EMV® Specification Bulletin No. 302
First Edition March 2024

Updates to EMV® Contactless Book C-8 – Kernel 8 Specification

Applicability

This Specification Bulletin applies to:

- *EMV Contactless Specifications for Payment Systems, Book C-8, Kernel 8 Specification, version 1.1, June 2023*

Related Documents

- *None*

Effective Date

- *Immediately*

Description

This specification bulletin contains clarifications and corrections to Book C-8, Kernel 8 Specification.

Details of Changes

This specification bulletin discusses changes to the following topics:

Key

Yellow highlights existing text with areas of addition or change.

Blue highlights existing text to be deleted.

Red text indicates final text; that is, text that has been changed or inserted.

A. Discretionary Data

In section A.1.52, Discretionary Data:

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

In section A.1.55, Error Indication:

Change:	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.
To:	Description:	Contains information regarding the nature of the error that has been encountered during the transaction processing.

B. DOL Handling

In section 4.1.4, DOL Handling, change the first bullet as follows:

Change:	• 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.
To:	• If the tag of any data object identified in the DOL is unknown or is not present or is empty or represents a constructed data object, the Kernel concatenates a value of hexadecimal zeroes with the length specified in the DOL entry.

C. Card-sourced Proprietary Tag

In **Table 2-7—Persistent Dataset for Kernel 8**:

Change:

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.

To:

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 range must be defined.

In section 4.1.3, Services, update **Boolean ParseAndStoreCardResponse(TLV String)** as follows:

Change:

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

To:

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) AND update conditions of T include RA Signal]
THEN
IF [(IsNotPresent(T) OR IsEmpty(T) OR (V = GetValue(T))) AND (L is within the range specified by Length field of the data object with tag T in the data dictionary OR L is within the range specified by Length field of the data object with tag T in the Proprietary Tags)]
THEN
Build L' as length field for tag T with the minimum number of bytes
Store L'V in the TLV Database for tag T
ELSE
Return FALSE
ENDIF
ENDIF
}
Return TRUE

D. Tag Mapping

In **Table A.38—Configuration Data Objects**:

Change:

Data Object	Presence	Example Value
Tag Mapping List	M	Empty list

To:

Data Object	Presence	Example Value
Tag Mapping List	O	

In section 4.3, List Handing, update **CreateDataRecord ()** as follows:

Change:

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
}
```

To:

CreateDataRecord ()

```
Initialise(Data Record)
FOR every T in the first column of Table A.11
{
    IF      [IsPresent(T)]
    THEN
            TLV := GetTLV(T)
            IF [IsPresent(TagOf(Tag Mapping List))]
            THEN
                    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
            ELSE
                    AddToList(TLV, Data Record)
            ENDIF
    ENDIF
}
```

In section 4.3, List Handling, update **CreateDiscretionaryData ()** as follows:

Change:

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
}
```

To:

CreateDiscretionaryData ()

```
Initialise(Discretionary Data)
AddToList(GetTLV(TagOf(Error Indication)), Discretionary Data)
IF [IsPresent(TagOf(Discretionary Data Tag List))]
THEN
    FOR every T in Discretionary Data Tag List
    {
            IF      [IsPresent(T)]
            THEN
                    TLV := GetTLV(T)
                    IF [IsPresent(TagOf(Tag Mapping List))]
                    THEN
                            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
                    ELSE
                            AddToList(TLV, Discretionary Data)
                    ENDIF
            ENDIF
    }
ENDIF
```

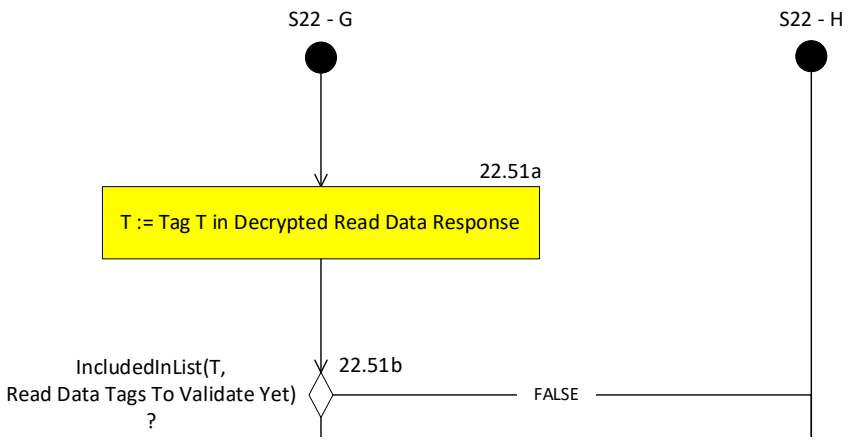
E. Handling Errors in Tag Lengths

Change the final paragraph of section 4.7.1 as follows:

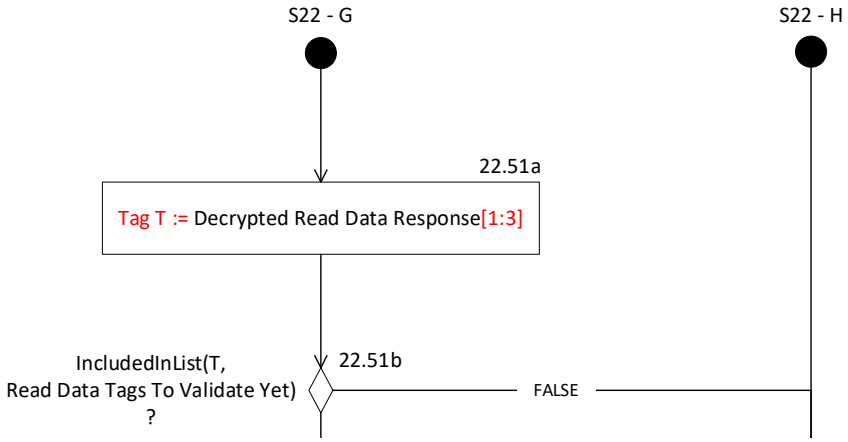
Change:	A TLV coded data object returned by the Card with a tag field with length greater than 3 (i.e. a tag with Byte 3, b8 = 1b) is considered by the Kernel as a format error and processed as described in section 4.7.2.
To:	A TLV coded data object returned by the Card with a tag field with length greater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

Change the final page of **Figure 6.8—State 22 Flow Diagram** as shown:

Change:

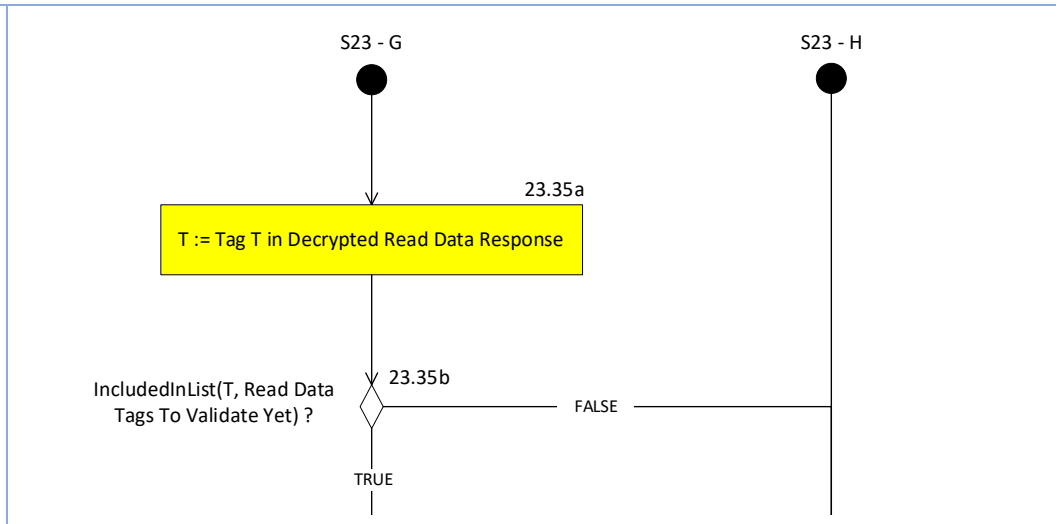


To:

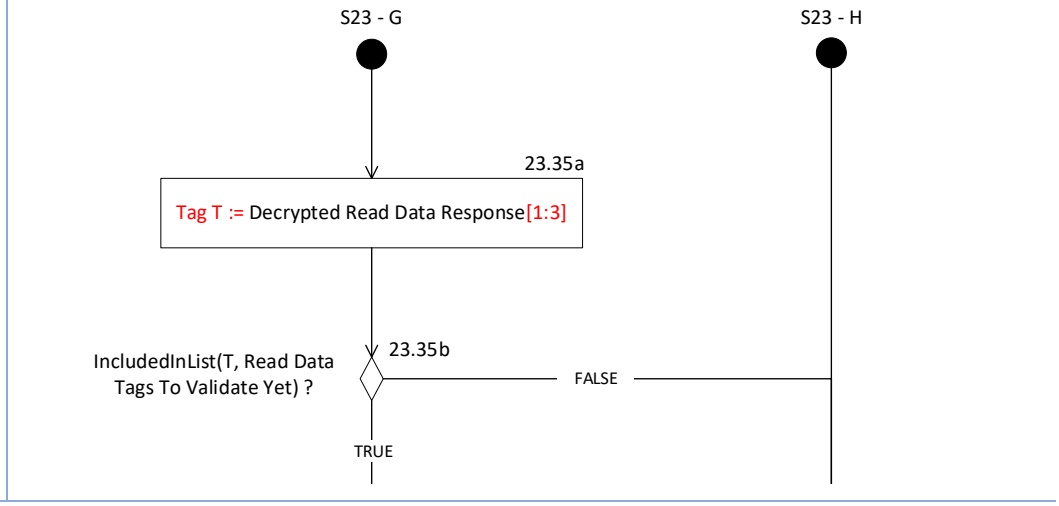


Change the final page of **Figure 6.9—State 23 Flow Diagram** as shown:

Change:

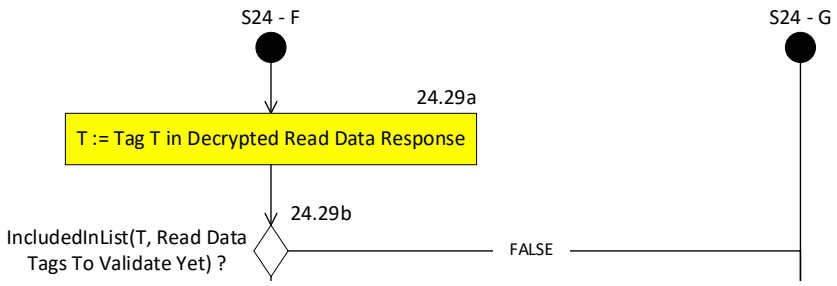


To:

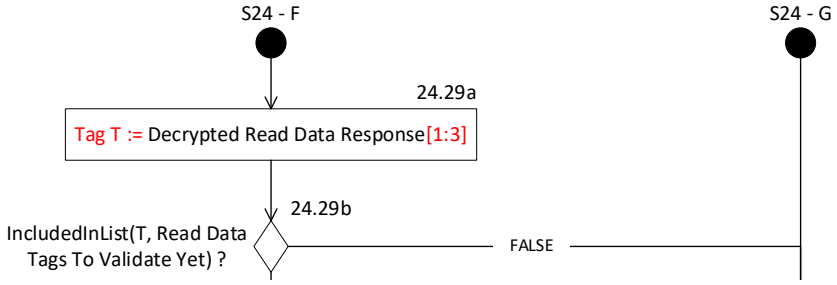


Change the final page of **Figure 6.11—State 24 Flow Diagram** as shown:

Change:

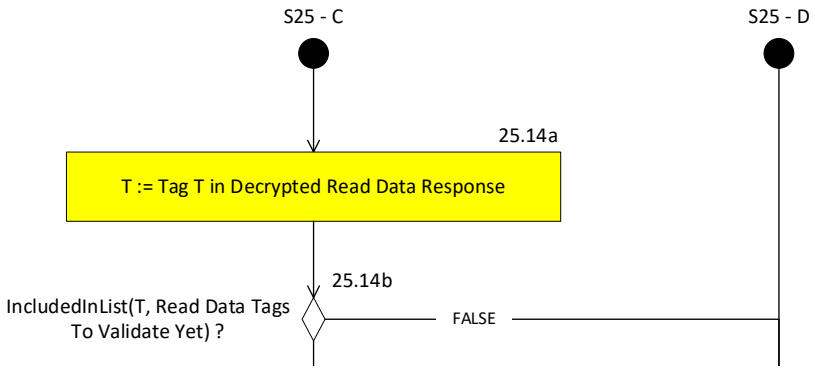


To:

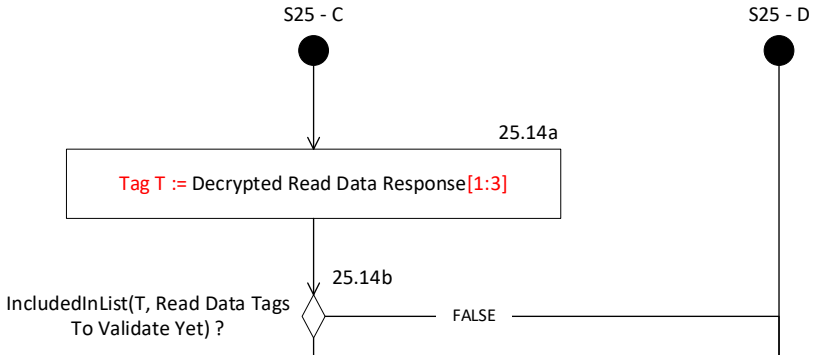


Change the final page of **Figure 6.12—State 25 Flow Diagram** as shown:

Change:



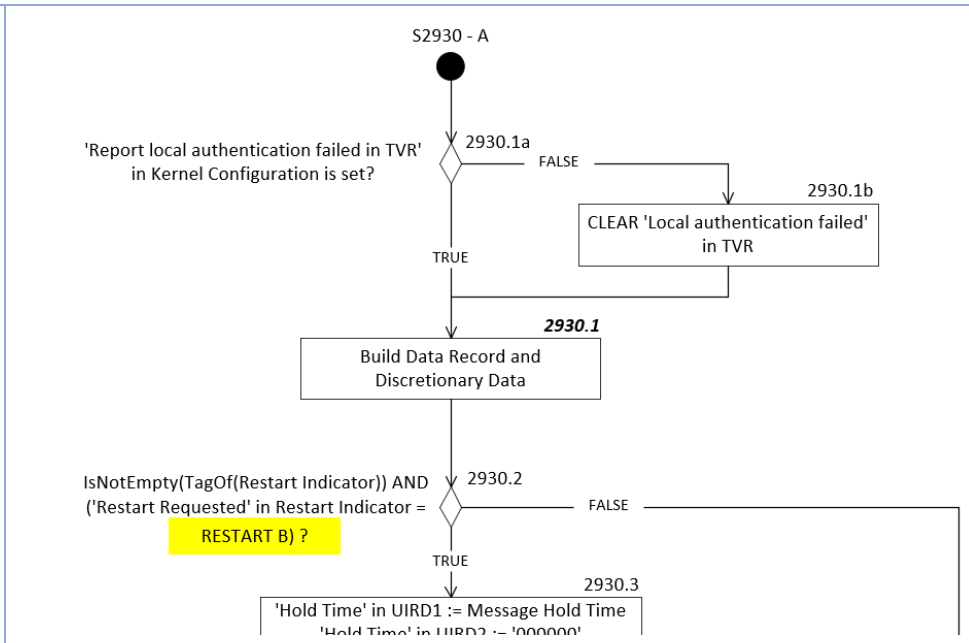
To:



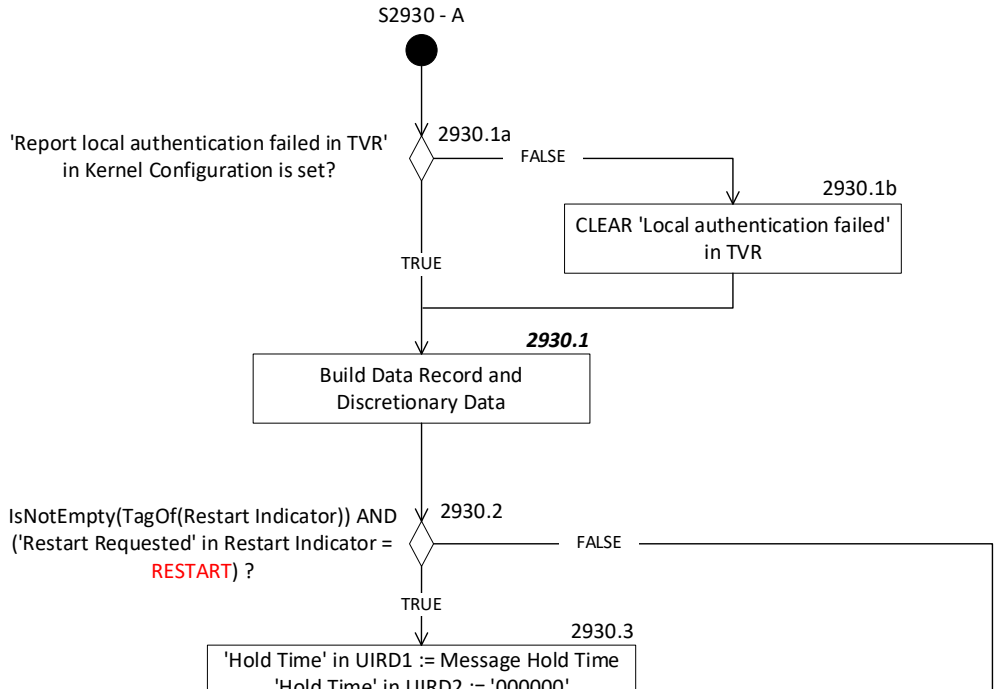
F. Incorrect RESTART Indication

On the first page of **Figure 6.20—States 29 and 30 – Common Processing – Flow Diagram**, at item 2930.2, change ‘RESTART B’ to ‘RESTART’, as shown:

Change:



To:



G. Minor Typos/Corrections

Application Currency Code, Application Currency Exponent, and Log Entry were included in the Data Dictionary but are not used in Book C-8.

***Replace** sections A.1.9, A.1.10, and A.1.84 with:*	**A.1.9 [deleted]** **A.1.10 [deleted]** **A.1.84 [deleted]**

In Table A.37—Data Objects by Tag, ***delete** the following rows:*	'9F42' Application Currency Code '9F44' Application Currency Exponent '9F4D' Log Entry

Token Requester was misspelt twice.

In the Revision log:

Change:	Token Requestor ID added to the data dictionary.
To:	Token Requester ID added to the data dictionary.

In section A.1.133, Token Requester ID:

Change:	Description:	Uniquely identifies the pairing of the Token Requestor with the Token Domain, as defined in [EMV Token].
To:	Description:	Uniquely identifies the pairing of the Token Requester with the Token Domain, as defined in [EMV Token].

Three references to EMV Books 3 and 4 listed slightly wrong section numbers.

In section A.1.39, CVM Results:

Change:	The CVM Results are coded as specified in [EMV Book 4] section A.4.
To:	The CVM Results are coded as specified in [EMV Book 4] section A4.

In section A.1.71, Issuer Code Table Index:

Change:	The Issuer Code Table Index is coded as specified in [EMV Book 3] section C.4.
To:	The Issuer Code Table Index is coded as specified in [EMV Book 3] section C4.

In section A.1.130, Terminal Type:

Change:	The Terminal Type is coded according to [EMV Book 4] section A.1.
To:	The Terminal Type is coded according to [EMV Book 4] section A1.

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.