+1 to 2*N_{FIELD}) returned by the Card in the GET PROCESSING OPTIONS response.

A.1.29 Card Qualifier

Tag: '9F2C'
Length: 7
Format: b
Update: K/RA
Description: Indicates the features supported by the Card.

Table A.6—Card Qualifier

Byte	Bit	Meaning
Byte 1	b8-1	Version '00': Not allowed '01': VERSION 1 '02': VERSION 2 Other values RFU
Byte 2-4		Secure channel ASIs supported by the Card, in the order preferred by the Card starting at the leftmost byte (byte 2). The Card may support up to three secure channel ASIs. Cards that support less than three secure channel ASIs fill the remaining unused bytes with 'FF'. Refer to [EMV Book E] section 8.4 for the definition of ASIs.
Byte 5	b8	Support for field off detection
	b7-1	Each bit RFU
Byte 6-7		RFU

A.1.30 Card TVR

Tag: '9F8104'
Length: 5
Format: b
Update: K/RA
Description: Terminal Verification Results returned by the Card in the response to the GENERATE AC command.

A.1.31 CDOL1

Tag:	'8C'
Length:	var. up to 250
Format:	b
Update:	K/RA
Description:	A data object in the Card that provides the Kernel with a list of data objects that must be passed to the Card in the data field of the GENERATE AC command.

A.1.32 CDOL1 Related Data

Tag:	—
Length:	var.
Format:	b
Update:	K
Description:	Command data field of the GENERATE AC command, coded according to CDOL1.

A.1.33 Certification Authority Public Key Index (Card)

Tag:	'8F'
Length:	1
Format:	b
Update:	K/RA
Description:	Identifies the Certification Authority Public Key in conjunction with the RID.

A.1.34 Cryptogram Information Data

Tag:	'9F27'
Length:	1
Format:	b
Update:	K/RA
Description:	Indicates the type of cryptogram and the actions to be performed by the Kernel.

Table A.7—Cryptogram Information Data

Byte	Bit	Meaning	
Byte 1	b8-7	AC type	
			00: AAC
			01: TC
			10: ARQC
			11: RFU
	b6-1	See [EMV Book 3] Table 15	

A.1.35 Crypto Read Data Counter

Tag:	—
Length:	1
Format:	b
Update:	K
Description:	Represents the number of READ DATA responses that are being validated by Process C at a certain moment in time.

A.1.36 Crypto Read Record Counter

Tag:	—
Length:	1
Format:	b
Update:	K
Description:	Represents the number of encrypted records that are being decrypted by Process C at a certain moment in time.

A.1.37 CVM Capability – CVM Required

Tag: '9F8207'

Length: 1

Format: b

Update: K

Description: Indicates the CVM capability of the Terminal and Reader to be used when the transaction amount is greater than the Reader CVM Required Limit.

Table A.8—CVM Capability – CVM Required

Byte	Bit	Meaning
Byte 1	b8	RFU
	b7	Enciphered PIN for online verification
	b6	Signature
	b5	RFU
	b4	No CVM required
	b3	CDCVM
	b2-1	Each bit RFU

A.1.38 CVM Capability – No CVM Required

Tag: '9F8208'

Length: 1

Format: b

Update: K

Description: Indicates the CVM capability of the Terminal and Reader to be used when the transaction amount is less than or equal to the Reader CVM Required Limit.

Table A.9—CVM Capability – No CVM Required

Byte	Bit	Meaning
Byte 1	b8	RFU
	b7	Enciphered PIN for online verification
	b6	Signature
	b5	RFU
	b4	No CVM required
	b3	CDCVM
	b2-1	Each bit RFU

A.1.39 CVM Results

Tag: '9F34'

Length: 3

Format: b

Update: K

Description: Indicates the results of the last CVM performed.

The CVM Results are coded as specified in [EMV Book 4] section A.4.

Table A.10—CVM Results

Byte	Bit	Meaning
Byte 1	b8-1	CVM Performed
Byte 2	b8-1	CVM Condition
Byte 3	b8-1	CVM Result

A.1.40 Data Envelope 1 – 10

Tag:	'9F8111' – '9F811A'
Length:	var. up to 243
Format:	b
Update:	K/RA
Description:	The Data Envelopes contain proprietary information from the issuer, payment system or third party. The Data Envelope can be retrieved with the READ DATA command and updated with the WRITE DATA command.

A.1.41 Data Envelopes To Write

Tag:	'BF8104'
Length:	var.
Format:	b
Update:	K/ACT/DET
Description:	Contains the Terminal data writing requests to be sent to the Card after processing the GENERATE AC command. The value of this data object is composed of a series of TLVs. The TLVs in Data Envelopes To Write are coded according to the BER-TLV coding rules in section 4.7.1. This data object may be provided several times by the Terminal in a series of DET Signals. Therefore, these values must be accumulated in Data Envelopes To Write Yet.

A.1.42 Data Envelopes To Write Yet

Tag:	—
Length:	var.
Format:	b
Update:	K
Description:	List of data objects that contains the accumulated Terminal data writing requests received in Data Envelopes To Write. The TLVs in Data Envelopes To Write Yet are coded according to the BER-TLV coding rules in section 4.7.1.

A.1.43 Data Needed

Tag: '9F8201'
Length: var.
Format: b
Update: K
Description: List of tags included in the DEK Signal to request information from the Terminal. The tags in Data Needed are coded according to the BER-TLV coding rules in section 4.7.1.

A.1.44 Data Record

Tag: 'BF8102'
Length: var.
Format: b
Update: K
Description: A list of TLV-coded data objects returned in the OUT Signal on the completion of transaction processing. The Data Record contains the necessary data objects for authorisation and clearing as shown in Table A.11. The data objects in the Data Record are coded according to the BER-TLV coding rules in section 4.7.1.

Table A.11—Data Objects Included in Data Record

Tag	Data Object
'9F02'	Amount, Authorised (Numeric)
'9F03'	Amount, Other (Numeric)
'9F26'	Application Cryptogram
'5F24'	Application Expiration Date
'82'	Application Interchange Profile
'50'	Application Label
'5A'	Application PAN
'5F34'	Application PAN Sequence Number
'9F12'	Application Preferred Name
'9F36'	Application Transaction Counter
'9F07'	Application Usage Control
'9F09'	Application Version Number (Reader)
'9F8106'	Authenticated Application Data

Tag	Data Object
'9F810D'	Card Capabilities Information
'9F27'	Cryptogram Information Data
'9F34'	CVM Results
'84'	DF Name
'9F1E'	Interface Device Serial Number
'9F10'	Issuer Application Data
'9F8109'	Issuer Application Data MAC
'9F11'	Issuer Code Table Index
'9F24'	Payment Account Reference
'9F33'	Terminal Capabilities
'9F1A'	Terminal Country Code
'9F35'	Terminal Type
'95'	Terminal Verification Results
'9F19'	Token Requester ID
'57'	Track 2 Equivalent Data
'5F2A'	Transaction Currency Code
'9A'	Transaction Date
'9C'	Transaction Type
'9F37'	Unpredictable Number

A.1.45 Data To Send

Tag:	'BF8101'
Length:	var.
Format:	b
Update:	K
Description:	A list of TLV-coded data objects that contains the accumulated data sent by the Kernel to the Terminal in a DEK Signal. The data objects in Data To Send are coded according to the BER-TLV coding rules in section 4.7.1. These data objects may correspond to Terminal reading requests, obtained from the Card by means of READ DATA or READ RECORD commands, or may correspond to data that the Kernel posts to the Terminal as part of its own processing.

A.1.46 Default CDOL1

Tag:	'9F8220'
Length:	var. up to 250
Format:	b
Update:	K
Description:	The default value for CDOL1 that the Kernel uses to construct the value field of the GENERATE AC command when the Card does not return a CDOL1.

A.1.47 Default IAD MAC Offset

Tag:	'9F821E'
Length:	1
Format:	b
Update:	K
Description:	The default offset of the Issuer Application Data MAC in Issuer Application Data when 'Copy IAD MAC in IAD' in Application Interchange Profile indicates that the Default IAD MAC Offset must be used. The offset is zero-based.

A.1.48 Device Estimated Transmission Time For Relay Resistance R-APDU

Tag:	—
Length:	2
Format:	b
Update:	K/RA
Description:	Indicates the time the Card expects to need for transmitting the EXCHANGE RELAY RESISTANCE DATA R-APDU. The Device Estimated Transmission Time For Relay Resistance R-APDU is expressed in units of hundreds of microseconds.

A.1.49 Device Relay Resistance Entropy

Tag:	—
Length:	4
Format:	b
Update:	K/RA
Description:	Unpredictable number returned by the Card in the response to the EXCHANGE RELAY RESISTANCE DATA command.

A.1.50 DF Name

Tag:	'84'
Length:	var. 5 to 16
Format:	b
Update:	K/RA
Description:	Identifies the name of the DF, as described in [ISO/IEC 7816-4].

A.1.51 Discretionary Data Tag List

Tag:	'9F821F'
Length:	var.
Format:	b
Update:	K
Description:	Contains a list of tags of data objects to be included in the Discretionary Data. The tags in Discretionary Data Tag List are coded according to the BER-TLV coding rules in section 4.7.1.

A.1.52 Discretionary Data

Tag: 'BF8103'

Length: var.

Format: b

Update: K

Description: A TLV-coded list of Kernel-specific data objects sent to the Terminal as a separate field in the OUT Signal and includes the data objects of which the tags are listed in Discretionary Data Tag List. The data objects in Discretionary Data are coded according to the BER-TLV coding rules in section 4.7.1.

A.1.53 EDA Status

Tag: —

Length: 1

Format: b

Update: K

Description: Indicates the progress made in performing local authentication and Enhanced Data Authentication MAC processing.

Table A.12—EDA Status

Byte	Bit	Meaning
Byte 1	b8	Process Issuer Certificate Requested
	b7	Process ICC Certificate Requested
	b6	Process EDA MAC Requested
	b5	Process Blinding Factor Requested
	b4	Issuer Certificate Processed
	b3	ICC Certificate Processed
	b2	EDA MAC Processed
	b1	Blinding Factor Processed

A.1.54 Enhanced Data Authentication MAC

Tag:	'9F8105'
Length:	8
Format:	b
Update:	K/RA
Description:	A MAC over the Application Cryptogram and Issuer Application Data MAC.

A.1.55 Error Indication

Tag:	'9F8204'
Length:	6
Format:	b
Update:	K
Description:	Contains information regarding the nature of the error that has been encountered during the transaction processing. This data object should be part of the Discretionary Data.

Table A.13—Error Indication

Data Field	Length	Format
L1	1	b (see Table A.14)
L2	1	b (see Table A.15)
L3	1	b (see Table A.16)
SW12	2	b
Msg On Error	1	b (see Message Identifier as defined in Table A.20b)

Table A.14—Error Indication – L1

Byte	Bit	Meaning	
Byte 1	b8-1	L1	
			00000000: OK
			00000001: TIMEOUT ERROR
			00000010: TRANSMISSION ERROR
			00000011: PROTOCOL ERROR
			Other values: RFU

Table A.15—Error Indication – L2

Byte	Bit	Meaning	
Byte 1	b8-1	L2	
			00000000: OK
			00000001: CARD DATA MISSING
			00000010: RFU
			00000011: STATUS BYTES
			00000100: PARSING ERROR
			00000101: RFU
			00000110: CARD DATA ERROR
			00000111 – 00001110: RFU
			00001111: TERMINAL DATA ERROR
			00010000: DECRYPTION FAILED
			00010001: RFU
			00010010: IAD MAC FAILED
			00010011: EDA MAC FAILED
			Other values: RFU

Table A.16—Error Indication – L3

Byte	Bit	Meaning	
Byte 1	b8-1	L3	
			00000000: OK
			00000001: TIMEOUT
			00000010: RFU
			00000011: TRANSACTION DATA MISSING
			Other values: RFU

A.1.56 Extended SDA Tag List

Tag:	'9F810A'
Length:	var.
Format:	b
Update:	K/RA
Description:	Contains a list of the tags of data objects to be included in the Static Data To Be Authenticated.

A.1.57 Extended SDA Tag List Related Data

Tag:	—
Length:	var.
Format:	b
Update:	K
Description:	Data object containing the concatenation of the TLV-coded data objects referenced in the Extended SDA Tag List.

A.1.58 File Control Information Issuer Discretionary Data

Tag:	'BF0C'
Length:	var. up to 220
Format:	b
Update:	N/A (this template is not stored in the TLV Database)
Description:	Issuer discretionary part of the File Control Information Template.

A.1.59 File Control Information Proprietary Template

Tag:	'A5'
Length:	var. up to 240
Format:	b
Update:	N/A (this template is not stored in the TLV Database)
Description:	Identifies the data object proprietary to this specification in the File Control Information Template, in accordance with [ISO/IEC 7816-4].

A.1.60 File Control Information Template

Tag: '6F'

Length: var. up to 250

Format: b

Update: N/A (this template is not stored in the TLV Database)

Description: Identifies the File Control Information Template, in accordance with [ISO/IEC 7816-4]. Table A.17 gives the File Control Information Template expected in response to a successful selection of a Card application matching Kernel 8.

Table A.17—File Control Information Template of an ADF

<table>
<tr><th>Tag</th><th colspan="4">Value</th></tr>
<tr><td rowspan="12">'6F'</td><td colspan="4">File Control Information Template</td></tr>
<tr><td>'84'</td><td colspan="3">DF Name</td></tr>
<tr><td rowspan="10">'A5'</td><td colspan="3">File Control Information Proprietary Template</td></tr>
<tr><td>'50'</td><td colspan="2">Application Label</td></tr>
<tr><td>'87'</td><td colspan="2">Application Priority Indicator</td></tr>
<tr><td>'5F2D'</td><td colspan="2">Language Preference</td></tr>
<tr><td>'9F38'</td><td colspan="2">PDOL</td></tr>
<tr><td>'9F11'</td><td colspan="2">Issuer Code Table Index</td></tr>
<tr><td>'9F12'</td><td colspan="2">Application Preferred Name</td></tr>
<tr><td rowspan="4">'BF0C'</td><td colspan="2">File Control Information Issuer Discretionary Data</td></tr>
<tr><td>'9F0A'</td><td>Application Selection Registered Proprietary Data</td></tr>
<tr><td>'9F2C'</td><td>Card Qualifier</td></tr>
<tr><td>'XXXX'</td><td>One or more additional data objects from application provider, Issuer, or ICC supplier</td></tr>
</table>

A.1.61 GENERATE AC Response Message Data Field

Tag:	—
Length:	var. up to 256
Format:	b
Update:	K
Description:	TLV-coded string included in R-APDU of GENERATE AC. (For details, see Table 5.8.) This data object is included in the data dictionary because it is used in multiple states (26 and 27).

A.1.62 Hold Time Value

Tag:	'9F8212'
Length:	1
Format:	b
Update:	K
Description:	Indicates the time that the field is to be turned off after the transaction is completed if requested to do so by the Card. The Hold Time Value is in units of 100ms.

A.1.63 IAD MAC Offset

Tag:	'9F8107'
Length:	1
Format:	b
Update:	K/RA
Description:	Indicates the offset of the Issuer Application Data MAC in Issuer Application Data when 'Copy IAD MAC in IAD' in Application Interchange Profile indicates that the IAD MAC Offset must be used. The offset is zero-based.

A.1.64 ICC ECC Public Key (for RSA Certificates)

Tag:	'9F810B'
Length:	32 or 66
Format:	b
Update:	K/RA
Description:	The ICC ECC Public Key (*x*-coordinate of ICC ECC Public Key point) is returned in a record referenced in the AFL in case RSA certificates are used.

A.1.65 ICC Public Key Certificate

Tag:	'9F46'
Length:	var. up to 248
Format:	b
Update:	K/RA
Description:	ICC public key certified by the issuer. The ICC Public Key Certificate is used to store RSA and ECC ICC public key certificates.

A.1.66 ICC RSA Public Key Exponent

Tag:	'9F47'
Length:	1 or 3
Format:	b
Update:	K/RA
Description:	Exponent of the ICC public key.

A.1.67 ICC RSA Public Key Remainder

Tag:	'9F48'
Length:	var.
Format:	b
Update:	K/RA
Description:	Remaining digits of the modulus of the ICC public key.

A.1.68 Interface Device Serial Number

Tag:	'9F1E'
Length:	8
Format:	an
Update:	K
Description:	Unique and permanent serial number assigned to the IFD by the manufacturer.

A.1.69 Issuer Application Data

Tag:	'9F10'
Length:	var. up to 32
Format:	b
Update:	K/RA
Description:	Contains proprietary application data for transmission to the issuer in an online transaction.

A.1.70 Issuer Application Data MAC

Tag:	'9F8109'
Length:	8
Format:	b
Update:	K
Description:	A MAC over static card data and transaction related data. The Issuer Application Data MAC may be copied by the Kernel in the Issuer Application Data as indicated by 'Copy IAD MAC in IAD' in Application Interchange Profile.

A.1.71 Issuer Code Table Index

Tag:	'9F11'
Length:	1
Format:	n 2
Update:	K/RA
Description:	Indicates the code table, in accordance with [ISO/IEC 8859], for displaying the Application Preferred Name. The Issuer Code Table Index is coded as specified in [EMV Book 3] section C.4.

A.1.72 Issuer Identification Number

Tag:	'42'
Length:	3
Format:	n 6
Update:	K/RA
Description:	The number that identifies the major industry and the Card issuer and that forms the first part of the Application PAN.

A.1.73 Issuer Identification Number Extended

Tag:	'9F0C'
Length:	var. 3 or 4
Format:	n 6 or 8
Update:	K/RA
Description:	The number that identifies the major industry and the Card issuer and that forms the first part (6 or 8-digits) of the Application PAN. While the first 6-digits of the Issuer Identification Number Extended and Issuer Identification Number are the same and there is no need to have both data objects on the Card, Cards may have both the Issuer Identification Number and Issuer Identification Number Extended data objects present.

A.1.74 Issuer Public Key Certificate

Tag:	'90'
Length:	var. up to 248
Format:	b
Update:	K/RA
Description:	Issuer public key certified by a certification authority. The Issuer Public Key Certificate is used to store RSA and ECC issuer certificates.

A.1.75 Issuer RSA Public Key Exponent

Tag:	'9F32'
Length:	1 or 3
Format:	b
Update:	K/RA
Description:	Exponent used for the recovery and verification of the ICC Public Key Certificate.

A.1.76 Issuer RSA Public Key Remainder

Tag:	'92'
Length:	$N_I – N_{CA} + 36$
Format:	b
Update:	K/RA
Description:	Remaining digits of the modulus of the Issuer public key.

A.1.77 Kernel Configuration

Tag: '9F8209'

Length: 2

Format: b

Update: K

Description: Indicates the Kernel Configuration Options.

Table A.18—Kernel Configuration

Byte	Bit	Meaning
Byte 1	b8-7	Each bit RFU
	b6	RSA certificates enabled
	b5	Relay resistance protocol enabled
	b4	Report local authentication failed in TVR (1)
	b3-1	Each bit RFU
Byte 2	b8-1	Each bit RFU

(1) 'Local authentication failed' bit in TVR is set after GENERATE AC command and may impact Application Cryptogram verification.

A.1.78 Kernel Decision

Tag: —
Length: 1
Format: b
Update: K
Description: An internal working variable used to indicate the decision of the Kernel about the outcome of the transaction.

Table A.19—Kernel Decision

<table>
<tr><th>Byte</th><th>Bit</th><th colspan="2">Meaning</th></tr>
<tr><td rowspan="6">Byte 1</td><td rowspan="5">b8-7</td><td colspan="2">Decision</td></tr>
<tr><td rowspan="4"></td><td>00: DECLINE</td></tr>
<tr><td>01: ACCEPT</td></tr>
<tr><td>10: ONLINE</td></tr>
<tr><td>11: RFU</td></tr>
<tr><td>b6-1</td><td colspan="2">Each bit RFU</td></tr>
</table>

A.1.79 Kernel Key Data

Tag: '9E'
Length: var. up to 132
Format: b
Update: K
Description: Used to transfer the (x, y) coordinates of the ephemeral kernel public key to the Card in the value field of the GET PROCESSING OPTIONS command.

A.1.80 Kernel Qualifier

Tag: '9F2B'

Length: 8

Format: b

Update: K

Description: Indicates to the Card any Kernel specific data that it needs to communicate. Kernel Qualifier is built by the Kernel with data from different configuration data objects.

Table A.20—Kernel Qualifier

Byte	Bit	Meaning
Byte 1	b8-1	Version '00': Not allowed '01': Not allowed '02': VERSION 2 Other values RFU
Byte 2	b8	Local authentication enabled
	b7-1	Each bit RFU
Byte 3		SC ASI Secure channel ASI selected by the Kernel
Byte 4-6		C ASI List Certificate ASIs supported by the Kernel, starting at the leftmost byte (byte 4). The Kernel may indicate support for up to three certificate ASIs. If less than three Certificate ASIs are included, then the remaining unused bytes are filled with 'FF'. Refer to [EMV Book E] section 8.4 for the definition of ASIs.
Byte 7-8		RFU

A.1.81 Kernel Reserved TVR Mask

Tag:	'9F821A'
Length:	5
Format:	b
Update:	K
Description:	Determines which bits in the Terminal Verification Results cannot be altered by the Card TVR returned by the Card in the response to the GENERATE AC command. The bits set to 1b cannot be altered by the Card.

A.1.82 Language Preference

Tag:	'5F2D'
Length:	2-8
Format:	an
Update:	K/RA
Description:	1-4 languages stored in order of preference, each represented by two alphabetical characters, in accordance with [ISO 639-1].

A.1.83 Last ERRD Response

Tag:	—
Length:	10
Format:	b
Update:	K
Description:	Contains the response message without Tag '80' and length '0A' of the last ERRD command.

A.1.84 Log Entry

Tag:	'9F4D'
Length:	2
Format:	b
Update:	K/RA
Description:	Provides the SFI of the Transaction Log file and its number of records.

A.1.85 Maximum Relay Resistance Grace Period

Tag:	'9F8214'
Length:	2
Format:	b
Update:	K
Description:	The Minimum Relay Resistance Grace Period and Maximum Relay Resistance Grace Period represent how far outside the window defined by the Card that the measured time may be and yet still be considered acceptable. The Maximum Relay Resistance Grace Period is expressed in units of hundreds of microseconds.

A.1.86 Max Time For Processing Relay Resistance APDU

Tag:	—
Length:	2
Format:	b
Update:	K/RA
Description:	Indicates the maximum estimated time the Card requires for processing the EXCHANGE RELAY RESISTANCE DATA command. The Max Time For Processing Relay Resistance APDU is expressed in units of hundreds of microseconds.

A.1.87 Measured Relay Resistance Processing Time

Tag:	—
Length:	2
Format:	b
Update:	K
Description:	Contains the time measured by the Kernel for processing the EXCHANGE RELAY RESISTANCE DATA command. The Measured Relay Resistance Processing Time is expressed in units of hundreds of microseconds.

A.1.88 Merchant Category Code

Tag:	'9F15'
Length:	2
Format:	n 4
Update:	K
Description:	Classifies the type of business being done by the merchant, represented in accordance with [ISO 8583:1993] for Card Acceptor Business Code.

A.1.89 Merchant Identifier

Tag:	'9F16'
Length:	15
Format:	ans 15
Update:	K
Description:	When concatenated with the Acquirer Identifier, uniquely identifies a given merchant.

A.1.90 Merchant Name and Location

Tag:	'9F4E'
Length:	var.
Format:	ans
Update:	K
Description:	Indicates the name and location of the merchant.

A.1.91 Message Hold Time

Tag:	'9F8211'
Length:	3
Format:	n 6
Update:	K
Description:	Indicates the default delay for the processing of the next MSG Signal. The Message Hold Time is an integer in units of 100ms.

A.1.92 Message Identifier

Tag: —
Length: 1
Format: b
Update: —
Description: Identifies the message to be displayed to the cardholder.

Table A.20b—Message Identifier

Message Identifier	Meaning
'17'	CARD READ OK
'21'	TRY AGAIN
'03'	APPROVED
'1A'	APPROVED – SIGN
'07'	DECLINED
'18'	TRY ANOTHER INTERFACE
'1C'	ERROR – OTHER CARD
'1D'	INSERT CARD
'20'	SEE PHONE
'1B'	AUTHORISING – PLEASE WAIT
'1E'	CLEAR DISPLAY
'FF'	N/A
	Other values: RFU

A.1.93 Message Identifiers On Restart

Tag:	'9F821D'
Length:	var. up to 32
Format:	b
Update:	K
Description:	The Message Identifiers On Restart is a configuration data object that defines the message identifiers that can be used by the Card in 'Message Identifier' in Restart Indicator. Each byte of the data object contains one message identifier as shown in Table A.21. Message Identifiers On Restart must only contain message identifiers that are supported by Process D.

Table A.21—Message Identifiers On Restart

Byte	Message Identifier
Byte 1	Message Identifier 1
Byte 2	Message Identifier 2
Byte 3	Message Identifier 3
…	…
Byte n	Message Identifier n

A.1.94 Message To Validate

Tag:	—
Length:	var.
Format:	b
Update:	K
Description:	An internal working variable used to store the message to validate by Process C for the WRITE DATA commands.

A.1.95 Minimum Relay Resistance Grace Period

Tag:	'9F8213'
Length:	2
Format:	b
Update:	K
Description:	The Minimum Relay Resistance Grace Period and Maximum Relay Resistance Grace Period represent how far outside the window defined by the Card that the measured time may be and yet still be considered acceptable. The Minimum Relay Resistance Grace Period is expressed in units of hundreds of microseconds.

A.1.96 Min Time For Processing Relay Resistance APDU

Tag:	—
Length:	2
Format:	b
Update:	K/RA
Description:	Indicates the minimum estimated time the Card requires for processing the EXCHANGE RELAY RESISTANCE DATA command. The Min Time For Processing Relay Resistance APDU is expressed in units of hundreds of microseconds.

A.1.97 Next Cmd

Tag:	—
Length:	1
Format:	b
Update:	K
Description:	An internal working variable used to indicate the C-APDU that is currently being processed by the Card.

Table A.22—Next Cmd

Byte	Meaning	
Byte 1	Next Cmd	
		'00': READ RECORD
		'01': READ DATA
		'02': NONE
		Other values RFU

A.1.98 Outcome Parameter Set

Tag:	'9F8210'
Length:	8
Format:	b
Update:	K
Description:	Used to indicate to the Terminal the outcome of the transaction processing by the Kernel. Its value is an accumulation of results about applicable parts of the transaction.

Table A.23—Outcome Parameter Set

Byte	Bit	Meaning	
Byte 1	b8-5	Status	
			0001: APPROVED
			0010: DECLINED
			0011: ONLINE REQUEST
			0100: END APPLICATION
			0101: SELECT NEXT
			0110: TRY ANOTHER INTERFACE
			0111: TRY AGAIN
			1111: N/A
			Other values: RFU
	b4-1	Each bit RFU	
Byte 2	b8-5	Start	
			0000: A
			0001: B
			0010: C
			0011: D
			1111: N/A
			Other values: RFU
	b4-1	Each bit RFU	

Byte	Bit	Meaning	
Byte 3	b8-5	Online Response Data	
			1111: N/A
			Other values: RFU
	b4-1	Each bit RFU	
Byte 4	b8-5	CVM	
			0000: NO CVM
			0001: OBTAIN SIGNATURE
			0010: ONLINE PIN
			0011: CONFIRMATION CODE VERIFIED
			1111: N/A
			Other values: RFU
	b4-1	Each bit RFU	
Byte 5	b8	UI Request on Outcome Present	
	b7	UI Request on Restart Present	
	b6	Data Record Present	
	b5	Discretionary Data Present	
	b4	Receipt	
			0: N/A
			1: YES
	b3-1	Each bit RFU	
Byte 6	b8-5	Alternate Interface Preference	
			0001: Contact Chip
			0010: Mag stripe
			1111: N/A
			Other values: RFU
	b4-1	Each bit RFU	
Byte 7	b8-1	Field Off Request	
			11111111: N/A
			Other values: Hold time in units of 100 ms
Byte 8	b8-1	Removal Timeout in units of 100 ms	

A.1.99 Payment Account Reference

Tag:	'9F24'
Length:	29
Format:	an
Update:	K/RA
Description:	The Payment Account Reference is a data object associated with an Application PAN. It allows acquirers and merchants to link transactions, whether tokenised or not, that are associated to the same underlying Application PAN. Lower case alphabetic characters are not permitted for the Payment Account Reference, however the Kernel is not expected to check this.

A.1.100 PDOL

Tag:	'9F38'
Length:	var. up to 240
Format:	b
Update:	K/RA
Description:	A data object in the Card that provides the Kernel with a list of data objects that must be passed to the Card in the GET PROCESSING OPTIONS command.

A.1.101 PDOL Related Data

Tag:	—
Length:	var.
Format:	b
Update:	K
Description:	Value of the command data field of the GET PROCESSING OPTIONS command.

A.1.102 PDOL Values

Tag:	—
Length:	var.
Format:	b
Update:	K
Description:	The value field of the Command Template (tag '83') stored in PDOL Related Data. PDOL Values is created by the Kernel according to PDOL as a concatenated list of data objects without tags or lengths following the rules specified in section 4.1.4. If PDOL is not returned by the Card, then PDOL Values is an empty string.

A.1.103 Proceed To First Write Flag

Tag:	'9F8202'
Length:	1
Format:	b
Update:	K/ACT/DET
Description:	Indicates that the Terminal will send no more requests to read data other than as indicated in Tags To Read. This data item indicates the point at which the Kernel shifts from the Card reading phase to the Card writing phase. If Proceed To First Write Flag is not present or is present with non-zero length and value other than zero, then the Kernel proceeds without waiting. If Proceed To First Write Flag is present with zero length, then the Kernel sends a DEK Signal to the Terminal and waits for the DET Signal. If Proceed To First Write Flag is present with non-zero length and value equal to zero, then the Kernel waits for a DET Signal from the Terminal without sending a DEK Signal.

A.1.104 Reader Contactless Floor Limit

Tag:	'9F820D'
Length:	6
Format:	n 12
Update:	K
Description:	Indicates the transaction amount above which transactions must be authorised online.

A.1.105 Reader CVM Required Limit

Tag:	'9F820E'
Length:	6
Format:	n 12
Update:	K
Description:	Indicates the transaction amount above which the Kernel instantiates the CVM capabilities field in Terminal Capabilities with CVM Capability – CVM Required.

A.1.106 Read Data Status

Tag:	'9F821C'
Length:	1
Format:	b
Update:	K
Description:	Information reported by the Kernel to the Terminal, about the processing of READ DATA commands. Possible values are 'completed' or 'not completed'. In the latter case, this status is not specific about which of the READ DATA commands failed, or about how many of these commands have failed or succeeded. This data object is part of the Discretionary Data provided by the Kernel to the Terminal.

Table A.24—Read Data Status

Byte	Bit	Meaning
Byte 1	b8	Completed
	b7-1	Each bit RFU

A.1.107 Read Data Tags To Validate Yet

Tag: —

Length: var.

Format: b

Update: K

Description: List of tags that contains the tags sent by the Kernel to the Card in the READ DATA command(s). The tags in Read Data Tags To Validate Yet are coded according to the BER-TLV coding rules in section 4.7.1.

The tag is removed from this list if the tag returned in the response of READ DATA command is present in the list.

Read Data Tags To Validate Yet is initialised with an empty list when the Kernel is started.

A.1.108 Reference Control Parameter

Tag: —

Length: 1

Format: b

Update: K

Description: Working variable to store the reference control parameter of the GENERATE AC command.

Table A.25—Reference Control Parameter

Byte	Bit	Meaning	
Byte 1	b8-7	AC type	
			00: AAC
			01: TC
			10: ARQC
			11: RFU
	b6-1	Each bit RFU	

A.1.109 Relay Resistance Accuracy Threshold

Tag:	'9F8217'
Length:	2
Format:	b
Update:	K
Description:	Represents the threshold above which the Kernel considers the variation between Measured Relay Resistance Processing Time and Min Time For Processing Relay Resistance APDU no longer acceptable. The Relay Resistance Accuracy Threshold is expressed in units of hundreds of microseconds.

A.1.110 Relay Resistance Time Excess

Tag:	'9F810C'
Length:	2
Format:	b
Update:	K
Description:	Contains the excess time on top of the Max Time For Processing Relay Resistance APDU used by the card to process the EXCHANGE RELAY RESISTANCE DATA command. The Relay Resistance Time Excess is expressed in units of hundreds of microseconds.

A.1.111 Relay Resistance Transmission Time Mismatch Threshold

Tag:	'9F8218'
Length:	1
Format:	b
Update:	K
Description:	Represents the threshold above which the Kernel considers the variation between Device Estimated Transmission Time For Relay Resistance R-APDU and Terminal Expected Transmission Time For Relay Resistance R-APDU no longer acceptable. The Relay Resistance Transmission Time Mismatch Threshold is a percentage and expressed as an integer.

A.1.112 Response Message Template Format 2

Tag:	'77'
Length:	var. up to 253
Format:	b
Update:	N/A (this template is not stored in the TLV Database)
Description:	Contains the data objects (with tags and lengths) returned by the Card in response to a command.

A.1.113 Restart Indicator

Tag:	'9F8108'
Length:	var. 2 to 5
Format:	b
Update:	K/RA
Description:	Indicator returned by the Card that comprises two fields: an indication that restart is needed and a message indicator.

Table A.26—Restart Indicator

<table>
<tr><th>Byte</th><th>Bit</th><th colspan="4">Meaning</th></tr>
<tr><td rowspan="10">Byte 1</td><td rowspan="5">b8-5</td><td colspan="4">Restart Requested</td></tr>
<tr><td rowspan="4"></td><td colspan="3">0001: NO RESTART</td></tr>
<tr><td colspan="3">0010: RESTART</td></tr>
<tr><td colspan="3">0011: TRY ANOTHER INTERFACE</td></tr>
<tr><td colspan="3">Other values RFU</td></tr>
<tr><td rowspan="5">b4-1</td><td colspan="4">Alternate Interface Preference</td></tr>
<tr><td rowspan="4"></td><td colspan="3">0001: Contact Chip</td></tr>
<tr><td colspan="3">0010: Mag stripe</td></tr>
<tr><td colspan="3">1111: N/A</td></tr>
<tr><td colspan="3">Other values: RFU</td></tr>
<tr><td rowspan="7">Byte 2</td><td rowspan="7">b8-1</td><td colspan="4">Message Identifier</td></tr>
<tr><td rowspan="6"></td><td colspan="3">00000000: Indicates Kernel default messaging
(when no Message Identifier has been received)</td></tr>
<tr><td colspan="3">Otherwise limited to one of the following Message Identifier values from Table A.20b:</td></tr>
<tr><td rowspan="4"></td><td>'21'</td><td>TRY AGAIN</td></tr>
<tr><td>'18'</td><td>TRY ANOTHER INTERFACE</td></tr>
<tr><td>'20'</td><td>SEE PHONE</td></tr>
<tr><td>Other values RFU</td><td></td></tr>
<tr><td>Byte 3-5</td><td></td><td colspan="4">RFU</td></tr>
</table>

A.1.114 RRP Counter

Tag: —
Length: 1
Format: b
Update: K
Description: Represents the number of retry attempts to send the EXCHANGE RELAY RESISTANCE DATA command to the Card within one transaction.

A.1.115 Security Capability

Tag: '9F820A'
Length: 1
Format: b
Update: K
Description: Indicates the security capability of the Kernel.

Table A.27—Security Capability

Byte	Bit	Meaning
Byte 1	b8-7	Each bit RFU
	b6	Card capture
	b5	Local authentication
	b4-1	Each bit RFU

A.1.116 Service Code

Tag: '5F30'
Length: 2
Format: n 3
Update: K/RA
Description: Service code as defined in Track 1 Data and Track 2 Data.

A.1.117 Start Time

Tag:	—
Length:	var.
Format:	b
Update:	K
Description:	Variable used to store the time in microseconds before sending ERRD command.

A.1.118 Tag Mapping List

Tag:	'9F8221'
Length:	var.
Format:	b
Update:	K
Description:	List of tags for which the Kernel will use a mapped tag to populate the Data Record and Discretionary Data. The tags in the Tag Mapping List are coded according to the BER-TLV coding rules in section 4.7.1. The tags in the Tag Mapping List are ordered in pairs of two tags: { Tag1 MappedTag1 Tag2 MappedTag2 … Tag*n* MappedTag*n* }

A.1.119 Tags To Read

Tag:	'9F8203'
Length:	var.
Format:	b
Update:	K/ACT/DET
Description:	List of tags indicating the data the Terminal has requested to be read. The tags in Tags To Read are coded according to the BER-TLV coding rules in section 4.7.1. This data object is present if the Terminal wants any data back from the Card before the Data Record. This could be in the context of data storage, or for non data storage usage reasons, for example the PAN. This data object may contain configured data. This data object may be provided several times by the Terminal. Therefore, the values of each of these tags must be accumulated in the Tags To Read Yet buffer.

A.1.120 Tags To Read Yet

Tag:	—
Length:	var.
Format:	b
Update:	K
Description:	List of tags that contains the accumulated Terminal data reading requests received in Tags To Read. The tags in Tags To Read Yet are coded according to the BER-TLV coding rules in section 4.7.1. Requested data objects that are sent to the Terminal are spooled from this buffer. Tags To Read Yet is initiated when the Kernel is started with Tags To Read if present in the ACT Signal. This list can be augmented with Terminal requested data objects provided during Kernel processing in DET Signals. The Kernel sends the requested data objects to the Terminal with the DEK Signal in Data To Send.

A.1.121 Terminal Action Code – Denial

Tag:	'9F820B'
Length:	5
Format:	b
Update:	K
Description:	Specifies the acquirer's conditions that cause the denial of a transaction without attempting to go online.

A.1.122 Terminal Action Code – Online

Tag:	'9F820C'
Length:	5
Format:	b
Update:	K
Description:	Specifies the acquirer's conditions that cause a transaction to be transmitted online on an online capable Terminal.

A.1.123 Terminal Capabilities

Tag: '9F33'

Length: 3

Format: b

Update: K

Description: Indicates the card data input, CVM, and security capabilities of the Terminal and Reader. The CVM capability (Byte 2) is instantiated with values depending on the transaction amount.

Table A.28—Terminal Capabilities

Byte	Bit	Meaning
Byte 1	b8	Manual key entry
	b7	Magnetic stripe
	b6	IC with contacts
	b5-1	Each bit RFU
Byte 2	b8	RFU
	b7	Enciphered PIN for online verification
	b6	Signature
	b5	RFU
	b4	No CVM required
	b3	CDCVM
	b2-1	Each bit RFU
Byte 3	b8-7	Each bit RFU
	b6	Card capture
	b5	Local authentication
	b4-1	Each bit RFU

A.1.124 Terminal Country Code

Tag: '9F1A'

Length: 2

Format: n 3

Update: K

Description: Indicates the country of the Terminal, represented in accordance with [ISO 3166-1].

A.1.125 Terminal Expected Transmission Time For Relay Resistance C-APDU

Tag: '9F8215'

Length: 2

Format: b

Update: K

Description: Represents the time that the Kernel expects to need for transmitting the EXCHANGE RELAY RESISTANCE DATA command to the Card. The Terminal Expected Transmission Time For Relay Resistance C-APDU is expressed in units of hundreds of microseconds.

A.1.126 Terminal Expected Transmission Time For Relay Resistance R-APDU

Tag: '9F8216'

Length: 2

Format: b

Update: K

Description: Represents the time that the Kernel expects that the Card will need for transmitting the EXCHANGE RELAY RESISTANCE DATA R-APDU. The Terminal Expected Transmission Time For Relay Resistance R-APDU is expressed in units of hundreds of microseconds.

A.1.127 Terminal Identification

Tag: '9F1C'

Length: 8

Format: an 8

Update: K

Description: Designates the unique location of the Terminal.

A.1.128 Terminal Relay Resistance Entropy

Tag: —

Length: 4

Format: b

Update: K

Description: Contains a Kernel challenge (unpredictable) to be used in the value field of the EXCHANGE RELAY RESISTANCE DATA command.

A.1.129 Terminal Risk Management Data

Tag: '9F1D'

Length: 8

Format: b

Update: K

Description: Application-specific value used by the Card for risk management purposes.

Table A.29—Terminal Risk Management Data

Byte	Bit	Meaning
Byte 1	b8	RFU
	b7	Enciphered PIN verified online (Contactless)
	b6	Signature (Contactless)
	b5	RFU
	b4	No CVM required (Contactless)
	b3	CDCVM (Contactless)
	b2-1	Each bit RFU
Byte 2	b8	CVM Limit exceeded
	b7	Enciphered PIN verified online (Contact)
	b6	Signature (Contact)
	b5	Enciphered PIN verification performed by ICC (Contact)
	b4	No CVM required (Contact)
	b3	CDCVM (Contact)
	b2	Plaintext PIN verification performed by ICC (Contact)
	b1	RFU
Byte 3	b8-1	Each bit RFU
Byte 4	b8	CDCVM bypass requested
	b7	SCA exempt
	b6-1	Each bit RFU
Byte 5	b8-1	Each bit RFU
Byte 6	b8-1	Each bit RFU
Byte 7	b8-1	Each bit RFU

Byte	Bit	Meaning
Byte 8	b8-1	Each bit RFU

A.1.130 Terminal Type

Tag: '9F35'
Length: 1
Format: n 2
Update: K
Description: Indicates the environment of the Terminal, its communications capability, and its operational control.
The Terminal Type is coded according to [EMV Book 4] section A.1.

A.1.131 Terminal Verification Results

Tag: '95'
Length: 5
Format: b
Update: K
Description: Status of the different functions from the Terminal perspective.

Table A.30—Terminal Verification Results

Byte	Bit	Meaning
Byte 1	b8	Local authentication was not performed
	b7-4	Each bit RFU
	b3	Local authentication failed
	b2-1	Each bit RFU
Byte 2	b8	ICC and terminal have different application versions
	b7	Expired application
	b6	Application not yet effective
	b5	Requested service not allowed for card product
	b4-1	Each bit RFU
Byte 3	b8	Cardholder verification was not successful
	b7-4	Each bit RFU
	b3	Online CVM captured
	b2-1	Each bit RFU

<table>
<tr><th>Byte</th><th>Bit</th><th colspan="2">Meaning</th></tr>
<tr><td rowspan="2">Byte 4</td><td>b8</td><td colspan="2">Transaction exceeds floor limit</td></tr>
<tr><td>b7-1</td><td colspan="2">Each bit RFU</td></tr>
<tr><td rowspan="10">Byte 5</td><td>b8</td><td colspan="2">Kernel 8 processing and TVR format</td></tr>
<tr><td>b7</td><td colspan="2">AID mismatch between card and terminal</td></tr>
<tr><td>b6-5</td><td colspan="2">Each bit RFU</td></tr>
<tr><td>b4</td><td colspan="2">Relay resistance threshold exceeded</td></tr>
<tr><td>b3</td><td colspan="2">Relay resistance time limits exceeded</td></tr>
<tr><td rowspan="5">b2-1</td><td colspan="2">Relay resistance performed</td></tr>
<tr><td rowspan="4"></td><td>00: Relay resistance protocol not supported (not used by this version of the specification)</td></tr>
<tr><td>01: RRP NOT PERFORMED</td></tr>
<tr><td>10: RRP PERFORMED</td></tr>
<tr><td>11: RFU</td></tr>
</table>

A.1.132 Timeout Value

Tag: '9F820F'
Length: 2
Format: b
Update: K
Description: Defines the time in ms before the timer generates a TIMEOUT Signal.

A.1.133 Token Requester ID

Tag: '9F19'
Length: 6
Format: n 11
Update: K/RA
Description: Uniquely identifies the pairing of the Token Requestor with the Token Domain, as defined in [EMV Token].

A.1.134 Track 1 Discretionary Data

Tag: '9F1F'

Length: var. up to 54

Format: ans

Update: K/RA

Description: Discretionary part of track 1 according to [ISO/IEC 7813].

A.1.135 Track 2 Discretionary Data

Tag: '9F20'

Length: var. up to 16

Format: cn

Update: K/RA

Description: Discretionary part of track 2 according to [ISO/IEC 7813].

A.1.136 Track 2 Equivalent Data

Tag: '57'

Length: var. up to 19

Format: b

Update: K/RA

Description: Contains the data objects of the track 2, in accordance with [ISO/IEC 7813], excluding start sentinel, end sentinel, and LRC.

Track 2 Equivalent Data has a maximum length of 37 positions and is present in the file read using the READ RECORD command. It is made up of the following sub-fields:

Table A.31—Track 2 Equivalent Data

Data Field	Length	Format
Primary Account Number	var. up to 19 nibbles	n
Field Separator	1 nibble	b ('D')
Expiration Date (YYMM)	2	n (YYMM)
Service Code	3 nibbles	n
Discretionary Data	var.	n
Padded with 'F' if needed to ensure whole bytes	1 nibble	b

A.1.137 Transaction Currency Code

Tag:	'5F2A'
Length:	2
Format:	n 3
Update:	K/ACT
Description:	Indicates the currency code of the transaction, in accordance with [ISO 4217].

A.1.138 Transaction Currency Exponent

Tag:	'5F36'
Length:	1
Format:	n 1
Update:	K/ACT
Description:	Indicates the implied position of the decimal point from the right of the transaction amount represented, in accordance with [ISO 4217].

A.1.139 Transaction Date

Tag:	'9A'
Length:	3
Format:	n 6 (YYMMDD)
Update:	K/ACT/DET
Description:	Local date that the transaction was performed. Transaction Date is processed as per the requirements outlined in [EMV Book 4] section 6.7.3.

A.1.140 Transaction Time

Tag:	'9F21'
Length:	3
Format:	n 6 (HHMMSS)
Update:	K/ACT/DET
Description:	Local time at which the transaction was performed.

A.1.141 Transaction Type

Tag:	'9C'
Length:	1
Format:	n 2
Update:	K/ACT
Description:	Indicates the type of financial transaction, represented by the first two digits of [ISO 8583:1987] Processing Code.

A.1.142 Unpredictable Number

Tag:	'9F37'
Length:	4
Format:	b
Update:	K
Description:	Contains a Kernel challenge (unpredictable) to be used by the Card to ensure the variability and uniqueness to the generation of a cryptogram.

A.1.143 User Interface Request Data 1

Tag:	'9F8205'
Length:	13
Format:	b
Update:	K
Description:	The TLV Database of the Kernel includes two UIRDs: User Interface Request Data 1 and User Interface Request Data 2. A UIRD combines all user interface request parameters to be sent with the OUT Signal or MSG Signal. User Interface Request Data 1 is included in the OUT Signal if at least one of the flags 'UI Request on Outcome Present' or 'UI Request on Restart Present' is set. If both flags are set, then User Interface Request Data 1 includes the user interface request parameters to be acted upon as the outcome is processed. User Interface Request Data 2 is only included in the OUT Signal if both flags 'UI Request on Outcome Present' and 'UI Request on Restart Present' in Outcome Parameter Set are set. In this case, User Interface Request Data 2 includes the user interface request parameters to be acted upon at the restart of the transaction.

Table A.32—UIRD

Data Field	Length	Format
Message Identifier	1	b (see Table A.20b)
Status	1	b (see Table A.34)
Hold Time	3	n 6
Language Preference	8	an (padded with hexadecimal zeroes if length of tag '5F2D' is less than 8 bytes)

Table A.33—(obsolete)

Table A.34—UIRD – Status

Byte	Bit	Meaning	
Byte 1	b8-1	Status	
			00000000: NOT READY
			00000001: IDLE
			00000010: READY TO READ
			00000011: PROCESSING
			00000100: CARD READ SUCCESSFULLY
			00000101: PROCESSING ERROR
			11111111: N/A
			Other values: RFU

Table A.35—(obsolete)

A.1.144 User Interface Request Data 2

Tag:	'9F8219'
Length:	13
Format:	b
Update:	K
Description:	Refer to description of User Interface Request Data 1.

A.1.145 Write Data Status

Tag:	'9F821B'
Length:	1
Format:	b
Update:	K
Description:	Information reported by the Kernel to the Terminal, about the processing of WRITE DATA commands. Possible values are 'completed' or 'not completed'. In the latter case, this status is not specific about which of the WRITE DATA commands failed, or about how many of these commands have failed or succeeded. This data object is part of the Discretionary Data provided by the Kernel to the Terminal.

Table A.36—Write Data Status

Byte	Bit	Meaning
Byte 1	b8	Completed
	b7-1	Each bit RFU

A.2 Data Objects by Tag

Table A.37 lists all data objects known by the Kernel, ordered by tag number.

Table A.37—Data Objects by Tag

Tag	Data Object
'42'	Issuer Identification Number
'50'	Application Label
'57'	Track 2 Equivalent Data
'5A'	Application PAN
'5F24'	Application Expiration Date
'5F2A'	Transaction Currency Code
'5F2D'	Language Preference
'5F30'	Service Code
'5F34'	Application PAN Sequence Number
'5F36'	Transaction Currency Exponent
'5F57'	Account Type
'82'	Application Interchange Profile
'84'	DF Name
'87'	Application Priority Indicator
'8C'	CDOL1
'8F'	Certification Authority Public Key Index (Card)
'90'	Issuer Public Key Certificate
'92'	Issuer RSA Public Key Remainder
'94'	Application File Locator
'95'	Terminal Verification Results
'9A'	Transaction Date
'9C'	Transaction Type
'9E'	Kernel Key Data
'9F01'	Acquirer Identifier
'9F02'	Amount, Authorised (Numeric)
'9F03'	Amount, Other (Numeric)
'9F06'	Application Identifier (Configuration Data)
'9F07'	Application Usage Control
'9F09'	Application Version Number (Reader)

Tag	Data Object
'9F0A'	Application Selection Registered Proprietary Data
'9F0C'	Issuer Identification Number Extended
'9F10'	Issuer Application Data
'9F11'	Issuer Code Table Index
'9F12'	Application Preferred Name
'9F15'	Merchant Category Code
'9F16'	Merchant Identifier
'9F19'	Token Requester ID
'9F1A'	Terminal Country Code
'9F1C'	Terminal Identification
'9F1D'	Terminal Risk Management Data
'9F1E'	Interface Device Serial Number
'9F1F'	Track 1 Discretionary Data
'9F20'	Track 2 Discretionary Data
'9F21'	Transaction Time
'9F24'	Payment Account Reference
'9F26'	Application Cryptogram
'9F27'	Cryptogram Information Data
'9F2B'	Kernel Qualifier
'9F2C'	Card Qualifier
'9F32'	Issuer RSA Public Key Exponent
'9F33'	Terminal Capabilities
'9F34'	CVM Results
'9F35'	Terminal Type
'9F36'	Application Transaction Counter
'9F37'	Unpredictable Number
'9F38'	PDOL
'9F40'	Additional Terminal Capabilities
'9F42'	Application Currency Code
'9F44'	Application Currency Exponent
'9F46'	ICC Public Key Certificate
'9F47'	ICC RSA Public Key Exponent
'9F48'	ICC RSA Public Key Remainder

Tag	Data Object
'9F4D'	Log Entry
'9F4E'	Merchant Name and Location
'9F8102'	Cardholder Verification Decision
'9F8103'	Card Key Data
'9F8104'	Card TVR
'9F8105'	Enhanced Data Authentication MAC
'9F8106'	Authenticated Application Data
'9F8107'	IAD MAC Offset
'9F8108'	Restart Indicator
'9F8109'	Issuer Application Data MAC
'9F810A'	Extended SDA Tag List
'9F810B'	ICC ECC Public Key
'9F810C'	Relay Resistance Time Excess
'9F810D'	Card Capabilities Information
'9F8111' – '9F811A' [9]	Data Envelope 1 – 10
'9F8201' [9]	Data Needed
'9F8202' [9]	Proceed To First Write Flag
'9F8203' [9]	Tags To Read
'9F8204'	Error Indication
'9F8205'	User Interface Request Data 1
'9F8206'	Card Data Input Capability
'9F8207'	CVM Capability – CVM Required
'9F8208'	CVM Capability – No CVM Required
'9F8209'	Kernel Configuration
'9F820A'	Security Capability
'9F820B'	Terminal Action Code – Denial
'9F820C'	Terminal Action Code – Online
'9F820D'	Reader Contactless Floor Limit
'9F820E'	Reader CVM Required Limit
'9F820F' [9]	Timeout Value
'9F8210'	Outcome Parameter Set
'9F8211'	Message Hold Time

[9] Only implemented for the DE/DS Implementation Option

Tag	Data Object
'9F8212'	Hold Time Value
'9F8213'	Minimum Relay Resistance Grace Period
'9F8214'	Maximum Relay Resistance Grace Period
'9F8215'	Terminal Expected Transmission Time For Relay Resistance C-APDU
'9F8216'	Terminal Expected Transmission Time For Relay Resistance R-APDU
'9F8217'	Relay Resistance Accuracy Threshold
'9F8218'	Relay Resistance Transmission Time Mismatch Threshold
'9F8219'	User Interface Request Data 2
'9F821A'	Kernel Reserved TVR Mask
'9F821B' [10]	Write Data Status
'9F821C' [10]	Read Data Status
'9F821D'	Message Identifiers On Restart
'9F821E'	Default IAD MAC Offset
'9F821F'	Discretionary Data Tag List
'9F8220'	Default CDOL1
'9F8221'	Tag Mapping List
'BF8101' [10]	Data To Send
'BF8102'	Data Record
'BF8103'	Discretionary Data
'BF8104' [10]	Data Envelopes To Write

[10] Only implemented for the DE/DS Implementation Option

A.3 Configuration Data Objects

Table A.38 lists all configuration data objects. When a mandatory configuration data object is not present in Configuration Data, then the transaction is aborted. When an optional configuration data object is not present in Configuration Data, then it will also not be present in the TLV Database (i.e. IsPresent() will return FALSE). Refer to section 4.1.4 for the handling of such data when requested in a DOL.

Note that values of configuration data objects may vary by Application Identifier and Transaction Type.

Configuration data objects listed in the data dictionary with update conditions including ACT/DET may not be present in Configuration Data, but may instead be sent to the Kernel with the ACT Signal at the start of the Kernel or with a DET Signal during transaction processing.

Table A.38—Configuration Data Objects

Data Object	Presence	Example Value
Account Type	O	
Acquirer Identifier	O	
Additional Terminal Capabilities	M	'0000000000'
Application Identifier (Configuration Data)	M	
Application Version Number (Reader)	M	'0002'
Card Data Input Capability	M	'00'
CVM Capability – CVM Required	M	'00'
CVM Capability – No CVM Required	M	'00'
Data Envelopes To Write	O	
Default CDOL1	O	
Default IAD MAC Offset	M	'00'
Discretionary Data Tag List	M	'9F8204'
Hold Time Value	M	'0D'
Interface Device Serial Number	O	
Kernel Configuration	M	'0000'
Kernel Reserved TVR Mask	M	'84000080CF'
Maximum Relay Resistance Grace Period	M	'0032'
Merchant Category Code	O	
Merchant Identifier	O	
Merchant Name and Location	O	

Data Object	Presence	Example Value
Message Hold Time	M	'000013'
Message Identifiers On Restart	M	'211820'
Minimum Relay Resistance Grace Period	M	'0014'
Proceed To First Write Flag	O	
Reader Contactless Floor Limit	M	'000000000000'
Reader CVM Required Limit	M	'000000000000'
Relay Resistance Accuracy Threshold	M	'012C'
Relay Resistance Transmission Time Mismatch Threshold	M	'32'
Security Capability	M	'00'
Tag Mapping List	M	Empty list
Tags To Read	O	
Terminal Action Code – Denial	M	'8400000040'
Terminal Action Code – Online	M	'840084804C'
Terminal Country Code	M	'0000'
Terminal Expected Transmission Time For Relay Resistance C-APDU	M	'0012'
Terminal Expected Transmission Time For Relay Resistance R-APDU	M	'0018'
Terminal Identification	O	
Terminal Risk Management Data (1)	M	'0000000000000000'
Terminal Type	M	'00'
Timeout Value	M	'01F4'
Transaction Currency Code	O	
Transaction Currency Exponent	O	
Transaction Type	M	

(1) Note that bits 3, 4, 6, and 7 in byte 1 and bit 8 in byte 2 in Terminal Risk Management Data are dynamically set by the Kernel based on the CVM characteristics of the transaction.

Annex B (formerly ECC Certificate Format)

The content of this annex has been moved to [EMV Book E].

- For Issuer ECC Certificate Format, see [EMV Book E] section 5.1.
- For ICC ECC Certificate Format, see [EMV Book E] section 5.3.

Annex C (formerly Algorithm Suite Indicators)

The content of this annex has been moved to [EMV Book E] section 8.4.

Annex D (formerly Curve P-256)

The content of this annex has been moved to [EMV Book E] section 8.8.1.

Annex E Abbreviations

Table E.1 lists the abbreviations that are used in this document. For information on terms used in this specification, see section 1.4.

Table E.1—Abbreviations

Abbreviation	Description
AAC	Application Authentication Cryptogram
AC	Application Cryptogram
ADF	Application Definition File
AES	Advanced Encryption Standard
AFL	Application File Locator
AID	Application Identifier
AIP	Application Interchange Profile
an	Alphanumeric characters
ans	Alphanumeric and Special characters
APDU	Application Protocol Data Unit
ARQC	Authorisation Request Cryptogram
ASI	Algorithm Suite Indicator
ATC	Application Transaction Counter
b	Binary
BCD	Binary Coded Decimal
BDH	Blinded Diffie-Hellman
BER	Basic Encoding Rules
C	Conditional
CA	Certification Authority
C-APDU	Command APDU
CDCVM	Consumer Device Cardholder Verification Method
CDOL	Card Risk Management Data Object List
CID	Cryptogram Information Data
CLA	Class byte of command message

Abbreviation	Description
CMAC	Cipher-based Message Authentication Code
CMC	Card Message Counter
cn	Compressed Numeric
CRL	Certification Revocation List
CVM	Cardholder Verification Method
DE	Data Exchange
DE/DS	Data Exchange & Data Storage Implementation Option
DEK	Data Exchange Kernel
DES	Data Encryption Standard
DET	Data Exchange Terminal
DF	Dedicated File
DOL	Data Object List
DS	Data Storage
ECC	Elliptic Curve Cryptography
ECSDSA	Elliptic Curve Schnorr Digital Signature Algorithm
EDA	Enhanced Data Authentication
ERRD	Exchange Relay Resistance Data
FCI	File Control Information
IAD	Issuer Application Data
IAD MAC	Issuer Application Data MAC
ICC	Integrated Circuit Card
ID	Identifier
INS	Instruction byte of command message
ISO	International Organization for Standardization
KMC	Kernel Message Counter
M	Mandatory
MAC	Message Authentication Code
n	Numeric
N/A	Not applicable
N_{CA}	Length of Certification Authority Public Key Modulus

Abbreviation	Description
N_I	Length of Issuer Public Key Modulus
N_{IC}	Length of ICC Public Key Modulus
O	Optional
PAN	Primary Account Number
PDOL	Processing Options Data Object List
PIN	Personal Identification Number
POS	Point of Sale
PPSE	Proximity Payment System Environment
R-APDU	Response APDU
RFU	Reserved for Future Use
RID	Registered Application Provider Identifier
RNG	Random Number Generator
RRP	Relay Resistance Protocol
SC	Secure Channel
SCA	Strong Customer Authentication
SFI	Short File Identifier
SHA	Secure Hash Algorithm
SW	Status Words
TC	Transaction Certificate
TL	Tag Length
TLV	Tag Length Value
TRMD	Terminal Risk Management Data
TVR	Terminal Verification Results
UIRD	User Interface Request Data
UN	Unpredictable Number
var.	Variable