Light EMV Kernel
Application Programming Guide

for EMV IC Card Payment Application

Version 2.9

Disclaimer of Liability

Although we have carefully checked the contents of this publication for conformity with the hardware and software described, we cannot guarantee complete conformity since errors cannot be excluded. The information provided in this manual is checked at regular intervals and any corrections that might become necessary are included in the next releases. Any suggestions for improvement are welcome.

Subject to change without prior notice.
Release Date: July 6, 2018

Copyright

The LANDI Logo is registered trademarks of Fujian Landi Commercial Equipment Co.,Ltd. The LANDI Documentation is Copyright © 2018 LANDI. All rights reserved.

No part of these publications may be stored in a retrieval system, transmitted, or reproduced in anyway, including but not limited to photocopy, photographic, magnetic, or other record, without the prior agreement and written permission of LANDI Corporation.

LANDI reserves the right to revise and periodically update these documents and to make modifications to the content of these documents at any time without obligation to notify any party, person or device. These documents are guides and not intended to be all encompassing.

Contents

1. Overview

1.1 Introduction

This document introduces an EMV library which covers various kinds of EMV transaction flow and their complicated logic process. It is easy for an application developer to program a payment application and make the application easy to maintain. It is an easier, faster and flexible EMV library.

1.2 Intended Audience

This document is intended for

Developers: in order to be sure they are developing the right EMV payment application that fulfills requirements provided in this document.
Users: in order to get familiar with the idea of EMV application programming.

1.3 Coding Scheme

BCD (Binary-Coded Decimal)

BCD encoded Amount
e.g. 6 Byte Authorized Amount: 123.45 -- BCD coding as: \x00\x00\x00\x01\x23\x45 (Amount should be right-justified)

BCD encoded Date
e.g. March 10th, 2015 -- BCD coding as: \x20\x15\x03\x10

BCD encoded Time
e.g. 08:10:59 -- BCD coding as: \x08\x10\x59

HEX (Hexadecimal Coding)

HEX encoded AID:
e.g. AID: A000000333010101 -- HEX coding as: \xA0\x00\x00\x03\x33\x01\x01\x01

1.4 Supported Transaction Type

- EMV Contact Level 2 4.3
- VISA PayWave 2.1.3 (qVSDC)/2.2
- MasterCard PayPass 3.0
- PBOC 3.0 Contact Level 2

- PBOC 3.0 Contactless Level 2
- PBOC 3.0 Ecash
- PBOC 3.0 Quick Pay (qPBOC)
- American Express ExpressPay 3.1 (AMEX 3.1)
- Discover® Contactless D-PAS v1.1 & ZIP Payment v3.1.2 (DISCOVER)
- JCB
- RuPay (qSPARC)

1.5 Chapter Guidance

This document helps readers to understand the basic structure of the light EMV kernel to implement the EMV transaction, and the implementation mechanism of the interaction between the API and the application side provided outside, so that the developers can quickly start the application programming of EMV.

This document contains main chapters described as below, and the developer should focus on Chapter 4 & 5:

Chapter 2 Application Module Architecture: introduces the relative relationship between the light kernel and the upper application and the underlying driver.

Chapter 3 Transaction Time Sequence: introduces the implementation flow of EMV transaction and the events generated and the timing of event processing signals.

Chapter 4 Event & Signal: introduces the names of all possible events generated, the corresponding callback functions, and the feedback signals and data of the light kernel in various transactions.

Chapter 5 Transaction Flow Chart: introduces the different events generated by the light kernel in dealing with different transaction processes, and the data that the application side needs to be fed back to the kernel after processing the event.

Chapter 6 API Specification: introduces the use of the API provided by the kernel.

Chapter 7 Structure Definition: introduces the significance of each member of the kernel's output parameter structure.

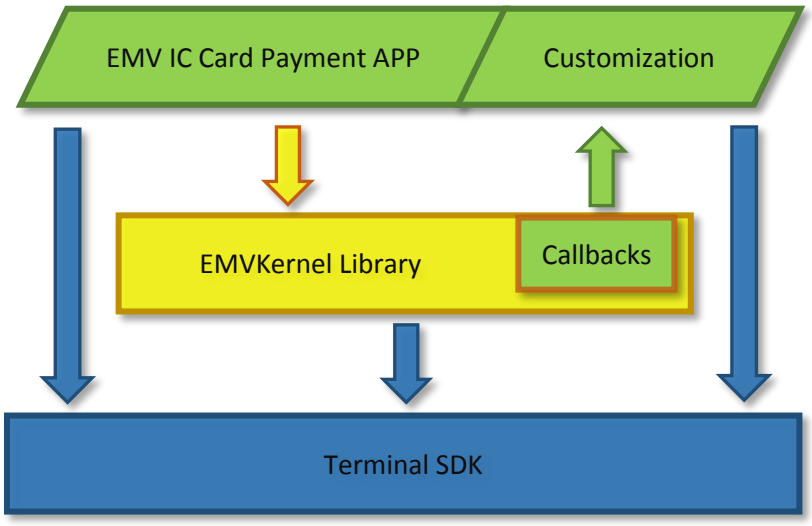
Chapter 8 Obtain Kernel Debug Log: introduces how to get the debug log for the EMV kernel.

Appendix: a reference dictionary including label definition list, macro definition set, transaction return code, answers for common errors.

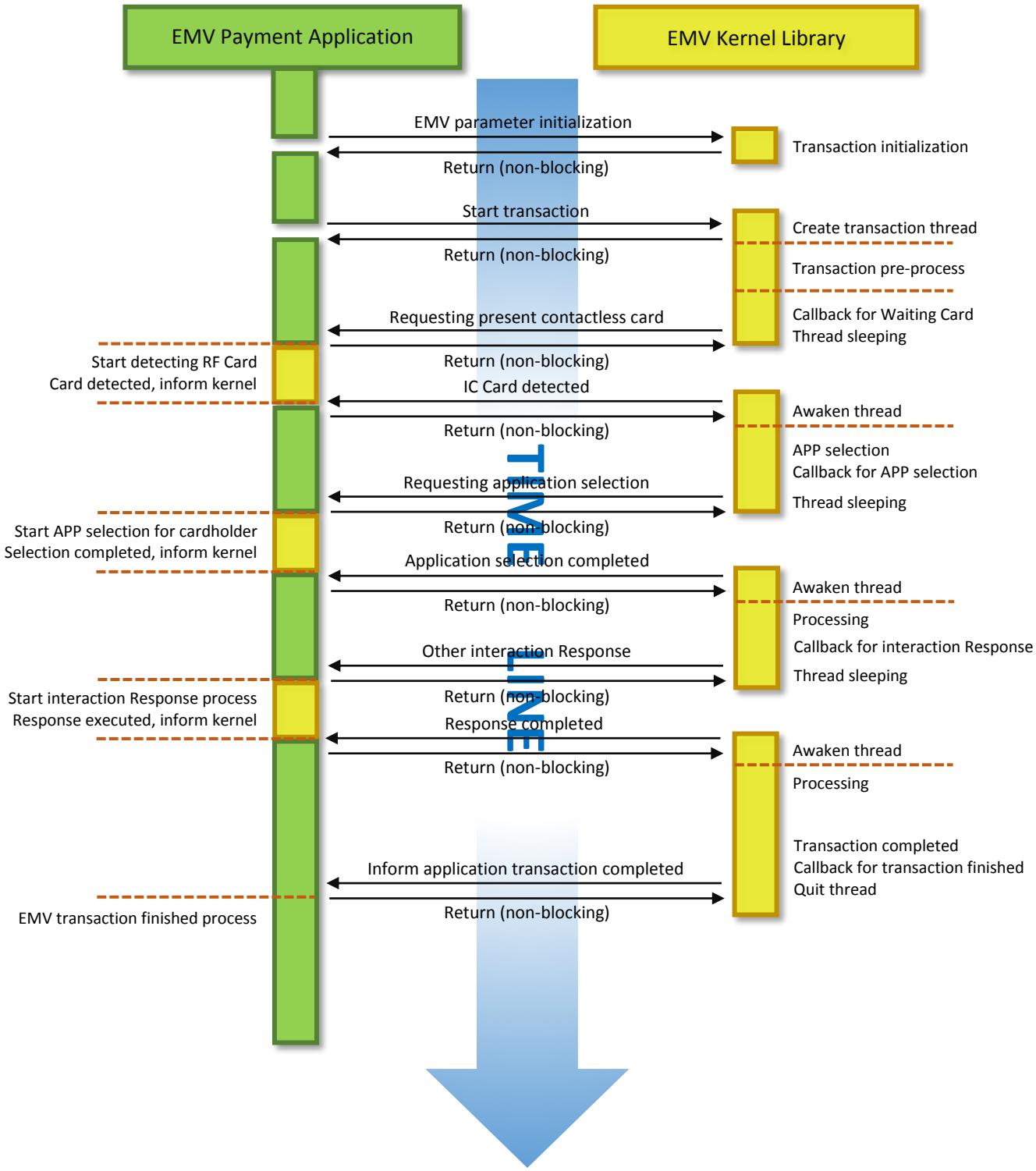
2. Application Module Architecture

The SDK APIs provide chipcard/contactless device I/O and APDU APIs, such as open/close device and send APDU command.

The libEMVKernel is a library based on SDK, which provides APIs to process many kinds of transaction flow, such as EMV, VISA, Master Card, PBOC, qPBOC, AMEX 3.1, DISCOVER.



3. Transaction Time Sequence



4. Event and Signal

The following table lists all of the events, signal and data that the kernel may generate during the EMV transaction.

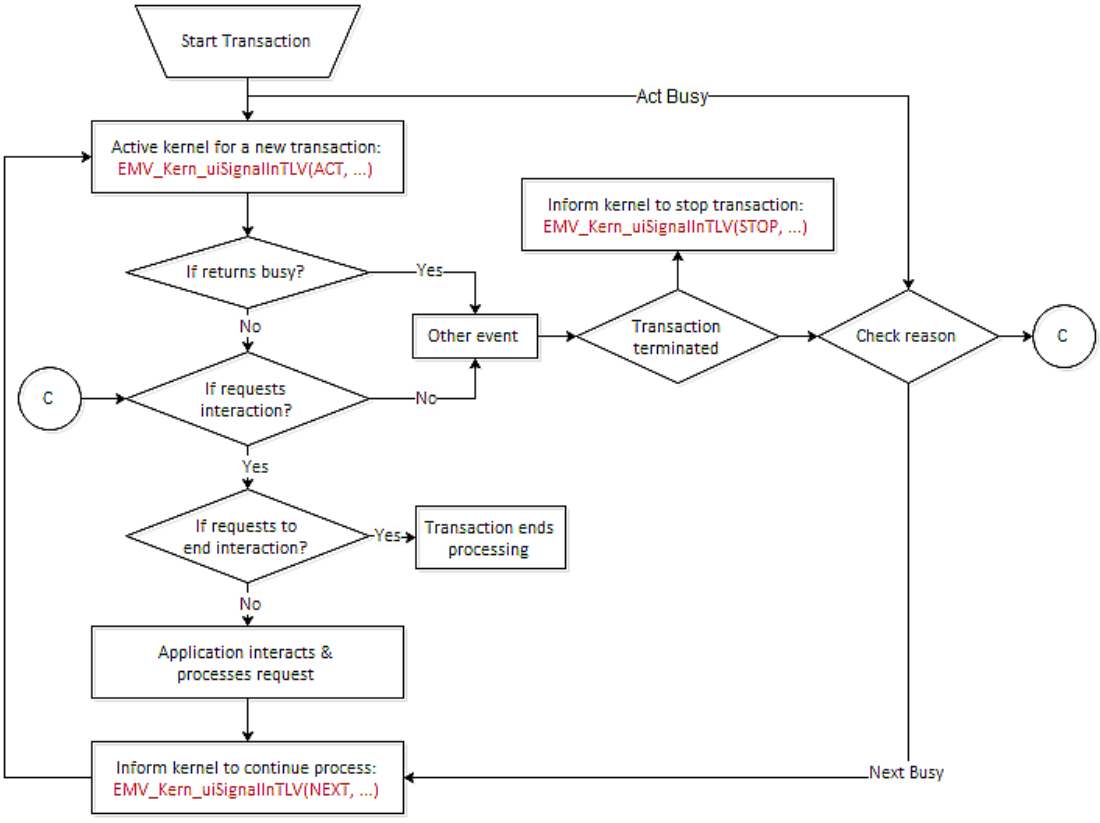
Table 4-1 Event and Signal

Note: M: Mandatory, O: Optional, C: Conditional

Events	Callback Function	Request Description	Wait Signal	Signal Response
		Activate a new EMV transaction Transaction starting Form application selection	EMV_SIGNAL_ACT	DEF_TAG_PSE_FLAG (O) DEF_TAG_START_RECOVERY (O) DEF_TAG_QUERY_ICCLOG (O)
Wait Contactless Card	EXEP_ucWaitCard	Request cardholder to show contactless card	EMV_SIGNAL_NEXT	NULL
Application Selection	EXEP_ucAppSelection	Request cardholder to select IC card application	EMV_SIGNAL_NEXT	EMV_TAG_TM_AID (M)
After Final Selection	EXEP_ucFinalSlt	Supply a time slot for application to set parameters according finial selected AID.	EMV_SIGNAL_NEXT	Please refer to the parameter requirement table in various transaction flows of Chapter 5.
After Read Record	EXEP_ucReadRecord	Supply a time slot for application to process card records and set parameters, such as display card No., find blacklist, set public key, etc.	EMV_SIGNAL_NEXT	DEF_TAG_PAN_IN_BLACK (O) DEF_TAG_ACCUMULATE_AMOUNT (O)
Cardholder Verification	EXEP_ucCardHolderVerify	Request cardholder to made a verification according CVM which indicated by EMV kernel.	EMV_SIGNAL_NEXT	DEF_TAG_CHV_STATUS (M)
Online Authorization	EXEP_ucOnlineProcess	Request the application to perform online authorization	EMV_SIGNAL_NEXT	DEF_TAG_ONLINE_STATUS (M) If online communication successes, the following is necessary while returned by host service. EMV_TAG_TM_ARC (C) DEF_TAG_AUTHORIZE_FLAG (C) EMV_TAG_TM_AUTHCODE (C) DEF_TAG_HOST_TLVDATA (C)
Transaction Finished	EXEP_vEndProcess	Inform the application that transaction finished and kernel exited.	No response	
Kernel Obtain	EXEP_vObtain	Application data are requested from application by EMV kernel, such as torn transaction information	EMV_SIGNAL_NEXT	
Kernel Sendout	EXEP_vSendOut	Send out some messages from EMV kernel to application, such as notifications, torn transaction information, logs.	No response	
		Request EMV kernel to terminate a transaction after transaction activated but haven't finished.	EMV_SIGNAL_STOP	

5. Transaction Flow Chart

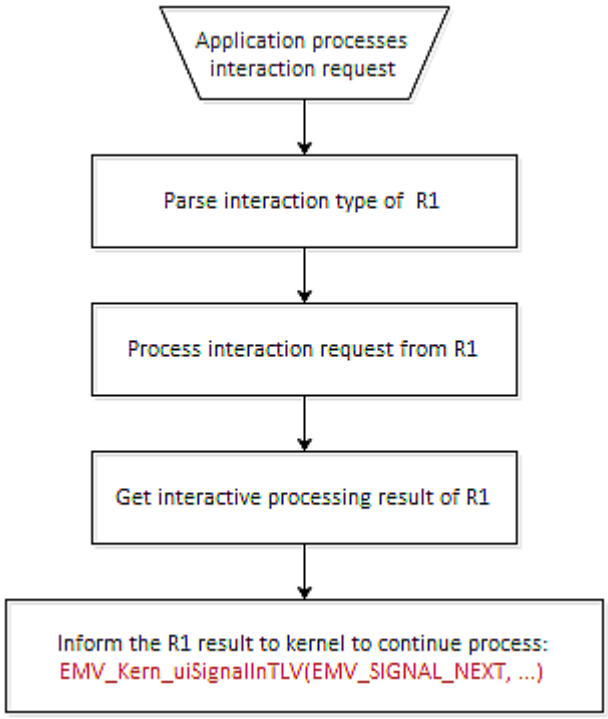
EMV transaction is running based on event and signal mechanism. After activing a transaction, EMV kernel will process and conduct the procedures till transaction is completed. During transaction process, EMV kernel will generate one or multiple events for interactions when necessary. These events have different purposes, including cardholder application selection, cardholder verification or transaction related parameters setting, etc. These events are generated depending on transaction related parameters and IC card logic design, and it is unpredictable. Event is generated based on callback mechanism. After generating events, EMV kernel will go into sleeping mode waiting for the signal to awake after finishing event process by application. After EMV kernel awakens, EMV kernel will proceed with transaction until it issues a notification that the transaction is over.



For example: When EMV kernel generates an event (marked as R1), R1 callback function is implemented as follows:

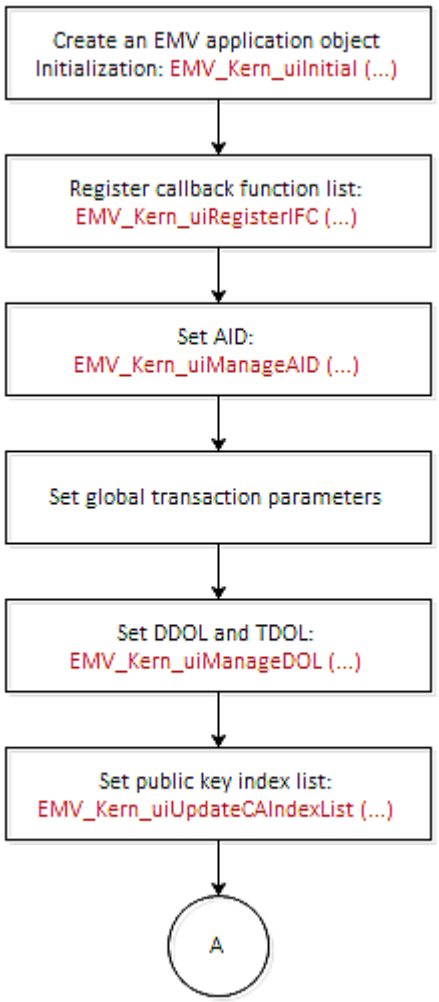
1. Store the parameters which are returned with event R1.
2. Set the event flag as R1.
3. Return success after end processing.

Note: Never call the API "EMV_Kern_uiSignalInTLV" in the implementation of callback function. Please refer to the implementation example: EMV_Kern_uiCreateObject



5.1 Kernel Initialization

Before activating any transaction, EMV kernel initialization is necessary and mandatory. The initialization should be executed once after EMV payment application starting. After kernel initialization, application is allowed to set the parameters which are global and always with same values for difference transaction.



5.2 EMV Contact Level 2

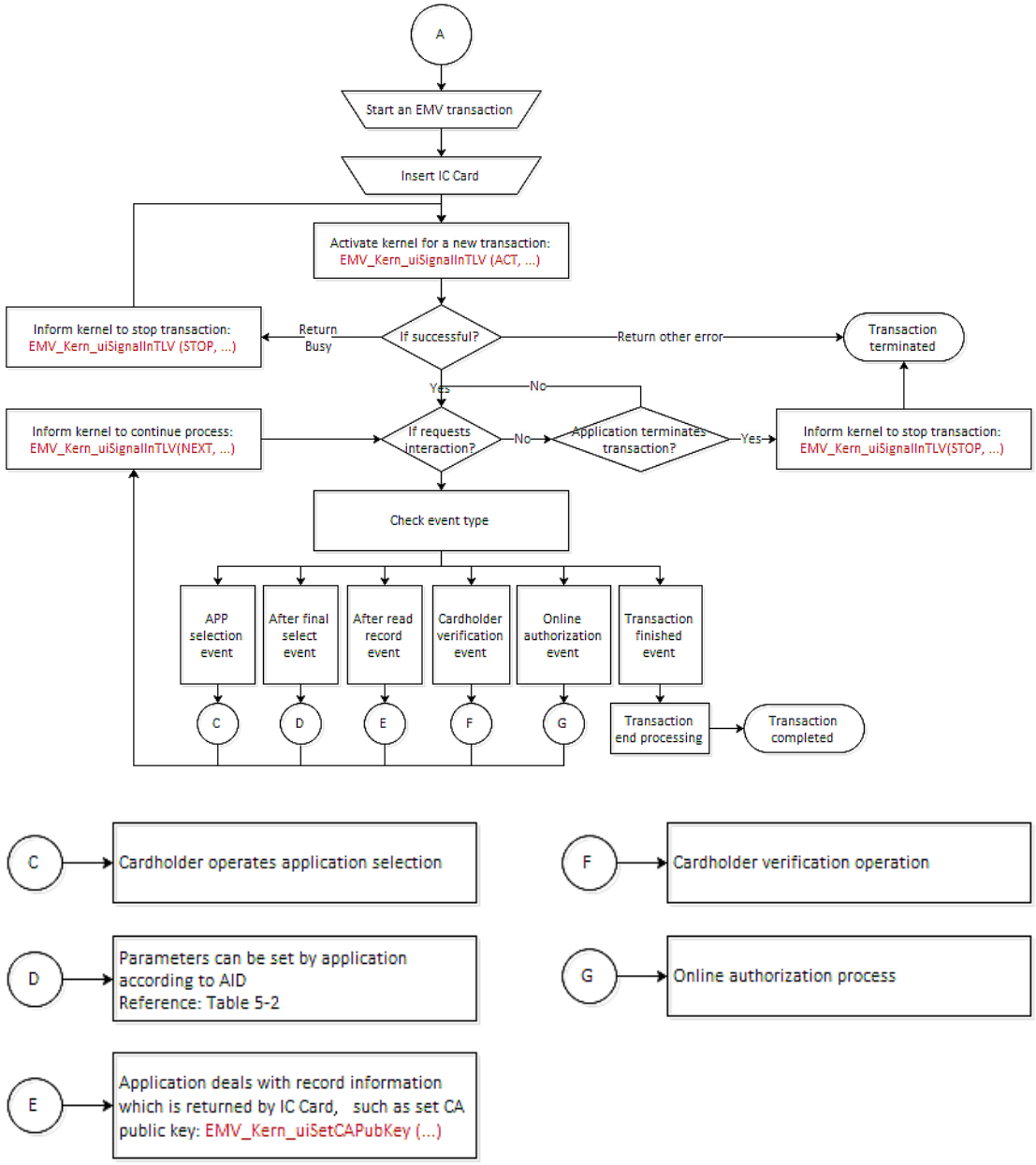
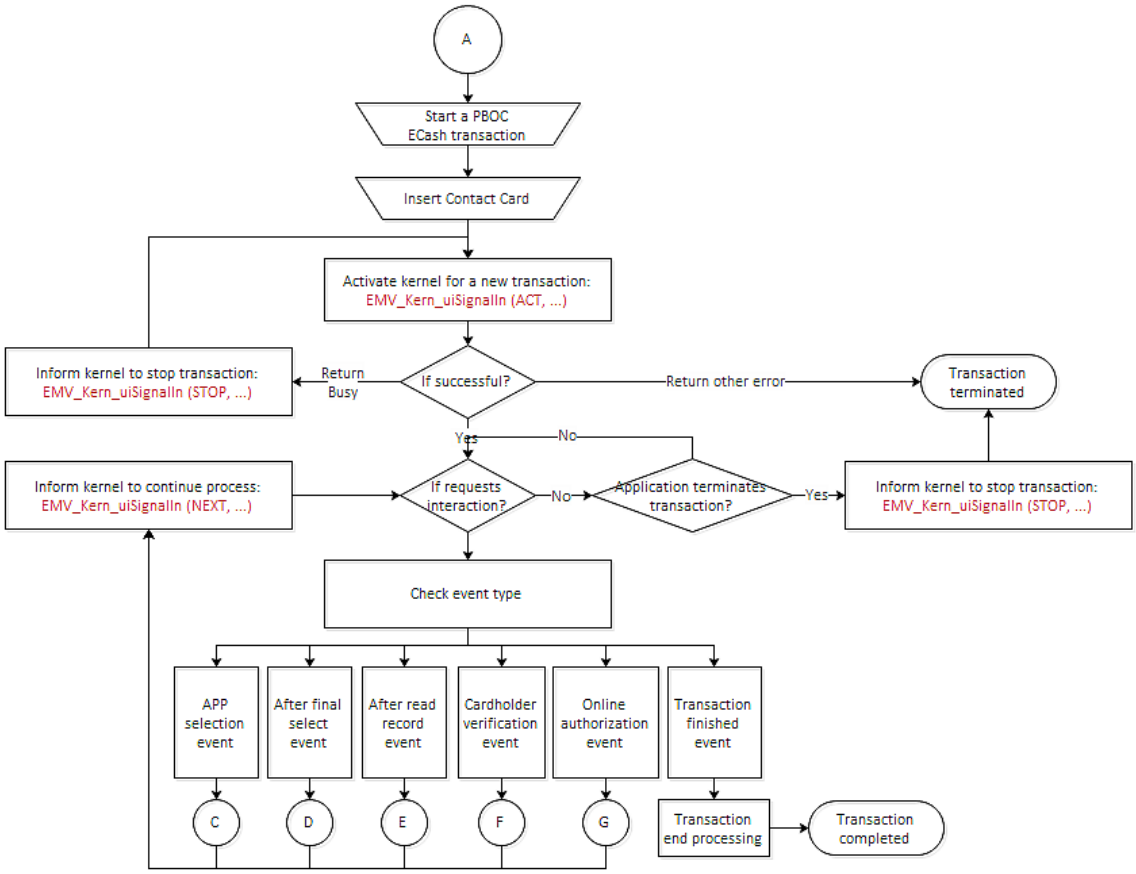


Table 5-2 Transaction Parameter Requirements for contact EMV L2

Note: REF_V: Reference Value, M: Mandatory, O: Optional, C: Conditional

Event Type	**Transaction Parameter Requirement**	
D: After Final Selection	After final application selection, the AID is decided. Some parameters can be set with different values based on different AID. The following parameters are recommended for contact EMV L2 flow:	
	EMV_TAG_TM_AUTHAMNTN (M)	REF_V: 0x000000000100(1.00)
	EMV_TAG_TM_OTHERAMNTN (O)	REF_V:0x000000000000
	EMV_TAG_TM_TRANSDATE (M)	REF_V: 0x171216
	EMV_TAG_TM_TRANSTIME (M)	REF_V: 0x131535
	EMV_TAG_TM_TRSEQCNTR (M)	REF_V: 0x00001234
	EMV_TAG_TM_TERMTYPE (M)	REF_V: 0x22
	EMV_TAG_TM_CAP (M)	REF_V: 0xE0F8C8
	EMV_TAG_TM_CAP_AD (M)	REF_V: 0x6000F0A001
	EMV_TAG_TM_CNTRYCODE (M)	REF_V: 0x0156
	EMV_TAG_TM_CURCODE (M)	REF_V: 0x0156
	EMV_TAG_TM_TRANSTYPE (M)	REF_V: 0x00(Purchase)
	EMV_TAG_TM_FLOORLMT (O)	REF_V: 0x00002710(100.00)
	DEF_TAG_TAC_DECLINE (O)	REF_V: 0x0000000000
	DEF_TAG_TAC_ONLINE (O)	REF_V: 0xFFFFFFFFFF
	DEF_TAG_TAC_DEFAULT (O)	REF_V: 0xFFFFFFFFFF
	DEF_TAG_GAC_CONTROL (O)	REF_V: 0x00
	DEF_TAG_SERVICE_TYPE (M)	REF_V: 0x00(Purchase)
	DEF_TAG_RAND_SLT_THRESHOLD (M)	REF_V:0x000000001000(10)
	DEF_TAG_RAND_SLT_PER (M)	REF_V: 0x30
	DEF_TAG_RAND_SLT_MAXPER (M)	REF_V: 0x90
	C_TAG_TM_DF69 (O)	REF_V: 0x01
	Tag definition refers to Appendix A **Via API:** EMV_Kern_uiSetTLV(ucKernelID = EMV_KERNELID_EMV) EMV_Kern_uiSetTLVList(ucKernelID = EMV_KERNELID_EMV)	

5.3 PBOC Ecash



C → Cardholder operates application selection

E → Application deals with record information which is returned by IC Card, such as set CA public key:
For RSA algorithm: EMV_Kern_uiSetCAPubKey (...)
For SM algorithm : EMV_Kern_uiSetCAPubKey_SM (...)

D → Parameters can be set by application according to AID
Reference: Table 5-3

F → Cardholder verification operation

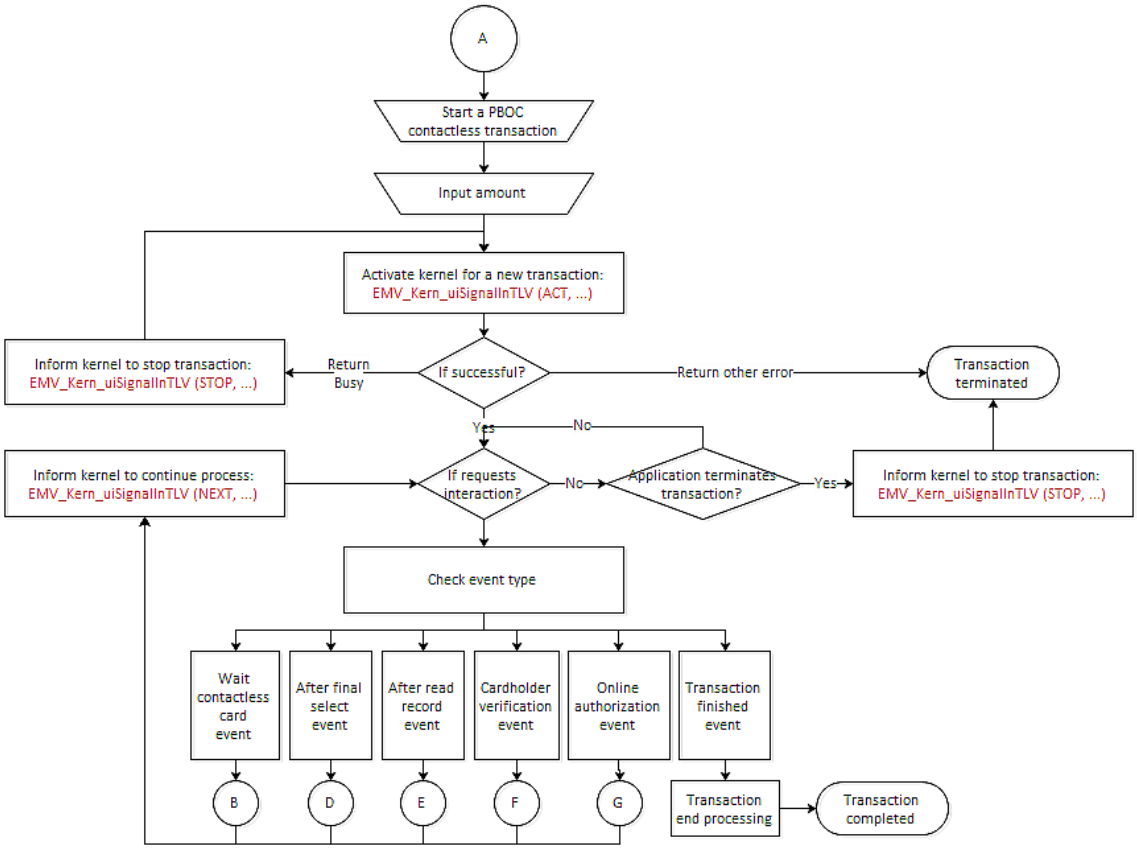
G → Online authorization process: It may require to input online PIN at this callback in Ecash Transaction. Application needs to check CVM type of kernel return, if the value is EMV_CVMFLAG_ECASHPIN(0x21) and no PIN input during the cardholder verification operation, then PIN input is mandatory.

Table 5-3 Transaction Parameter Requirements for PBOC Ecash

Note: REF_V: Reference Value, M: Mandatory, O: Optional, C: Conditional

Event Type	**Transaction Parameter Requirement**	
D: After Final Selection	After final application selection, the AID is decided. Some parameters can be set with different value based on different AID. The following parameters are recommended for PBOC Ecash:	
	EMV_TAG_TM_AUTHAMNTN (M)	REF_V: 0x000000000100(1.00)
	EMV_TAG_TM_OTHERAMNTN (O)	REF_V: 0x000000000000
	EMV_TAG_TM_TRANSDATE (M)	REF_V: 0x171216
	EMV_TAG_TM_TRANSTIME (M)	REF_V: 0x131535
	EMV_TAG_TM_TRSEQCNTR (M)	REF_V: 0x00001234
	EMV_TAG_TM_TERMTYPE (M)	REF_V:0x22
	EMV_TAG_TM_CAP (M)	REF_V: 0xE0F8C8
	EMV_TAG_TM_CAP_AD (M)	REF_V: 0x6000F0A001
	EMV_TAG_TM_CNTRYCODE (M)	REF_V: 0x0156
	EMV_TAG_TM_CURCODE (M)	REF_V: 0x0156
	EMV_TAG_TM_TRANSTYPE (M)	REF_V: 0x00(Purchase)
	EMV_TAG_TM_FLOORLMT (O)	REF_V: 0x00002710(100.00)
	DEF_TAG_TAC_DECLINE (O)	REF_V: 0x0000000000
	DEF_TAG_TAC_ONLINE (O)	REF_V: 0xFFFFFFFFFF
	DEF_TAG_TAC_DEFAULT (O)	REF_V: 0xFFFFFFFFFF
	DEF_TAG_SERVICE_TYPE (M)	REF_V: 0x00(Purchase)
	C_TAG_TM_9F7A (M)	REF_V: 0x01
	C_TAG_TM_DF69 (O)	REF_V: 0x01
	C_TAG_TM_9F7B (M)	REF_V: 0x000000010000(100.00)
	Tag definition refers to Appendix A **Via API:** EMV_Kern_uiSetTLV(ucKernelID = EMV_KERNELID_PBOC) EMV_Kern_uiSetTLVList(ucKernelID =EMV_KERNELID_PBOC)	

5.4 PBOC Contactless Level 2



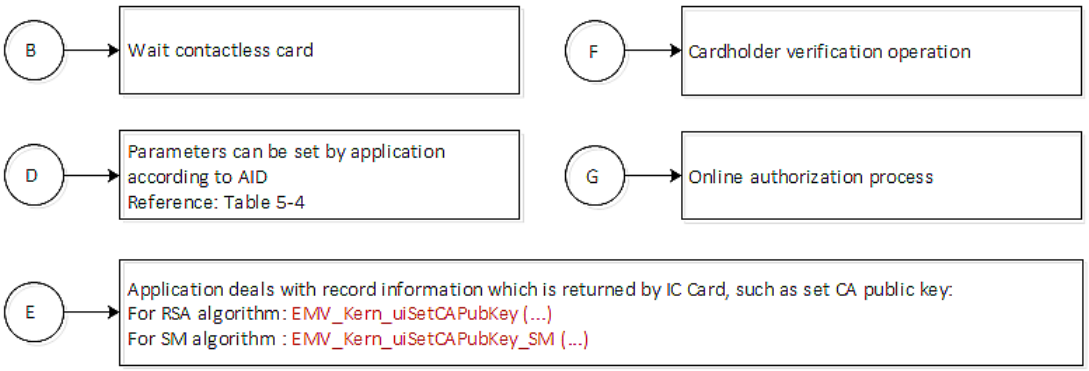


Table 5-4 Transaction Parameter Requirements for PBOC Contactless L2

Note: REF_V: Reference Value, M: Mandatory, O: Optional, C: Conditional

Event Type	**Transaction Parameter Requirement**	
D: After Final Selection	After final application selection, the AID is decided. Some parameters can be set with different values based on different AID. The following parameters are recommended for PBOC contactless flow:	
	EMV_TAG_TM_AUTHAMNTN (M)	REF_V: 0x000000000100(1.00)
	EMV_TAG_TM_OTHERAMNTN (O)	REF_V: 0x000000000000
	EMV_TAG_TM_TRANSDATE (M)	REF_V: 0x171216
	EMV_TAG_TM_TRANSTIME (M)	REF_V: 0x131535
	EMV_TAG_TM_TRSEQCNTR (M)	REF_V: 0x00001234
	EMV_TAG_TM_TERMTYPE (M)	REF_V: 0x22
	EMV_TAG_TM_CAP (M)	REF_V: 0xE0F8C8
	EMV_TAG_TM_CAP_AD (M)	REF_V: 0x6000F0A001
	EMV_TAG_TM_CNTRYCODE (M)	REF_V: 0x0156
	EMV_TAG_TM_CURCODE (M)	REF_V: 0x0156
	EMV_TAG_TM_TRANSTYPE (M)	REF_V: 0x00(Purchase)
	EMV_TAG_TM_FLOORLMT (O)	REF_V: 0x00002710(100.00)
	DEF_TAG_TAC_DECLINE (O)	REF_V: 0x0000000000
	DEF_TAG_TAC_ONLINE (O)	REF_V: 0xFFFFFFFFFF
	DEF_TAG_TAC_DEFAULT (O)	REF_V: 0xFFFFFFFFFF
	DEF_TAG_GAC_CONTROL (O)	REF_V: 0x00
	DEF_TAG_SERVICE_TYPE (M)	REF_V: 0x00(Purchase)
	DEF_TAG_RAND_SLT_THRESHOLD (M)	REF_V: 0x000000001000(10)
	DEF_TAG_RAND_SLT_PER (M)	REF_V: 0x30
	DEF_TAG_RAND_SLT_MAXPER (M)	REF_V: 0x90
	C_TAG_TM_9F66 (M)	REF_V: 0x67004080
	C_TAG_TM_DF69 (O)	REF_V: 0x01
	C_TAG_TM_TRANS_LIMIT (O)	REF_V: 0x000000010000(100.00)
	C_TAG_TM_CVM_LIMIT (O)	REF_V: 0x000000010000(100.00)
	C_TAG_TM_FLOOR_LIMIT (O)	REF_V: 0x000000010000(100.00)
	C_TAG_TM_RD_RCP (O)	REF_V: 0x7C00
	Tag definition refers to Appendix A **Via API:** EMV_Kern_uiSetTLV(ucKernelID = EMV_KERNELID_PBOC) EMV_Kern_uiSetTLVList(ucKernelID = EMV_KERNELID_PBOC)	

5.5 PBOC Quick Pay (qPBOC)

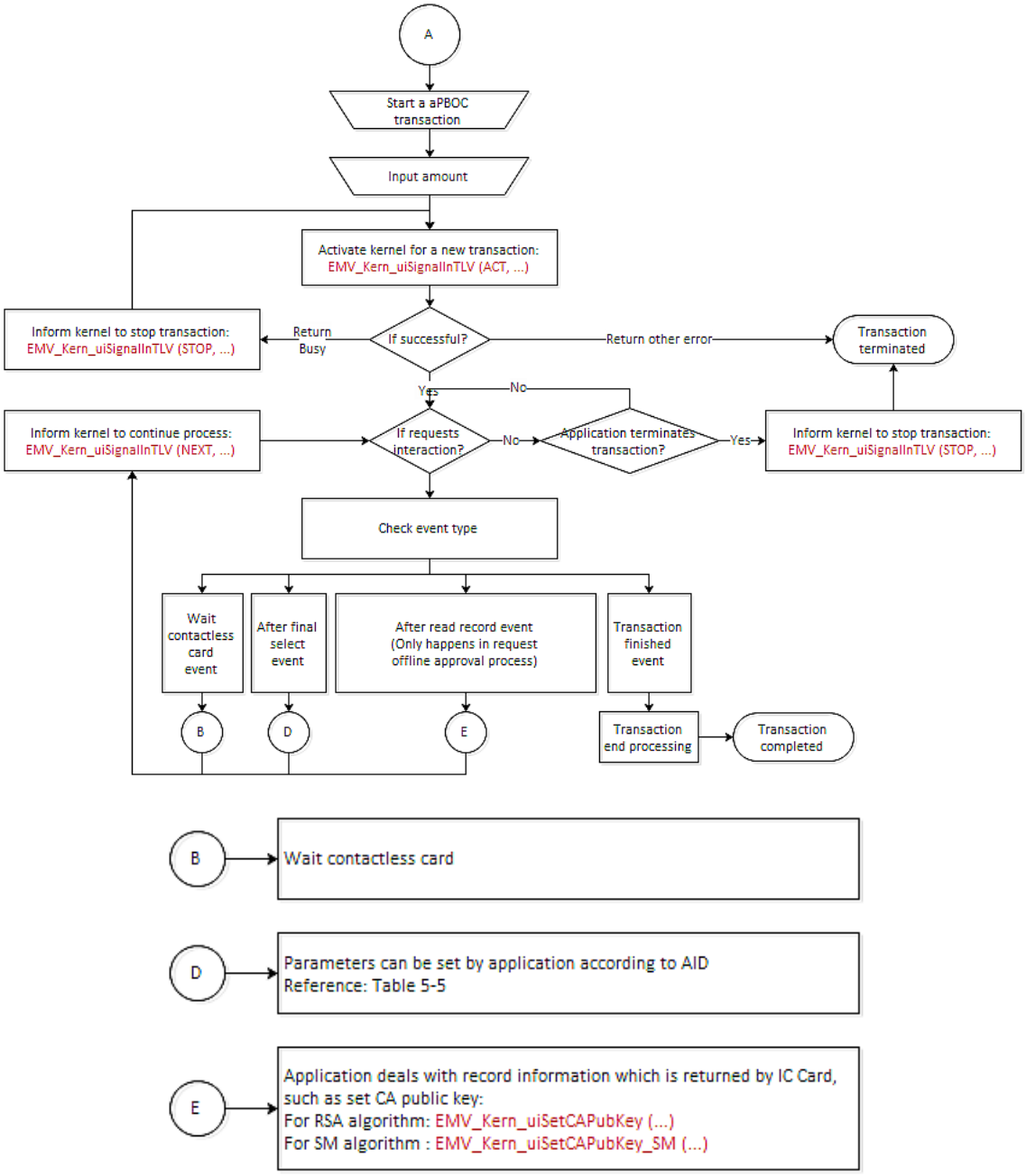


Table 5-5 Transaction Parameter Requirements for qPBOC

Note: REF_V: Reference Value, M: Mandatory, O: Optional, C: Conditional

Event Type	Transaction Parameter Requirement	
D: After Final Selection	After final application selection, the AID is decided. Some parameters can be set with different values based on different AID. The following parameters are recommended for qPBOC flow:	
	EMV_TAG_TM_AUTHAMNTN (M)	REF_V: 0x000000000100(1.00)
	EMV_TAG_TM_OTHERAMNTN (O)	REF_V: 0x000000000000
	EMV_TAG_TM_TRANSDATE (M)	REF_V: 0x171216
	EMV_TAG_TM_TRANSTIME (M)	REF_V: 0x131535
	EMV_TAG_TM_TRSEQCNTR (M)	REF_V: 0x00001234
	EMV_TAG_TM_CURCODE (O)	REF_V: 0x0156
	EMV_TAG_TM_TRANSTYPE (O)	REF_V: 0x00(Purchase)
	C_TAG_TM_9F66 (M)	REF_V: 0x27004080
	C_TAG_TM_DF69 (O)	REF_V: 0x01
	C_TAG_TM_TRANS_LIMIT (O)	REF_V: 0x000000010000(100.00)
	C_TAG_TM_CVM_LIMIT (O)	REF_V: 0x000000010000(100.00)
	C_TAG_TM_FLOOR_LIMIT (O)	REF_V: 0x000000010000(100.00)
	DEF_TAG_TORN_SUPPORT (O)	REF_V: 0x01
	DEF_TAG_ALLOW_DUP_ICC_SAMEVALUE (O)	REF_V: 0x01
	C_TAG_TM_RD_RCP (O)	REF_V: 0x7C00
	Tag definition refers to Appendix A **Via API:** EMV_Kern_uiSetTLV(ucKernelID = EMV_KERNELID_PBOC) EMV_Kern_uiSetTLVList(ucKernelID = EMV_KERNELID_PBOC)	

5.6 VISA (PayWave)

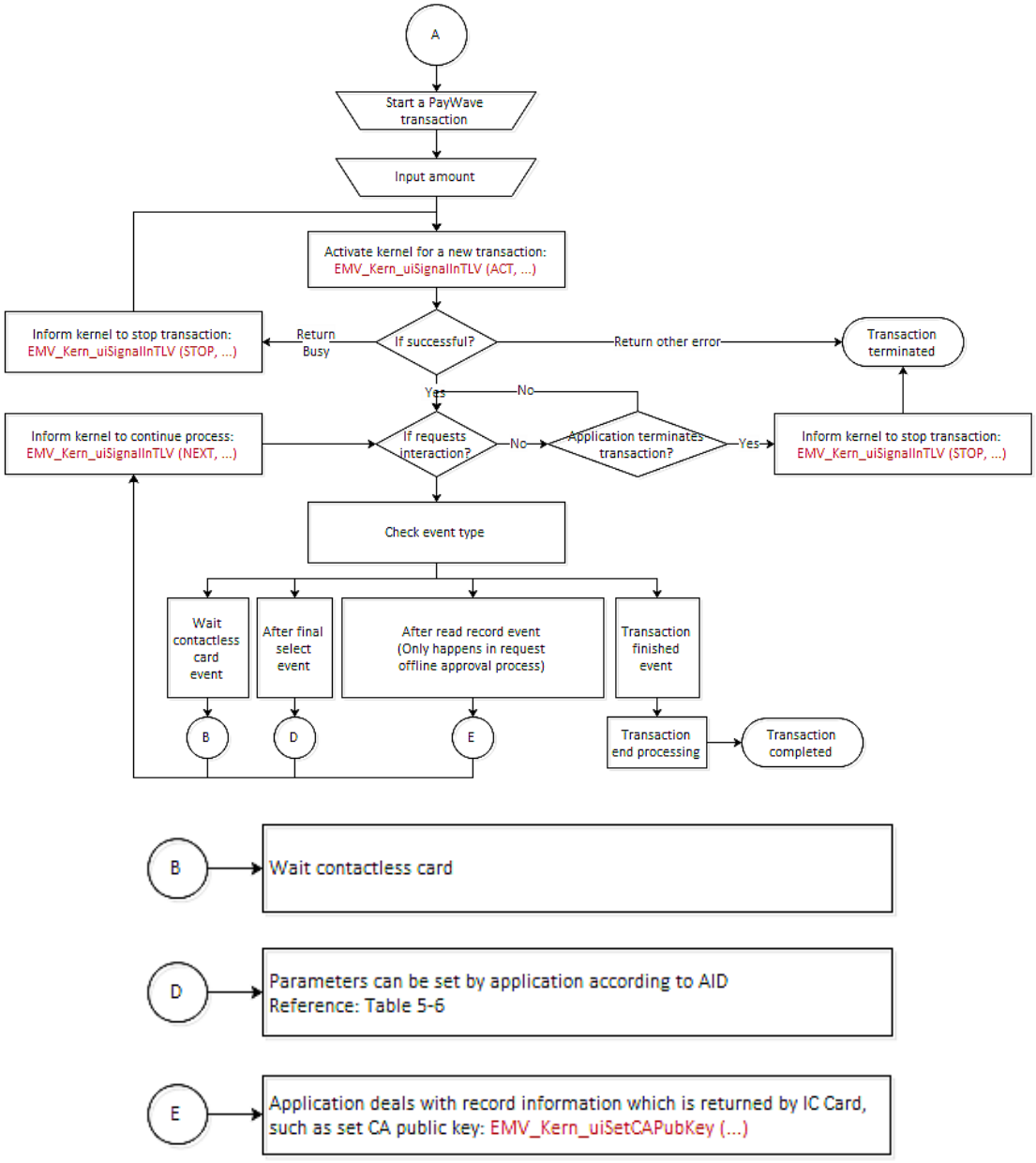


Table 5-6 Transaction Parameter Requirements for PayWave

Note: REF_V: Reference Value, M: Mandatory, O: Optional, C: Conditional

Event Type	Transaction Parameter Requirement	
ACT: Active kernel of a new transaction	DEF_TAG_PSE_FLAG(M)	REF_V: 0x03
	Tag definition refers to Appendix A **Via API:** **EMV_Kern_uiSetTLV**(ucKernelID = EMV_KERNELID_VISA) **EMV_Kern_uiSetTLVList**(ucKernelID = EMV_KERNELID_VISA)	
D: After Final Selection	After final application selection, the AID is decided. All parameters tag are required to set at this step as bellow:	
	EMV_TAG_TM_TRANSTYPE(O)	REF_V: 0x00(Purchase)
	EMV_TAG_TM_AUTHAMNTN(M)	REF_V: 0x000000000100(1.00)
	EMV_TAG_TM_OTHERAMNTN(O)	REF_V: 0x0000000000
	EMV_TAG_TM_TRANSDATE(M)	REF_V: 0x171216
	EMV_TAG_TM_TRANSTIME(M)	REF_V: 0x131535
	EMV_TAG_TM_TRSEQCNTR(O)	REF_V: 0x00001234
	EMV_TAG_TM_TERMTYPE(O)	REF_V: 0x22
	EMV_TAG_TM_CNTRYCODE(M)	REF_V: 0x0840
	EMV_TAG_TM_CURCODE(M)	REF_V: 0840
	V_TAG_TM_9F66(M)	REF_V: 0x26004000
	V_TAG_RD_RCP(O)	REF_V: 0x7C00
	V_TAG_TM_TRANS_LIMIT(O)	REF_V: 0x000000010000(100.00)
	V_TAG_TM_FLOOR_LIMIT(O)	REF_V: 0x000000008000(80.00)
	V_TAG_TM_CVM_LIMIT(O)	REF_V: 0x000000006000(60.00)
	EMV_TAG_TM_FLOORLMT(O)[1]	REF_V: 0x00002710(100.00)
	Tag definition refers to Appendix A **Via API:** EMV_Kern_uiSetTLV(ucKernelID = EMV_KERNELID_VISA) EMV_Kern_uiSetTLVList(ucKernelID = EMV_KERNELID_VISA)	
E: After Read Application Data	DEF_TAG_PAN_IN_BLACK(O)	REF_V:0x00 or 0x01
	Tag definition refers to Appendix A **Via API:** EMV_Kern_uiSetTLV(ucKernelID = EMV_KERNELID_VISA) EMV_Kern_uiSetTLVList(ucKernelID = EMV_KERNELID_VISA)	

Remarks:

EMVKernel will not require an 'Online Authorization Event' when the transaction performs an approved offline. EMVKernel will only require ONLINE_PIN after 'Transaction End Event', and application can get CVM Type from data of 'EMV_tTransData->ucCVM'. The application performs its own online PIN verification operation, and subsequent processing does not require interaction with the kernel. And Kernel will not require an ONLINE_PIN if transaction performs an approved offline.

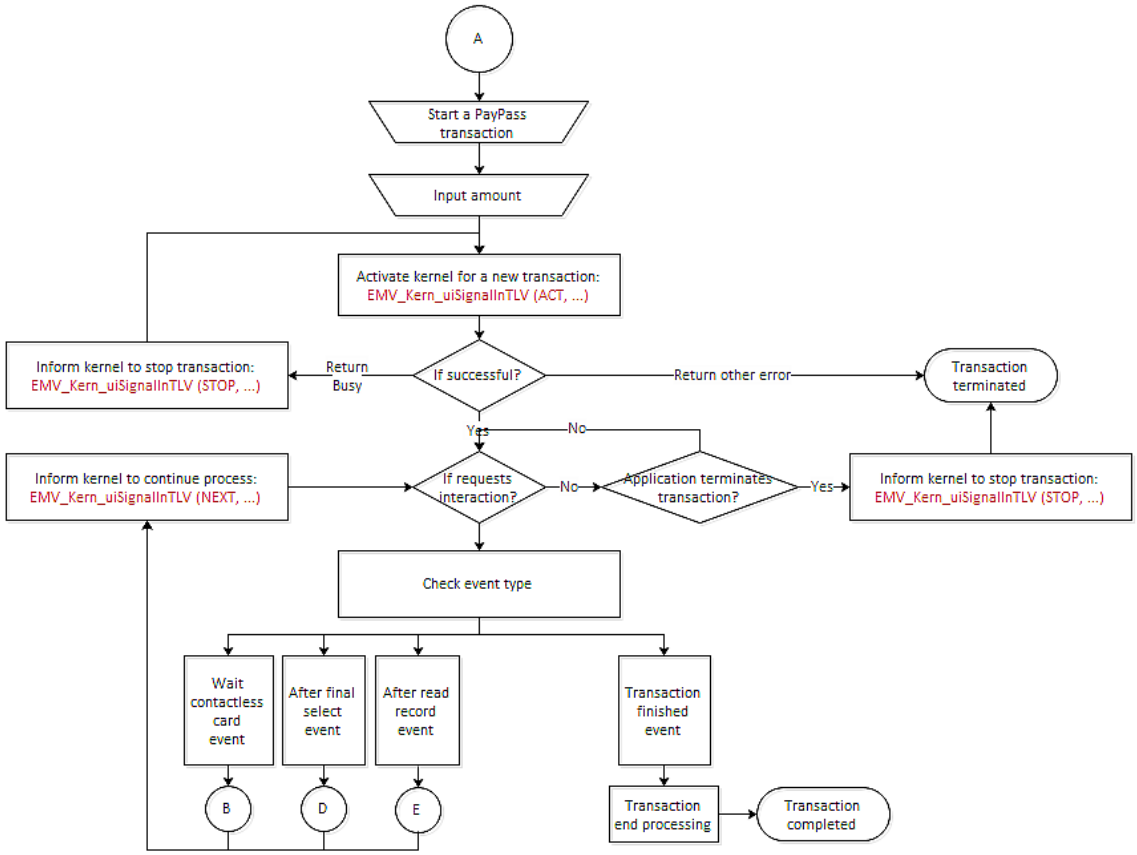
[1] EMV_TAG_TM_FLOORLMT (Tag: 9F1B) is a 4-byte Hex code, such as $100.00 needs to set as "\x00\x00\x27\x10".

For the detailed Return Code of Transaction End Event, you can refer to the <Appendix C Transaction Return Code>.

The Return Code about 'EMV_RESULT_REPOWERICC' which is returned from kernel when doing a Visa transaction commented as bellow:

In GPO Step, Kernel will return "EMV_RESULT_REPOWERICC" when the card returns SW=6986, and the application needs to power off for 1~1.5 seconds and power on again, and prompt the cardholder the message "Refer to you device", then present the card again to complete the transaction. During the second Present Card, the context of the current transaction persists unchanged, including Amount, Authorized and Transaction Currency code, and Pre-Condition and Pre-Preprocessing data.

5.7 Master Card (PayPass)



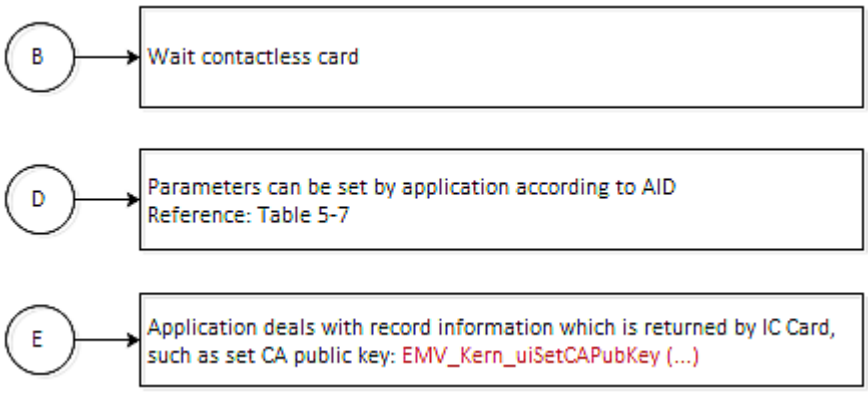
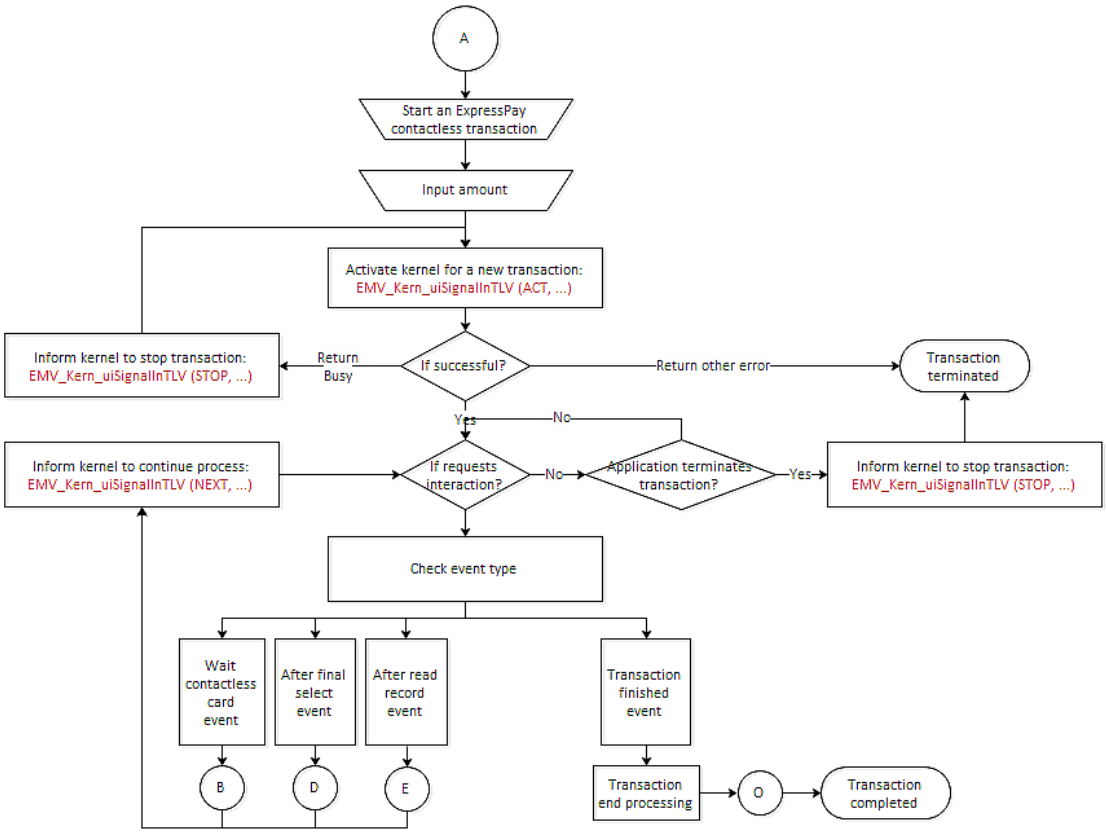


Table 5-7 Transaction Parameter Requirements for PayPass

Note: REF_V: Reference Value, M: Mandatory, O: Optional, C: Conditional

<table>
<tr><th>Event Type</th><th>Transaction Parameter Requirement</th></tr>
<tr><td>D: After Final Selection</td><td>After final application selection, the AID is decided. Some parameters can be set with different value based on different AID.
The following parameters are recommended for PayPass flow:
<table>
<tr><th>Parameters Tag</th><th>Property</th><th>Reference Value</th><th>Default Value</th></tr>
<tr><td>EMV_TAG_TM_TERMTYPE</td><td>M</td><td>0x22</td><td></td></tr>
<tr><td>EMV_TAG_TM_TRANSTYPE</td><td>O</td><td>0x00(Purchase)</td><td>0x00</td></tr>
<tr><td>EMV_TAG_TM_CAP</td><td>O</td><td>0xE0F808</td><td>0x000000</td></tr>
<tr><td>EMV_TAG_TM_CAP_AD</td><td>O</td><td>0x6000F0A001</td><td>0x0000000000</td></tr>
<tr><td>EMV_TAG_TM_TRANSDATE</td><td>M</td><td>0x171216</td><td></td></tr>
<tr><td>EMV_TAG_TM_TRANSTIME</td><td>M</td><td>0x131535</td><td></td></tr>
<tr><td>EMV_TAG_TM_AUTHAMNTN</td><td>M</td><td>0x000000000100</td><td></td></tr>
<tr><td>EMV_TAG_TM_CNTRYCODE</td><td>M</td><td>0x0156</td><td></td></tr>
<tr><td>EMV_TAG_TM_CURCODE</td><td>M</td><td>0x0156</td><td></td></tr>
<tr><td>DEF_TAG_TAC_DECLINE</td><td>O</td><td>0x840000000C</td><td>0x840000000C</td></tr>
<tr><td>DEF_TAG_TAC_ONLINE</td><td>O</td><td>0x840000000C</td><td>0x840000000C</td></tr>
<tr><td>DEF_TAG_TAC_DEFAULT</td><td>O</td><td>0x840000000C</td><td>0x840000000C</td></tr>
<tr><td>DEF_TAG_M_TRANS_MOD</td><td>O</td><td>0x02</td><td>0x02</td></tr>
<tr><td>DEF_TAG_M_BALANCE_SUP</td><td>O</td><td>0x00</td><td>0x00</td></tr>
<tr><td>DEF_TAG_TORN_SUPPORT</td><td>O</td><td>0x00</td><td>0x00</td></tr>
<tr><td>DEF_TAG_M_CDV_SUP</td><td>O</td><td>0x00</td><td>0x00</td></tr>
<tr><td>M_TAG_TM_TRANS_LIMIT</td><td>M</td><td>0x000000030000 (300.00)</td><td>0x000000000000</td></tr>
<tr><td>M_TAG_TM_TRANS_LIMIT_CDV</td><td>M</td><td>0x000000050000 (500.00)</td><td>0x000000000000</td></tr>
<tr><td>M_TAG_TM_CVM_LIMIT</td><td>O</td><td>0x000000001000(10.00)</td><td>0x000000000000</td></tr>
<tr><td>M_TAG_TM_FLOOR_LIMIT</td><td>O</td><td>0x000000010000(100.00)</td><td>0x000000000000</td></tr>
<tr><td>DEF_TAG_M_REQ_CVM</td><td>O</td><td>0x60</td><td>0x00</td></tr>
<tr><td>DEF_TAG_M_REQ_NOCVM</td><td>O</td><td>0x08</td><td>0x00</td></tr>
<tr><td>DEF_TAG_M_MAG_REQ_CVM</td><td>O</td><td>0x10</td><td>0xF0</td></tr>
<tr><td>DEF_TAG_M_MAG_REQ_NOCVM</td><td>O</td><td>0x00</td><td>0xF0</td></tr>
<tr><td>EMV_TAG_TM_CUREXP</td><td>O</td><td>0x02</td><td></td></tr>
<tr><td>M_TAG_TM_9F7C</td><td>O</td><td>0x010101010101010101010101010101010101010101</td><td></td></tr>
<tr><td>M_TAG_TM_9F53</td><td>O</td><td>0x01</td><td></td></tr>
<tr><td>M_TAG_TM_9F6D</td><td>O</td><td>0x0001</td><td>0x0001</td></tr>
</table>
Tag Definition refers to Appendix A
Via API:
EMV_Kern_uiSetTLV
EMV_Kern_uiSetTLVList</td></tr>
</table>

5.8 AMEX


A
Start an ExpressPay contactless transaction
Input amount
Activate kernel for a new transaction: EMV_Kern_uiSignalInTLV (ACT, ...)
If successful?
Return Busy
Inform kernel to stop transaction: EMV_Kern_uiSignalInTLV (STOP, ...)
Return other error
Transaction terminated
Yes
No
If requests interaction?
No
Application terminates transaction?
Yes
Inform kernel to continue process: EMV_Kern_uiSignalInTLV (NEXT, ...)
Inform kernel to stop transaction: EMV_Kern_uiSignalInTLV (STOP, ...)
Check event type
Wait contactless card event
After final select event
After read record event
Transaction finished event
B
D
E
Transaction end processing
O
Transaction completed

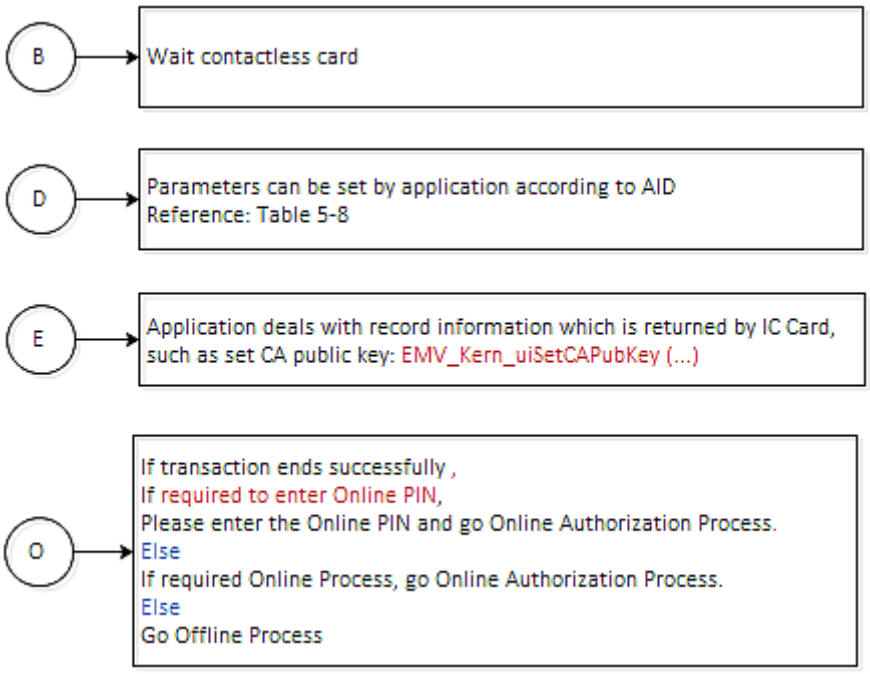

B
Wait contactless card
D
Parameters can be set by application according to AID
Reference: Table 5-8
E
Application deals with record information which is returned by IC Card, such as set CA public key: EMV_Kern_uiSetCAPubKey (...)
O
If transaction ends successfully ,
If required to enter Online PIN,
Please enter the Online PIN and go Online Authorization Process.
Else
If required Online Process, go Online Authorization Process.
Else
Go Offline Process

Table 5-8 Transaction Parameter Requirements for AMEX

Note: REF_V: Reference Value, M: Mandatory, O: Optional, C: Conditional

Event Type	Transaction Parameter Requirement	
ACT: Active kernel of a new transaction	DEF_TAG_PSE_FLAG(M)	REF_V: 0x03
	DEF_TAG_PPSE_6A82_TURNTO_AID LIST(M)	REF_V: 0x01
	Tag Definition refers to Appendix A **Via API:** **EMV_Kern_uiSetTLV**(ucKernelID = EMV_KERNELID_AMEX) **EMV_Kern_uiSetTLVList**(ucKernelID = EMV_KERNELID_AMEX)	
D: After Final Selection	After final application selection, the AID is decided. All parameters Tag are required to set at this step as bellow:	
	EMV_TAG_TM_TERMTYPE(M)	REF_V: 0x22
	EMV_TAG_TM_CAP(M)	REF_V: 0xE0E8C8
	EMV_TAG_TM_CAP_AD(M)	REF_V: 0x6000F0B001
	A_TAG_TM_9F6D(M)	REF_V: 0xC0
	A_TAG_TM_9F6E(M)	REF_V: 0xD8E00000
	EMV_TAG_TM_CNTRYCODE(M)	REF_V: 0x0620
	EMV_TAG_TM_CURCODE(M)	REF_V: 0x0978
	EMV_TAG_TM_APPVERNO(M)	REF_V: 0x0001(Fixed Value)
	DEF_TAG_TAC_DECLINE(O)	REF_V: 0x0000000000
	DEF_TAG_TAC_ONLINE(O)	REF_V: 0x0000000000
	DEF_TAG_TAC_DEFAULT(O)	REF_V: 0x0000000000
	A_TAG_TM_TRANS_LIMIT(O)	REF_V: 0x000000015000
	A_TAG_TM_FLOOR_LIMIT(O)	REF_V: 0x000000010000
	A_TAG_TM_CVM_LIMIT(O)	REF_V: 0x000000005000
	EMV_TAG_TM_FLOORLMT(O)[2]	REF_V: 0x00002710(100.00)
	EMV_TAG_TM_TRANSTYPE(M)	REF_V: 0x00(Purchase)
	EMV_TAG_TM_AUTHAMNTN(M)	REF_V: 0x000000000100(1.00)
	EMV_TAG_TM_OTHERAMNTN(O)	REF_V: 0x0000000000
	EMV_TAG_TM_TRANSDATE(M)	REF_V: 0x161216
	EMV_TAG_TM_TRANSTIME(M)	REF_V: 0x161535
	EMV_TAG_TM_TRSEQCNTR(O)	REF_V: 0x00000001
	A_TAG_PREAGAIN(C)	REF_V: 0x00 or 0x01
	Tag Definition refers to Appendix A **Via API:** **EMV_Kern_uiSetTLV**(ucKernelID = EMV_KERNELID_AMEX) **EMV_Kern_uiSetTLVList**(ucKernelID = EMV_KERNELID_AMEX)	
E: After Read Application Data	DEF_TAG_PAN_IN_BLACK(O)	REF_V: 0x00 or 0x01
	A_TAG_TM_TRANS_LIMIT(C)	REF_V: Sets of Dynamic Reader Limits
	A_TAG_TM_FLOOR_LIMIT(C)	REF_V: Sets of Dynamic Reader Limits
	A_TAG_TM_CVM_LIMIT(C)	REF_V: Sets of Dynamic Reader Limits
	A_TAG_TM_IN_CARD_BIN_RANGE(O)	REF_V:0xA0 or 0x00
	Tag Definition refers to Appendix A **Via API:** **EMV_Kern_uiSetTLV**(ucKernelID = EMV_KERNELID_AMEX) **EMV_Kern_uiSetTLVList**(ucKernelID = EMV_KERNELID_AMEX)	

[2] EMV_TAG_TM_FLOORLMT (Tag: 9F1B) is a 4-byte Hex Code, such as $100.00 needs to set as "\x00\x00\x27\x10".

Remarks:
EMVKernel will not require an 'Online Authorization Event' when the transaction performs an approved offline. EMVKernel will only require ONLINE_PIN after 'Transaction End Event', and application can get CVM Type from data of 'EMV_tTransData->ucCVM'. The application performs its own online PIN verification operation, and subsequent processing does not require interaction with the kernel.

The following underlined content is only using for Kernel Version which is less than or equal to Ver17-11-08 Version.

~~About Return Code, you can refer to the table below priority, and then refer to the <Appendix C Transaction Return Code>.~~

~~#define AMEX_ACTION_SWITCH_INTERFACE 0xF40000E1//18.2.1 Try Another Interface~~
~~#define AMEX_ACTION_REQUEST_ANOTHER_PAYMENT 0xF40000E2//18.2.3 Try Another Payment~~
~~#define AMEX_ACTION_RESTART_TRY_AGAIN 0xF40000E3//Start,And Try Again~~
~~#define AMEX_ACTION_ENDAPPLICATION 0xF40000E5//End Application~~

~~The Return Code about 'AMEX_ACTION_RESTART_TRY_AGAIN' which return from kernel when do a Visa transaction commented as bellow:~~

~~In 2 Steps (which Mobile CVM performed but failed in CVM Step or which Card return 6984 in TAA1 step), Kernel will return AMEX_ACTION_RESTART_TRY_AGAIN, and application need power off RF field period 1 to 3 seconds (default 1.5s) and power on it again, and prompt the cardholder the message "See Phone for Instructions", application shall indicates A_TAG_PREAGAIN(Tag: DF8130) to 0x01 to the Kernel, and then present the card again do the transaction to completion. During the second Present Card, the context of the current transation persists unchanged, including Amount, Authorized and Transaction Currency code, and Pre-Condition and Pre-Preprocessing data remains unchanged.~~

5.9 DISCOVER

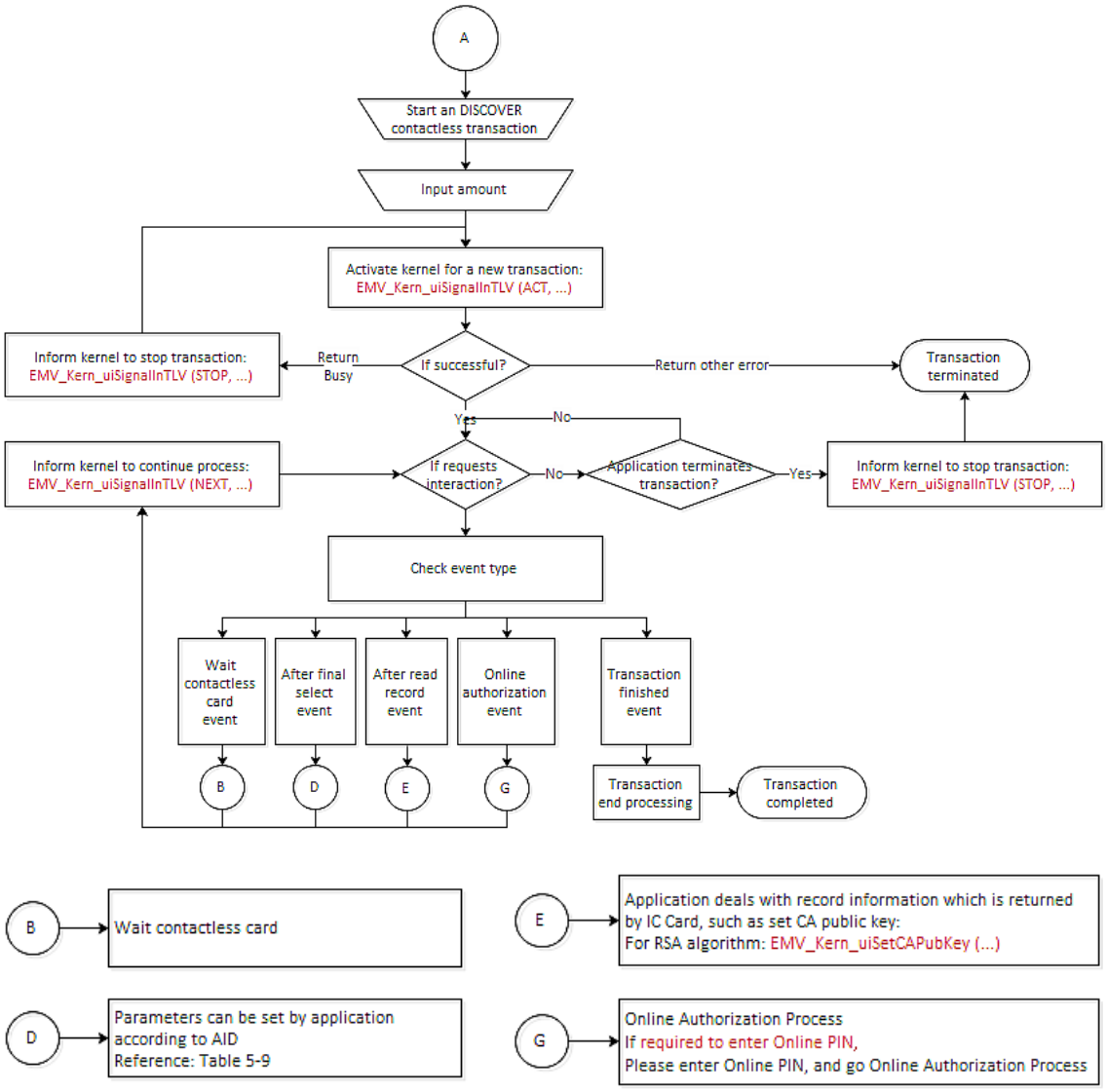

A
Start an DISCOVER contactless transaction
Input amount
Activate kernel for a new transaction: EMV_Kern_uiSignalInTLV (ACT, ...)
If successful?
Return Busy
Inform kernel to stop transaction: EMV_Kern_uiSignalInTLV (STOP, ...)
Return other error
Transaction terminated
Yes
No
If requests interaction?
No
Application terminates transaction?
Yes
Inform kernel to continue process: EMV_Kern_uiSignalInTLV (NEXT, ...)
Inform kernel to stop transaction: EMV_Kern_uiSignalInTLV (STOP, ...)
Check event type
Wait contactless card event
After final select event
After read record event
Online authorization event
Transaction finished event
B
D
E
G
Transaction end processing
Transaction completed
B
Wait contactless card
E
Application deals with record information which is returned by IC Card, such as set CA public key:
For RSA algorithm: EMV_Kern_uiSetCAPubKey (...)
D
Parameters can be set by application according to AID
Reference: Table 5-9
G
Online Authorization Process
If required to enter Online PIN,
Please enter Online PIN, and go Online Authorization Process

Table 5-9 Transaction Parameter Requirements for DISCOVER

Note: REF_V: Reference Value, M: Mandatory, O: Optional, C: Conditional

Event Type	Transaction Parameter Requirement	
ACT: active kernel of a new transaction	DEF_TAG_PSE_FLAG(M)	REF_V: 0x03
	(ZIP mode please refer to bit2 of 'ucPartSIt' from 6.3 EMV_Kern_uiManageAID)	
	EMV_TAG_TM_AUTHAMNTN(M)	REF_V: 0x000000000100(1.00)
	EMV_TAG_TM_OTHERAMNTN(O)	REF_V: 0x0000000000
	D_TAG_TM_9F66(M)	REF_V: 0x26004000
	D_TAG_TM_RD_RCP(O)	REF_V: 0xFC80
	D_TAG_TM_TRANS_LIMIT(O)	REF_V: 0x000000015000
	D_TAG_TM_FLOOR_LIMIT(O)	REF_V: 0x000000010000
	D_TAG_TM_CVM_LIMIT(O)	REF_V: 0x000000005000
	EMV_TAG_TM_FLOORLMT(O)[3]	REF_V: 0x00002710(100.00)
	DEF_TAG_D_ISSUERSCRIPT_EXCUTIVE(C)	REF_V: 0x01 or 0x00
	Tag Definition refers to Appendix A **Via API:** **EMV_Kern_uiSetTLV**(ucKernelID = EMV_KERNELID_DISCOVER) **EMV_Kern_uiSetTLVList**(ucKernelID = EMV_KERNELID_DISCOVER)	
D: After Final Selection	After final application selection, the AID is decided. All parameters Tag are required to set at this step as bellow:	
	EMV_TAG_TM_TERMTYPE(M)	REF_V: 0x22
	EMV_TAG_TM_CAP(M)	REF_V: 0xE0E8C8
	EMV_TAG_TM_CAP_AD(M)	REF_V: 0x6000F0B001
	EMV_TAG_TM_CNTRYCODE(M)	REF_V: 0x0840
	EMV_TAG_TM_CURCODE(M)	REF_V: 0x0840
	EMV_TAG_TM_APPVERNO(M)	REF_V: 0x0100
	DEF_TAG_TAC_DECLINE(O)	REF_V: 0x0000000000
	DEF_TAG_TAC_ONLINE(O)	REF_V: 0x0000000000
	DEF_TAG_TAC_DEFAULT(O)	REF_V: 0x0000000000
	EMV_TAG_TM_TRANSTYPE(M)	REF_V: 0x00(Purchase)
	EMV_TAG_TM_TRANSDATE(M)	REF_V: 0x161216
	EMV_TAG_TM_TRANSTIME(M)	REF_V: 0x161535
	EMV_TAG_TM_TRSEQCNTR(O)	REF_V: 0x00000001
	Tag Definition refers to Appendix A **Via API:** **EMV_Kern_uiSetTLV**(ucKernelID = EMV_KERNELID_DISCOVER) **EMV_Kern_uiSetTLVList**(ucKernelID = EMV_KERNELID_DISCOVER)	
E: After Read Application Data	DEF_TAG_PAN_IN_BLACK(O)	REF_V: 0x00 or 0x01
	Tag Definition refers to Appendix A **Via API:** **EMV_Kern_uiSetTLV**(ucKernelID = EMV_KERNELID_DISCOVER) **EMV_Kern_uiSetTLVList**(ucKernelID = EMV_KERNELID_DISCOVER)	
G: After Online Authorization Process	DEF_TAG_ONLINE_STATUS(C)	REF_V: 0x00 or 0x01
	EMV_TAG_TM_ARC(C)	REF_V: ARC from Host
	DEF_TAG_AUTHORIZE_FLAG(C)	REF_V: 0x00 or 0x01
	EMV_TAG_TM_AUTHCODE(C)	REF_V: Var.
	DEF_TAG_HOST_TLVDATA(C)	REF_V: Var.,such as Issuer Script
	Tag Definition refers to Appendix A **Via API:** **EMV_Kern_uiSetTLV**(ucKernelID = EMV_KERNELID_DISCOVER) **EMV_Kern_uiSetTLVList**(ucKernelID = EMV_KERNELID_DISCOVER)	

[3]EMV_TAG_TM_FLOORLMT (Tag: 9F1B) is a 4-byte Hex code, such as $100.00 needs to set as "\x00\x00\x27\x10".

Remarks:
EMVKernel will not require an 'Online Authorization Event' when the transaction performs an approved offline. EMVKernel will only require ONLINE_PIN after 'Transaction End Event', and application can get CVM Type from data of 'EMV_tTransData->ucCVM'. The application performs its own online PIN verification operation, and subsequent processing does not require interaction with the kernel.

But EMVKernel will require an 'Online Authorization Event' when the transaction performs an Online required, and the online request output parameter EMV_tTransData->ucCVM specifies whether the online PIN is required. The application performs the online PIN verification operation and continues the online event processing until the transaction ends. At this point in the transaction end event, the kernel will not request cardholder verification again, i.e. a transaction kernel will only request cardholder verification once.

The following underlined content is only using for Kernel Version which is less than or equal to Ver17-11-08 Version.

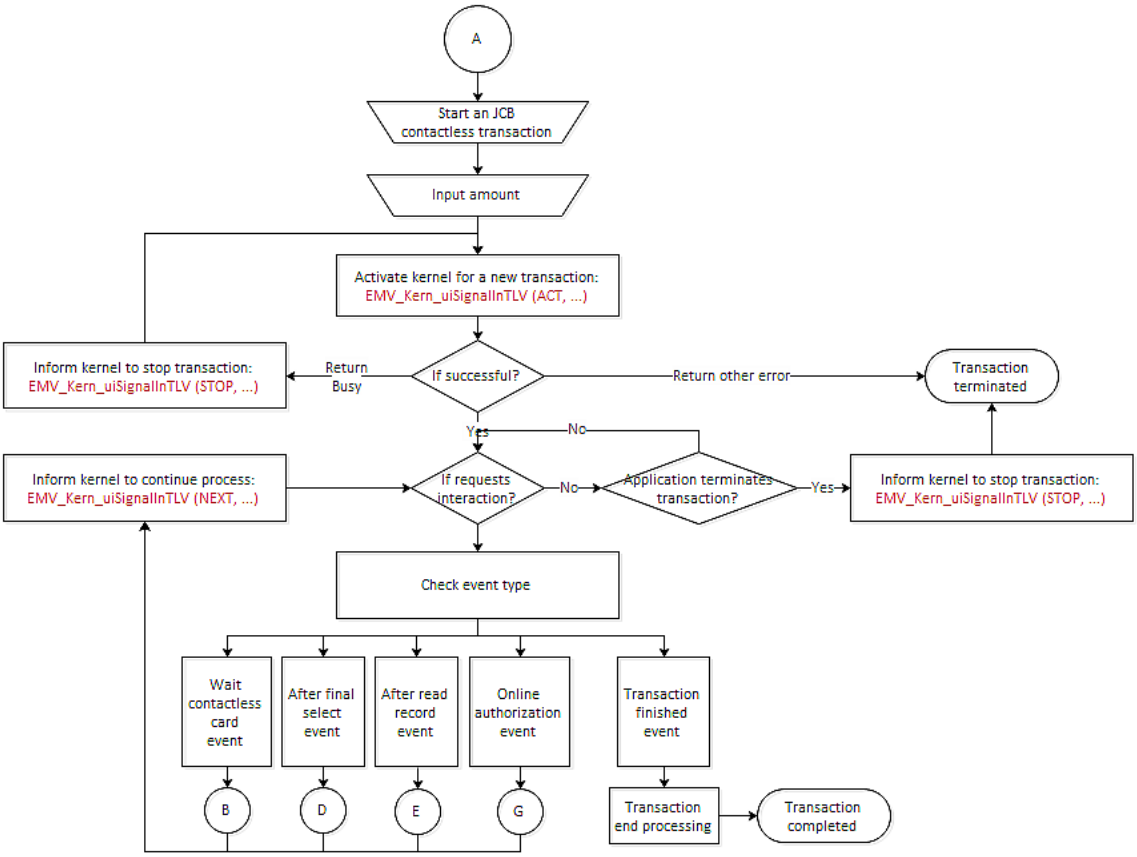
~~*About Return Code, you can refer to the table below priority, and then refer to the <Appendix C Transaction Return Code>.*~~

#define DISCOVER_ACTION_SWITCH_INTERFACE 0xF60000E1//Try Another Interface
~~*#define DISCOVER_ACTION_REQUEST_ANOTHER_PAYMENT 0xF60000E2//Try Another Payment*~~
#define DISCOVER_ACTION_RESTART_TRY_AGAIN 0xF60000E3//Start,And Try Again
#define DISCOVER_ACTION_ENDAPPLICATION 0xF60000E5//End Application(Terminate)
#define DISCOVER_ACTION_USE_ANOTHER_CARD 0xF60000E7//Try Another Card(Ask user to use another card)

~~*The Return Code about 'DISCOVER_ACTION_RESTART_TRY_AGAIN' which return from kernel when do a Visa transaction commented as bellow:*~~

~~*In GPO Step, Kernel will return DISCOVER_ACTION_RESTART_TRY_AGAIN when card return SW=6986, and application need not power off RF field, and prompt the cardholder the message "Please Removed Card, Present Card Again", then present the card again do the transaction to completion. During the second Present Card, the context of the current transaction persists unchanged, including Amount, Authorized and Transaction Currency code, and Pre-Condition and Pre-Preprocessing data remains unchanged.*~~

5.10 JCB


A
Start an JCB contactless transaction
Input amount
Activate kernel for a new transaction: EMV_Kern_uiSignalInTLV (ACT, ...)
If successful?
Return Busy
Return other error
Transaction terminated
Inform kernel to stop transaction: EMV_Kern_uiSignalInTLV (STOP, ...)
Yes
No
If requests interaction?
No
Application terminates transaction?
Yes
Inform kernel to stop transaction: EMV_Kern_uiSignalInTLV (STOP, ...)
Inform kernel to continue process: EMV_Kern_uiSignalInTLV (NEXT, ...)
Check event type
Wait contactless card event
After final select event
After read record event
Online authorization event
Transaction finished event
B
D
E
G
Transaction end processing
Transaction completed

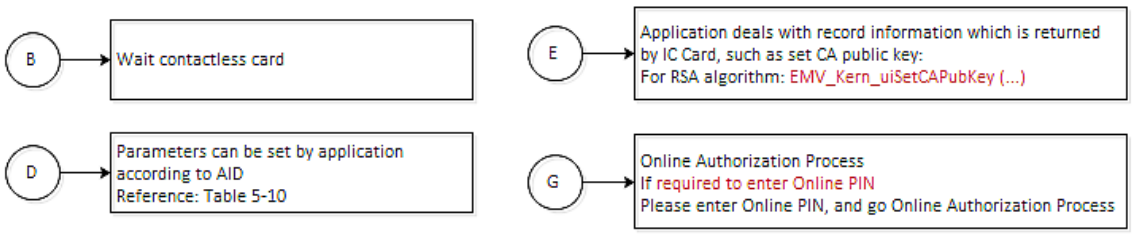
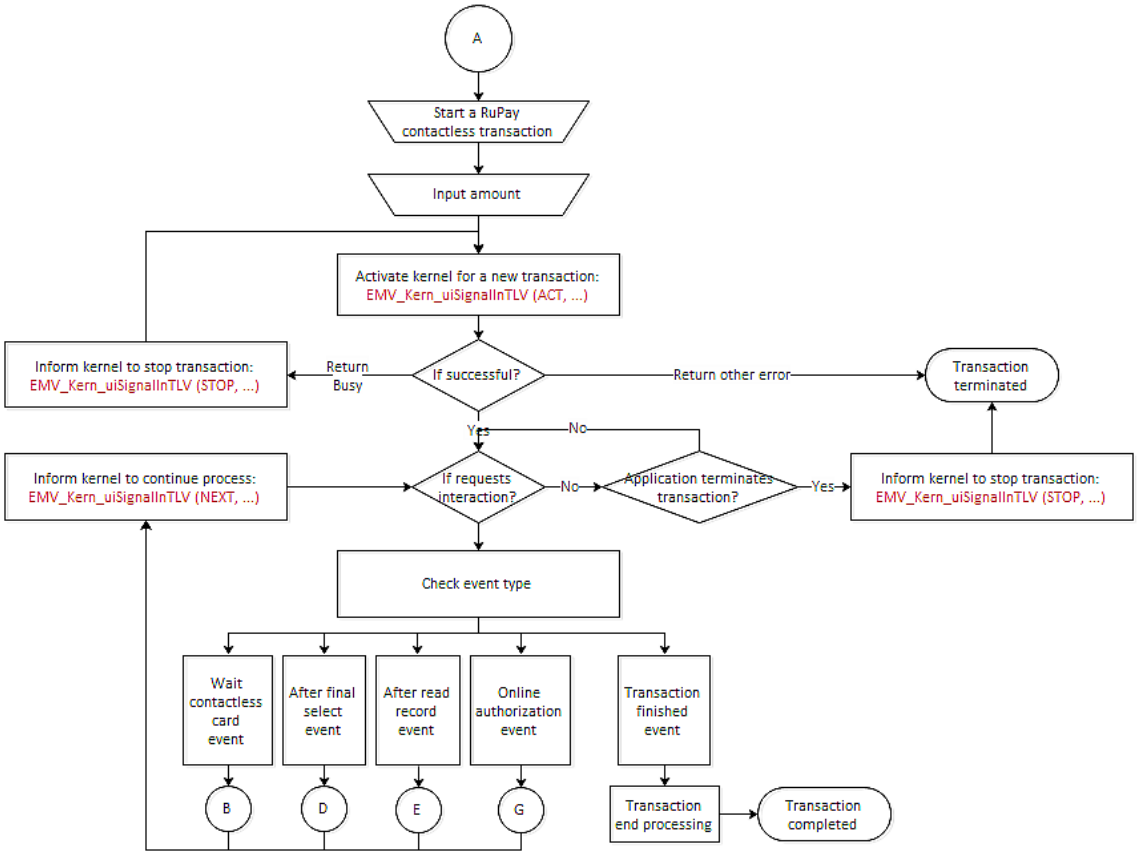

B
Wait contactless card
E
Application deals with record information which is returned by IC Card, such as set CA public key:
For RSA algorithm: EMV_Kern_uiSetCAPubKey (...)
D
Parameters can be set by application according to AID
Reference: Table 5-10
G
Online Authorization Process
If required to enter Online PIN
Please enter Online PIN, and go Online Authorization Process

Table 5-10 Transaction Parameter Requirements for JCB

Note: REF_V: Reference Value, M: Mandatory, O: Optional, C: Conditional

<table>
<tr><th>Event Type</th><th>Transaction Parameter Requirement</th></tr>
<tr><td>D: After Final Selection</td><td>After final application selection, the AID is decided. Some parameters can be set with different value based on different AID.
The following parameters are recommended for PayPass flow:
<table>
<tr><th>Parameters Tag</th><th>Property</th><th>Reference Value</th><th>Default Value</th></tr>
<tr><td>EMV_TAG_TM_TERMTYPE</td><td>M</td><td>0x22</td><td></td></tr>
<tr><td>EMV_TAG_TM_TRANSTYPE</td><td>O</td><td>0x00(Purchase)</td><td>0x00</td></tr>
<tr><td>EMV_TAG_TM_CAP</td><td>O</td><td>0xE0F808</td><td>0x000000</td></tr>
<tr><td>EMV_TAG_TM_CAP_AD</td><td>O</td><td>0x6000F0A001</td><td>0x0000000000</td></tr>
<tr><td>EMV_TAG_TM_TRANSDATE</td><td>M</td><td>0x171216</td><td></td></tr>
<tr><td>EMV_TAG_TM_TRANSTIME</td><td>M</td><td>0x131535</td><td></td></tr>
<tr><td>EMV_TAG_TM_AUTHAMNTN</td><td>M</td><td>0x000000000100</td><td></td></tr>
<tr><td>EMV_TAG_TM_CNTRYCODE</td><td>M</td><td>0x0156</td><td></td></tr>
<tr><td>EMV_TAG_TM_CURCODE</td><td>M</td><td>0x0156</td><td></td></tr>
<tr><td>DEF_TAG_TAC_DECLINE</td><td>O</td><td>0x840000000C</td><td>0x840000000C</td></tr>
<tr><td>DEF_TAG_TAC_ONLINE</td><td>O</td><td>0x840000000C</td><td>0x840000000C</td></tr>
<tr><td>DEF_TAG_TAC_DEFAULT</td><td>O</td><td>0x840000000C</td><td>0x840000000C</td></tr>
<tr><td>DEF_TAG_M_TRANS_MOD</td><td>O</td><td>0x02</td><td>0x02</td></tr>
<tr><td>DEF_TAG_M_BALANCE_SUP</td><td>O</td><td>0x00</td><td>0x00</td></tr>
<tr><td>DEF_TAG_TORN_SUPPORT</td><td>O</td><td>0x00</td><td>0x00</td></tr>
<tr><td>DEF_TAG_M_CDV_SUP</td><td>O</td><td>0x00</td><td>0x00</td></tr>
<tr><td>M_TAG_TM_TRANS_LIMIT</td><td>M</td><td>0x000000030000 (300.00)</td><td>0x000000000000</td></tr>
<tr><td>M_TAG_TM_TRANS_LIMIT_CDV</td><td>M</td><td>0x000000050000 (500.00)</td><td>0x000000000000</td></tr>
<tr><td>M_TAG_TM_CVM_LIMIT</td><td>O</td><td>0x000000001000(10.00)</td><td>0x000000000000</td></tr>
<tr><td>M_TAG_TM_FLOOR_LIMIT</td><td>O</td><td>0x000000010000(100.00)</td><td>0x000000000000</td></tr>
<tr><td>DEF_TAG_M_REQ_CVM</td><td>O</td><td>0x60</td><td>0x00</td></tr>
<tr><td>DEF_TAG_M_REQ_NOCVM</td><td>O</td><td>0x08</td><td>0x00</td></tr>
<tr><td>DEF_TAG_M_MAG_REQ_CVM</td><td>O</td><td>0x10</td><td>0xF0</td></tr>
<tr><td>DEF_TAG_M_MAG_REQ_NOCVM</td><td>O</td><td>0x00</td><td>0xF0</td></tr>
<tr><td>EMV_TAG_TM_CUREXP</td><td>O</td><td>0x02</td><td></td></tr>
<tr><td>M_TAG_TM_9F7C</td><td>O</td><td>0x010101010101010101010101010101010101010101</td><td></td></tr>
<tr><td>M_TAG_TM_9F53</td><td>O</td><td>0x01</td><td></td></tr>
<tr><td>M_TAG_TM_9F6D</td><td>O</td><td>0x0001</td><td>0x0001</td></tr>
<tr><td colspan="4">Tag Definition refers to Appendix A
Via API:
EMV_Kern_uiSetTLV
EMV_Kern_uiSetTLVList</td></tr>
</table>
</td></tr>
</table>

5.11 RuPay


A
Start a RuPay contactless transaction
Input amount
Activate kernel for a new transaction: EMV_Kern_uiSignalInTLV (ACT, ...)
If successful?
Return Busy
Inform kernel to stop transaction: EMV_Kern_uiSignalInTLV (STOP, ...)
Return other error
Transaction terminated
Yes
No
If requests interaction?
No
Application terminates transaction?
Yes
Inform kernel to continue process: EMV_Kern_uiSignalInTLV (NEXT, ...)
Inform kernel to stop transaction: EMV_Kern_uiSignalInTLV (STOP, ...)
Check event type
Wait contactless card event
After final select event
After read record event
Online authorization event
Transaction finished event
B
D
E
G
Transaction end processing
Transaction completed

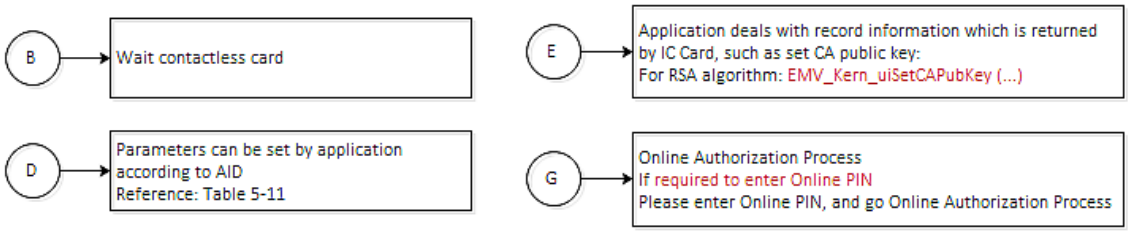

B
Wait contactless card
E
Application deals with record information which is returned by IC Card, such as set CA public key:
For RSA algorithm: EMV_Kern_uiSetCAPubKey (...)
D
Parameters can be set by application according to AID
Reference: Table 5-11
G
Online Authorization Process
If required to enter Online PIN
Please enter Online PIN, and go Online Authorization Process

Table 5-11 Transaction Parameter Requirements for RuPay

Note: REF_V: Reference Value, M: Mandatory, O: Optional, C: Conditional

<table>
<tr><th>Event Type</th><th>Transaction Parameter Requirement</th></tr>
<tr><td>ACT: active kernel of a new transaction</td><td>DEF_TAG_PSE_FLAG(M) REF_V: 0x03
EMV_TAG_TM_AUTHAMNTN(M) REF_V: 0x000000000100(1.00)
EMV_TAG_TM_OTHERAMNTN(O) REF_V: 0x0000000000
R_TAG_TM_TRANS_LIMIT(M) REF_V: 0x000000015000
EMV_TAG_TM_CAP(M) REF_V: 0xE06840

1) About CVM capability: RuPay only support 3 types of CVM as bellow: Online PIN B2b7=1, Signature B2b6=1, NoCVM B2b4=1
2) About security capability: RuPay only support qDDA for contactless, B3b7=1, so Byte3=0x40 is recommend.
DEF_TAG_START_RECOVERY(C) REF_V: 0x00 or 0x01 valid for RuPay
R_TAG_TM_TIMELIMIT(O) REF_V: 0x1E(30 seconds)

Tag Definition refers to Appendix A
Via API:
EMV_Kern_uiSetTLV(ucKernelID = EMV_KERNELID_RUPAY)
EMV_Kern_uiSetTLVList(ucKernelID = EMV_KERNELID_RUPAY)</td></tr>
<tr><td>D: After Final Selection</td><td>After final application selection, the AID is decided. All parameters Tags are required to set at this step as bellow:

EMV_TAG_TM_TERMTYPE(M) REF_V: show bellow
<table>
<tr><th>Operating Filed</th><th>Terminal Type</th><th>Value</th><th>Use Case</th></tr>
<tr><td rowspan="5">Retail</td><td>Attended Online only</td><td>21</td><td>Merchants</td></tr>
<tr><td>Attended Offline with Online capability</td><td>22</td><td>Merchants</td></tr>
<tr><td>Attended Offline only</td><td>23</td><td>Merchants</td></tr>
<tr><td>Unattended Offline with Online capability</td><td>25</td><td>Vending Machines</td></tr>
<tr><td>Unattended Offline only</td><td>26</td><td>Vending Machines</td></tr>
<tr><td rowspan="6">Transit</td><td rowspan="2">Attended Offline with Online capability</td><td>92</td><td>Transit with conductor</td></tr>
<tr><td>93</td><td>Fare Adjustment Terminal</td></tr>
<tr><td>Attended Offline only</td><td>94</td><td>Transit with conductor</td></tr>
<tr><td rowspan="2">Unattended Offline with Online capability</td><td>96</td><td>Transit without conductor</td></tr>
<tr><td>97</td><td>Fare Adjustment Terminal</td></tr>
<tr><td>Unattended Offline only</td><td>98</td><td>Transit without conductor</td></tr>
</table>
· EMV_TAG_TM_CAP_AD(M) REF_V: 0xF040F0B001
For this specification, Tag 9F40 B2b7 shall be 1 indicates service creation is support for the terminal to backward compatible with Legacy card.
· R_TAG_TM_SRTRANS_CAP(C) REF_V: 0x002C
1) B2b6: Transit Transaction
2) B2b5: CAT3 Transaction
3) B2b4: Money Add Transaction
4) B2b3: Void Transaction
5) Other bits was RFU
· EMV_TAG_TM_CNTRYCODE(M) REF_V: 0x0356
· EMV_TAG_TM_CURCODE(M) REF_V: 0x0356
· EMV_TAG_TM_APPVERNO(O) REF_V: 0x0002</td></tr>
</table>

	For this specification for this tag is a fixed value, kernel will be 0x0002 default value if application is not setting. · DEF_TAG_TAC_DECLINE(O) REF_V: 0x0000000000 · DEF_TAG_TAC_DEFAULT(O) REF_V: 0x0000000000 · EMV_TAG_TM_TRANSTYPE(M) REF_V: 0x00(Purchase) · EMV_TAG_TM_TRANSDATE(M) REF_V: 0x161216 · EMV_TAG_TM_TRANSTIME(M) REF_V: 0x161535 · EMV_TAG_TM_TRSEQCNTR(O) REF_V: 0x00000001 · R_TAG_TM_DF16(C) REF_V: 0x0000 *If card or terminal does not find mutual support Service ID or terminal does not support service-based transaction, the value shall always be all zeros.* *If card and terminal support service transaction and also find mutual support Service ID, Please refer to Section 5.11.1 for more details.* · R_TAG_TM_DF3A(O) REF_V:0040000000 *If terminal supports Services Creation Processing Capabilities, Tag DF3A B2b7 shall be to 1. If terminal does not support Service Creation, the value shall be set all zeros.* · R_TAG_TM_CVM_LIMIT(R) REF_V: 0x000000015000 · R_TAG_TM_FLOOR_LIMIT(O) · EMV_TAG_TM_FLOORLMT(O) **Tag Definition refers to Appendix A** **Via API:** **EMV_Kern_uiSetTLV**(ucKernelID = EMV_KERNELID_RUPAY) **EMV_Kern_uiSetTLVList**(ucKernelID = EMV_KERNELID_RUPAY)
E: After Read Application Data	· DEF_TAG_PAN_IN_BLACK(O) REF_V: 0x00 or 0x01 · DEF_TAG_RAND_SLT_THRESHOLD (M) REF_V:0x000000005000(50.00) · DEF_TAG_RAND_SLT_PER (M) REF_V: 0x30(BCD) · DEF_TAG_RAND_SLT_MAXPER (M) REF_V: 0x90(BCD) · R_TAG_TM_TSRQ(C) REF_V: 0x08 1010 9500 · R_TAG_TM_DF45(C) REF_V: Var. Max length <=96 · R_TAG_IC_DF47(C)(PRMiss for Legacy Mode or for Non-Legacy PRMacq key versioning) · R_TAG_TM_DF48(C)(PRMacq for Key plant Only) · R_TAG_IC_KCV(C)(PRMacq KCV for Key plant Only) · R_TAG_IC_PRMACQ_KEYINDEX(C)(PRMacq Index for Key plant Only) **Tag Definition refers to Appendix A** **Via API:** **EMV_Kern_uiSetTLV**(ucKernelID = EMV_KERNELID_RUPAY) **EMV_Kern_uiSetTLVList**(ucKernelID = EMV_KERNELID_RUPAY)
G:After Online Authorization Process	· DEF_TAG_ONLINE_STATUS(C) REF_V: 0x00 or 0x01 · EMV_TAG_TM_ARC(C) REF_V: ARC from Host · DEF_TAG_AUTHORIZE_FLAG(C) REF_V: 0x00 or 0x01 · EMV_TAG_TM_AUTHCODE(C) REF_V: Var. · DEF_TAG_HOST_TLVDATA(C) REF_V: Var.,such as Issuer Script · R_TAG_TM_ONLINE_ORNOT(C) REF_V: 0x00 or 0x01 **Tag Definition refers to Appendix A** **Via API:** **EMV_Kern_uiSetTLV**(ucKernelID = EMV_KERNELID_RUPAY) **EMV_Kern_uiSetTLVList**(ucKernelID = EMV_KERNELID_RUPAY)

Remarks:

EMVKernel will not require an 'Online Authorization Event' when the transaction performs a partial-online transaction in Non-Legacy Mode. EMVKernel will only require ONLINE_PIN after

‘Transaction End Event’, and application can get CVM Type from data of ‘EMV_tTransData->ucCVM’ and perform CVM indicate the cardholder.

But EMVKernel will require an ‘Online Authorization Event’ when the transaction performs a ‘full-online’ both Non-Legacy and Legacy Mode. Then application can get CVM Type from data of ‘EMV_tTransData->ucCVM’ and perform CVM indicate the cardholder.

5.11.1 About Service IDs

Terminal always maintains a list of Service IDs from Acquirer if both the terminal and the Acquirer support Service-based transaction.
Card will response a Service Directory(Tag DF07) which contains a list of Service IDs that card supported, and in Final Selection APDU card will provide Service IDs if the card supports a Service-based transaction.
In Final Selection callback Event, the application shall set Service ID(DF16) to kernel if it found a same Service ID that they support mutually and the Service ID must be non-zero value.
For Tag DF07 defined, please refer to **R_TAG_IC_DF07** in Table <**A.9 TAG of EMV_KERNELID_RUPAY**>.

For example,
a) Found a match Service ID
Terminal maintain a Service IDs:
Service ID, Service qualifier, Service data
1010, 08 1010 9500, 09 10 1506 1501 01 1122334455667788 000001 0001 0061
150406125703 000000010000 02 00 00 000A0101 000A0102 05
020001020102060000A01010302

Card Response Service IDs:
DF07=0100607484123000000020010F000A0000000000000000000001011AC0010109500
They found a same Service ID (‘1010’) support mutually, so set TLV (KID=RUPAY, ‘1010’) to kernel.

b) Not Found a match Service ID
Terminal maintain a Service IDs:
Service ID, Service qualifier, Service data
1010, 08 1010 9500, 09 10 1506 1501 01 1122334455667788 000001 0001 0061
150406125703 000000010000 02 00 00 000A0101 000A0102 05
020001020102060000A01010302

Card Response Service IDs: Not contain Service ID
DF07=0100607484123000000020010F000A00000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
00000
They did not found a same Service ID support mutually, so set TLV (KID=RUPAY, ‘0000’) to kernel.

5.11.2 About Torn Recovery AC

Both Non-Legacy and Legacy mode, a transaction can be torn when the card has completed its processing, but as removed prematurely from the contactless RF field before either the card could send its response, or the terminal receives the card's response to the 1st GENERATE AC command., e.g. the command timed out or the Card was detected as having left the RF field.

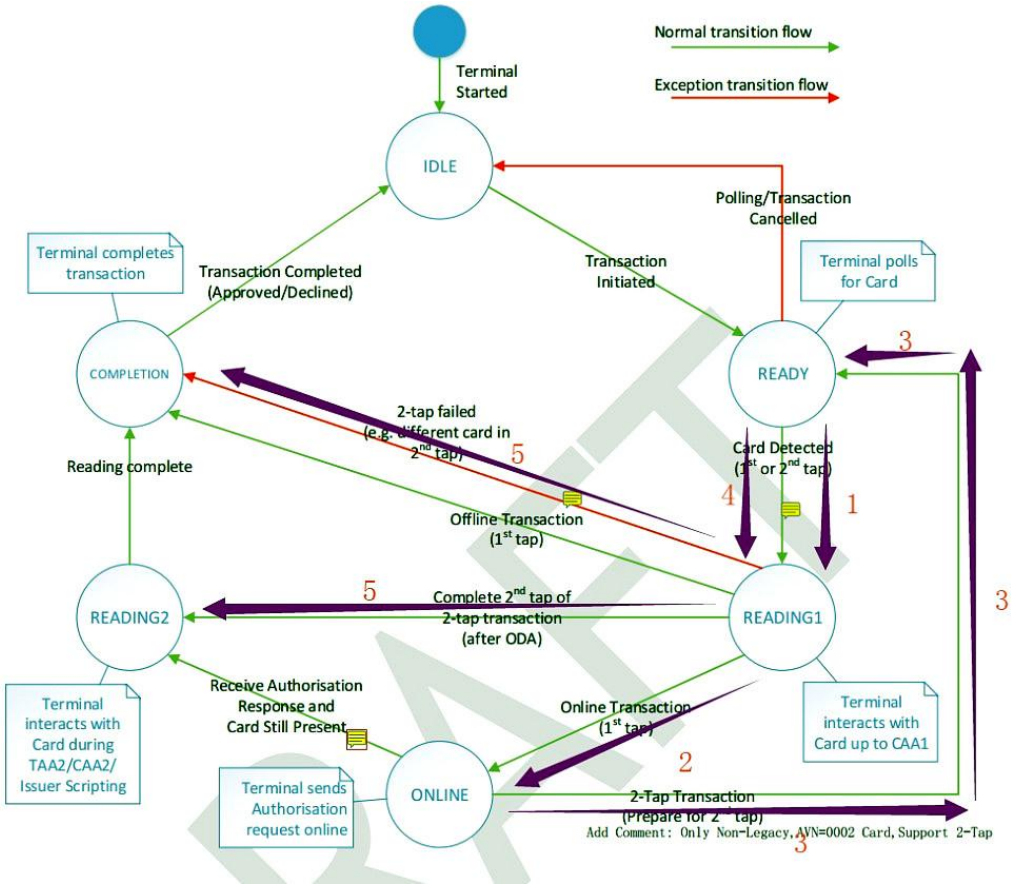
When Torn happened, to be able to recover to a torn transaction, the Kernel will generate a terminal torn recovery log file through Callback Event Function **EXEP_vSendOut**(EMV_INS_SET_TORN) to notify application to save the torn recovery log file which contains in Tag BF918103. And then kernel will perform Callback Event Function **OnWaitCard**(Flag=0x01) immediately, and application notifiies to cardholder to 'present card again' to do torn recovery. And application shall EMV_uiSetTLV(DF918105) to 0x01 indicates kernel it is a torn recovery transaction when application found had a torn recovery log file record was saved, and then kernel will perform Callback Event Function **EXEP_vSendOut**(EMV_INS_GET_TORN) to obtain last torn recovery log file and application needs to submit torn recovery log file contain in Tag BF918103 to kernel through **EMV_Kern_uiSignalInTLV**(EMV_SIGNAL_NEXT), this is termed 'Current Torn Recovery Transaction'.

Kernel also supports which we are termed it as 'completely torn recovery' that application restarts a new transaction completely, i.e. application can EMV_uiSetTLV(DF918105) to 0x01 to notify kernel when last transaction was torn recovery happened and application also found torn recovery log file record prior to start a new transaction.

A torn transaction is only recovered once. When a torn recovery process was done, then the kernel will erase the torn recovery log file through Callback Event Function **EXEP_vSendOut**(EMV_INS_DEL_TORN).

To recover a torn transaction, the terminal must have restarted the contactless transaction within the time limit defined. Please refer to **R_TAG_TM_TIMELIMIT** in Table <**A.9 TAG of EMV_KERNELID_RUPAY**>. If the application is not setting this tag to kernel, the kernel will use a default value as per specification. Else the kernel will use the setting value that application set.

5.11.3 About 2-Tap



State Transition Diagram

According to the above program (**Arrow 1->2->3->4->5**), we can know that:
After ONLINE processing (**Arrow 2**), if the Card supports 2-Tap (i.e. is a non-legacy Card, as indicated by Card AVN = '0002'), then instead of transitioning to READING2 state, the terminal returns to the READY state (**Arrow 3**), so it can process the 2nd tap of a 2-Tap transaction. The terminal must re-use the same transaction context for the 2nd tap, and cache the authorization response so that it can be provided to the Card during the 2nd tap.

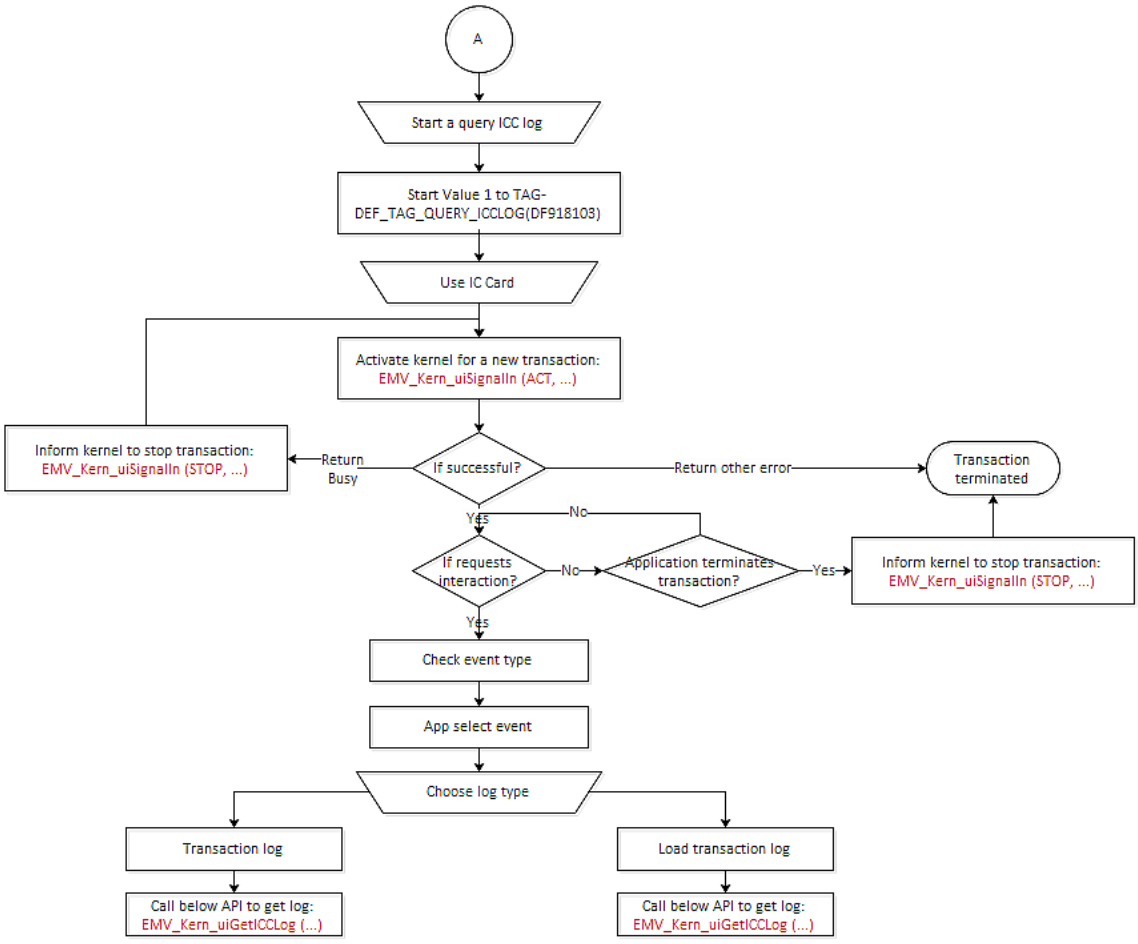
If the terminal is processing the 2nd tap of a 2-Tap transaction, then the terminal must ensure that the Card detected is the same card used in the 1st tap. If the Card is a different, then the terminal must request an AAC for the transaction to end the transaction. If the Card presented is the same card used in the 1st tap, and the GPO response corresponds to the transaction context of the 1st tap, then the terminal must transit to READING2 state immediately after performing Offline Data Authentication (**Arrow 5**).

When a Non-Legacy mode 2nd tap of 2-Tap transaction is to be performed, the kernel will perform Callback Event Function **OnWaitCard** (Flag=0x01) immediately after **EXEP_ucOnlineProcess** Event, and application notifies to cardholder to 'present card again' to do 2nd tap of 2-Tap transaction.

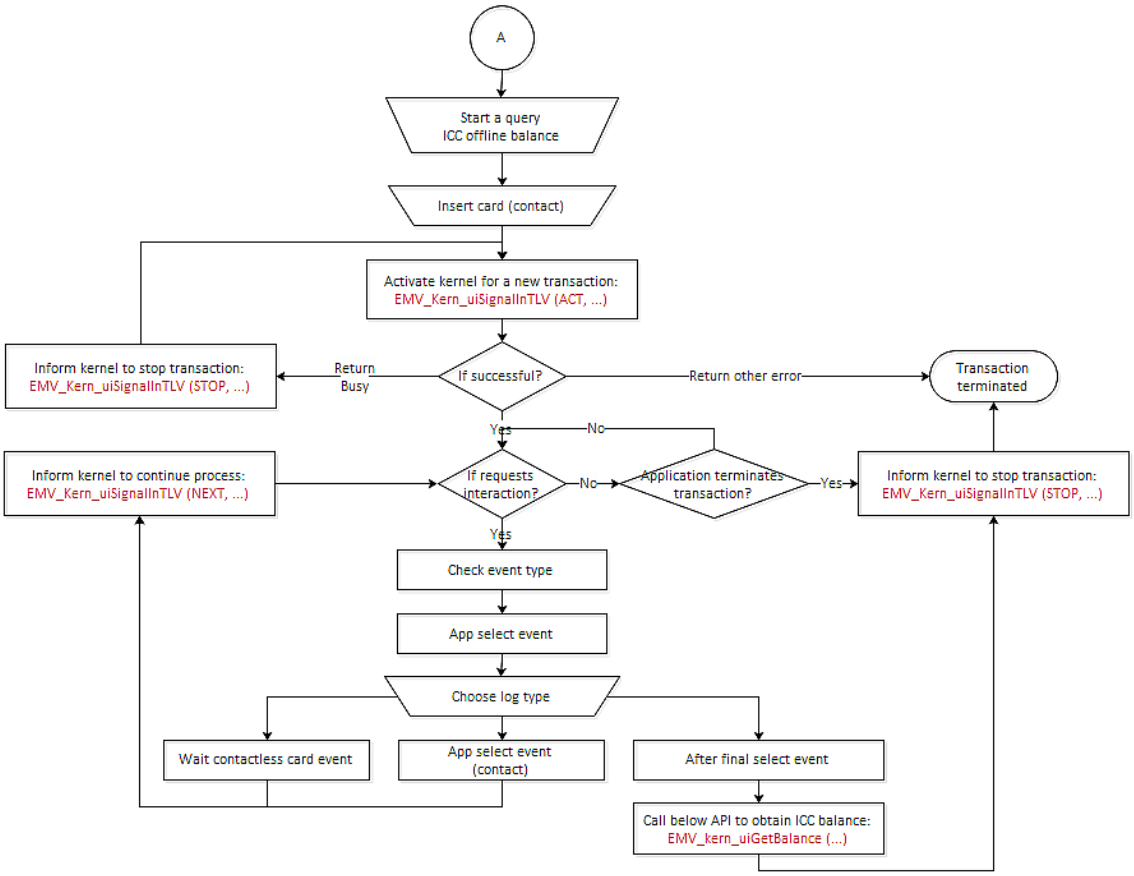
5.11.4 About UnBlock Process (Cards with blocked application)

Both Non-Legacy and Legacy mode, the kernel will proceed GPO up to CAA1 when FINAL SELECTION the card responses SW12=6283(means an application was blocked). Within a block application the card will response AAC (declined) to terminal while terminal requires an ARQC/TC in CAA1 step. Then the kernel switches AAC to ARQC (online) perform Callback Event function **EXEP_ucOnlineProcess** to go online within cryptogram data that card generated response of CAA1 and terminal obtains issuer script from issuer response after online processing. Then the kernel will exchange APDU command within APPLICATION UNBLOCK to the card during Issuer Script Processing.

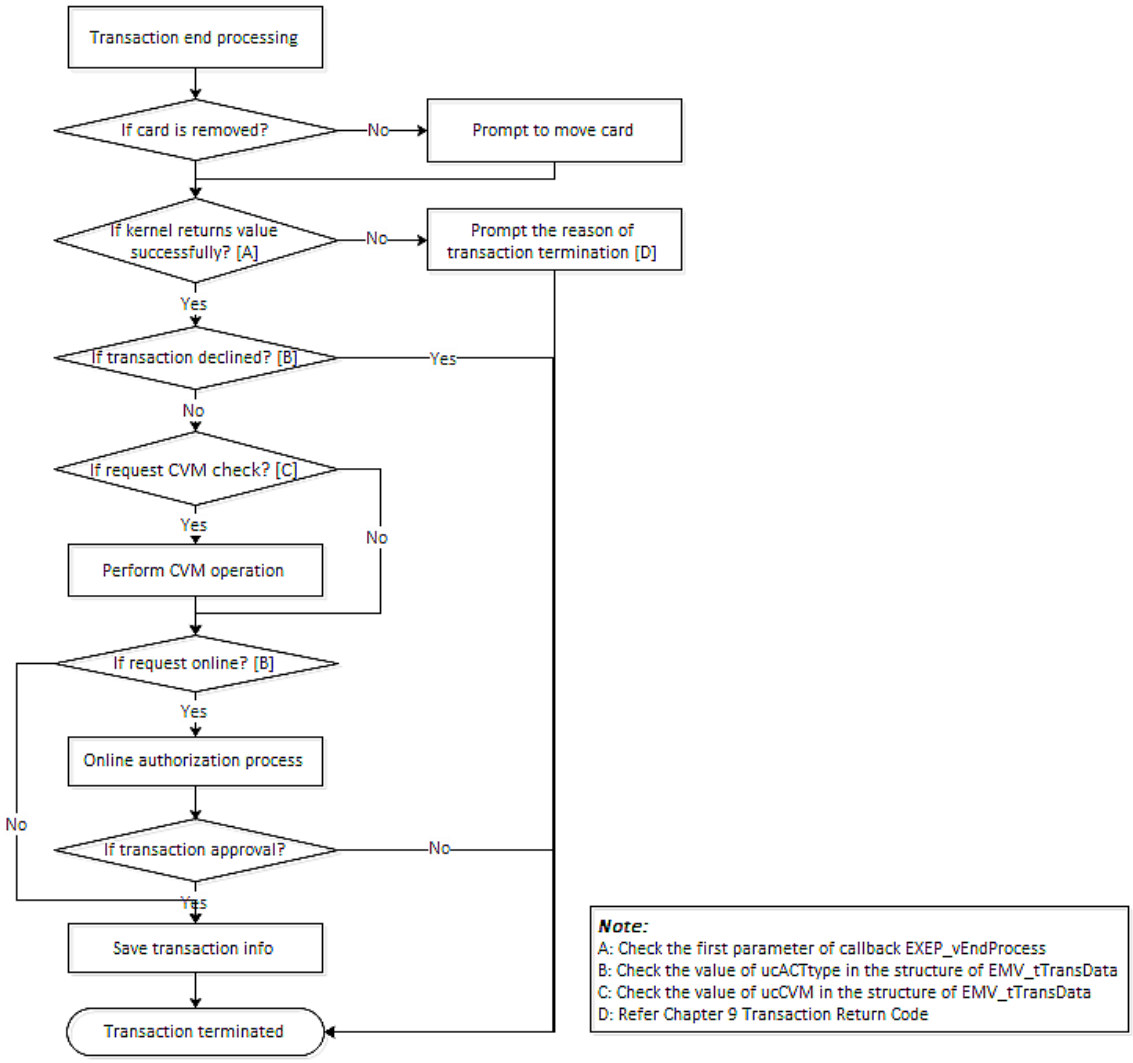
5.12 Query ICC Transaction Log



5.13 Query ICC Offline Balance



5.14 Transaction Finish Process



6. API Specification

6.1 EMV_Kern_uiCreateObject

Prototype	uint EMV_Kern_uiCreateObject(const EMV_Configuration *ptEMVConfiguation, EMV_OBJECT *ptEMVObject)		
Function	Create an EMV kernel Application Object.		
Parameter	**Type**	**Value Range**	**Specification**
ptEMVConfiguation	**In**	Configuration	Refer to EMV_Configuration
ptEMVObject	**Out**	An Application Object pointer	
Return	**Meaning**		
0	Success.		
Other	CreateObject was failed		

Programming Guide:

This API should be executed before any other APIs defined in this document. After this API executing, EMV Kernel will be turned into initialization mode. Any parameter which had been stored in EMV Kernel will be cleared.

6.2 EMV_Kern_uiDestroyObject

Prototype	uint EMV_Kern_uiDestroyObject(EMV_OBJECT tEMVObject)		
Function	Destroy an Application Object which was created successfully before.		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	**In**	A point to an Application Object	After call this function, Application needs to put this parameter to NULL.
Return	**Meaning**		
0	Success.		

Programming Guide:

Destroy a previously created EMV application object.

6.3 EMV_Kern_uiManageAID

Prototype	uint EMV_Kern_uiManageAID(EMV_OBJECT tEMVObject, uchar ucAction, uchar ucAIDLen, const uchar *pauAID, uchar ucPartSlt ,uchar ucKID)		
Function	AID Management		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	**In**		A point to an Application Object
ucAction	**In**	EMV_FLAG_ADD-ADD AID EMV_FLAG_DELETE-Delete AID EMV_FLAG_CLEAR-Clear AID	Action Flag. Refer to Macro Definition
ucAIDLen	**In**	5-16	AID length
pauAID	**In**	Hex	AID
ucPartSlt	**In**	0 or 1	Partial matching: 0-don't support, 1-support
ucKID	**In**	Kernel ID, Support List: EMV_KERNELID_EMV -0x00 EMV_KERNELID_MASTER -0x02 EMV_KERNELID_AMEX -0x04 EMV_KERNELID_JCB -0x05 EMV_KERNELID_DISCOVER -0x06 EMV_KERNELID_PBOC -0x07 EMV_KERNELID_NSICC -0xDA	0- means undefined Kernel ID

Return	Meaning
0	Success.
0xF1	AID length error.
0xF2	AID Pointer is NULL.
0xF3	Action Flag is illegal.
0xF4	AID list space is FULL (Max of AID number is <= 100).
0xF5	Partial matching flag is illegal.
Other value	Other Error.

Programming Guide:

Before any transaction activing, application should add one or more AID. Only when AID is supported by both application and IC card will AID be shown in candidate list.

6.4 EMV_Kern_uiUpdateCAIndexList

Prototype	uint EMV_Kern_uiUpdateCAIndexList(EMV_OBJECT tEMVObject, const uchar *pauRID, const uchar *pauIndexList, uchar ucListLen)		
Function	Updating CA public key		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	**In**	A point to an Application Object	
pauRID	**In**	Hex	RID
pauIndexList	**In**	Hex	CA public key index list
ucListLen	**In**	0~50	Length of index list. If zero, it will clear index list which had been stored in kernel.

Return	Meaning
0	Success.
0x01	(Max of CA public key index is <= 6)FULL.
0XF1	RID is NULL.
0xF2	Value of list length exceeds the max value.
Other value	Other Error.

Programming Guide:

The API is designed for qPBOC and qVSDC. It will reduce the risk of transaction offline failure owing to CA public key absence which will cause IC card offline balance decrease.

6.5 EMV_Kern_uiSetCAPubKey

Prototype	uint EMV_Kern_uiSetCAPubKey(EMV_OBJECT tEMVObject, const EMV_tPKFILESTRU*ptPubKey)		
Function	CA Public key setting for RSA		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	**In**	A point to an Application Object	
ptPubKey	**In**	Structure EMV_tPKFILESTRU pointer or NULL	Refer to EMV_tPKFILESTRU If NULL, Clear stored in kernel.

Return	Meaning
0	Success.
0XF1	Checking Hash value fail.
0xF2	Format error.
Other value	Other Error.

Programming Guide:

The API should be executed when “after read record event” occurs. Application should set CA public key according to RID and key index which is responded by “after read record event”. EMV kernel will keep only one group of public key data which is set at the latest.

6.6 EMV_Kern_uiSetCAPubKey_SM

Prototype	uint EMV_Kern_uiSetCAPubKey_SM(EMV_OBJECT tEMVObject, const EMV_tPKFILESTRU_SM *ptPubKey)		
Function	CA Public key setting for SM		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	**In**	A point to an Application Object	
ptPubKey	**In**	Structure EMV_tPKFILESTRU_SM pointer or NULL	Refer to EMV_tPKFILESTRU_SM If NULL, Clear stored in kernel.
Return	**Meaning**		
0	Success.		
0xF2	Format error.		
Other value	Other Error.		

Programming Guide:

The API should be executed when “after read record event” occurs. Application should set CA public key according to RID and key index which is responded by “after read record event”. EMV kernel will keep only one group of public key data which is set at the latest.

6.7 EMV_Kern_uiManageRecCert

Prototype	uint EMV_Kern_uiManageRecCert(EMV_OBJECT tEMVObject, uchar ucAction, const EMV_tRecCert *ptRecCert)		
Function	Revocate Public Key Certificate Management		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	**In**	A point to an Application Object	
ucAction	**In**	EMV_FLAG_ADD–Add Key EMV_FLAG_DELETE -Delete Key EMV_FLAG_CLEAR -Clear Key	Action Flag. Refer to Macro Definition
ptRecCert	**In**	Structure EMV_tRecCert pointer	Refer to EMV_tRecCert
Return	**Meaning**		
0	Success.		
0x10	FULL.		
0x11	Had existed, reset it.		
0Xe1	Action Flag illegal.		
0xE2	Parameter error.		
Other value	Other Error.		

Programming Guide:

This interface is used to manage the list of CA public key recovery certificates maintained by the EMV kernel, which can store up to 100.

6.8 EMV_Kern_uiManageDOL

Prototype	uint EMV_Kern_uiManageDOL(EMV_OBJECT tEMVObject, uchar ucFlag, uchar ucActon, uchar *paDoL, uchar *pcLen)		
Function	DDOL, TDOL, UDOL Management.		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	**In**	A point to an Application Object	
ucFlag	**In**	1-DDOL 2-TDOL 3-UDOL	DOL Flag
ucAction	**In**	0x01-Set DOL 0x02-Get DOL	Action Flag
paDoL	**In/Out**	Hex	If Set DOL, it is anew DOL. If Get DOL, it will be return DOL.
pcLen	**In/Out**	Length pointer	If Set DOL, it is length of new DOL. If Get DOL, it is length of return DOL

Return	Meaning
0	Success.
0x01	Parameter failed.
0x02	Action Flag illegal.
0X03	DOL Flag illegal.
0x11/0x21/0x31	DOL length error.
0x12/0x22/0x32	DOL Format error.
Other value	Other Error.

Programming Guide:

For example,

Setting DDOL=”9F3704”:

```
Uchar ucLen;
ucLen=3;
uiRet = EMV_Kern_uiManageDOL(1, 1, (uchar*)"\x9F\x37\x04", &ucLen);
```

Get DDOL:

```
Uchar ucLen;
Uchar auBuffer;
ucLen=3;
uiRet = EMV_Kern_uiManageDOL(1, 2, auBuffer, &ucLen);
```

6.9 EMV_Kern_uiSignalInTLV

Prototype	uint EMV_Kern_uiSignalInTLV(EMV_OBJECT tEMVObject, uchar ucSignal, const uchar*pauTLVData, uint uiTLVDataLen)		
Function	Transaction signal entry		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	**In**	A point to an Application Object	
ucSignal	**In**	EMV_SIGNAL_ACT EMV_SIGNAL_NEXT EMV_SIGNAL_STOP EMV_SIGNAL_CLEAN	Signal Type Refer to Macro Definition
pauTLVData	**In**	NULL or Signal parameter pointer	TLV Format parameters for activate a new transaction or response for different event requesting during transaction, refer to Table 4-1.
uiTLVDataLen	**In**	0(If signal parameter is NULL) Length of signal parameter	Length of TLV parameters

Return	Meaning
0	Success.
EMV_RESULT_BUSY	EMV thread is busy, try again later.
0xF001	Signal can't be accepted currently.
0xF0F1	Parameters don't format with TLV.
0xF1XX	Signal type is illegal.
0xF201	Length of parameter error.
Other value	Other Error.

Programming Guide:

```
// Activate a new transaction with only PSE way.
uchar auTLVData[100];
uint uiTLVLen;
uint uiRet;

uiTLVLen=0;
memset(auTLVData, 0, sizeof(auTLVData);
memcpy(auTLVData+uiTLVLen, DEF_TAG_PSE_FLAG, sizeof(DEF_TAG_PSE_FLAG));
uiTLVLen+= sizeof(DEF_TAG_PSE_FLAG);
auTLVData[uiTLVLen++]=0x01;
auTLVData[uiTLVLen++]=0x01;
uiRet =EMV_Kern_uiSignalInTLV(EMV_SIGNAL_ACT, auTLVData, uiTLVLen);
if(uiRet !=0) return fail.
```

6.10 EMV_Kern_uiSetTLV

Prototype	uint EMV_Kern_uiSetTLV(EMV_OBJECT tEMVObject, uchar ucKernelID, const uchar *paTag, uchar ucValueLen, const uchar *paValue)		
Function	Set TLV data element.		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	**In**	A point to an Application Object	
ucKernelID	**In**	Hex	Tag name defined with a prefix such as: EMV_TAG_XXXX (For EMV) C_TAG_XXXX (For PBOC) V_TAG_XXXX (For VISA) M_TAG_XXXX (For MASTER) A_TAG_XXXX (For AMEX) D_TAG_XXXX (For DISCOVER) J_TAG_XXXX (For JCB) DEF_TAG_XXXX (For DEFINE) Refer to Appendix A TAG Dictionary.
paTag	**In**	Hex	Tag
ucValueLen	**In**	0 or 1~ 0xFF	Value length If zero, clear stored one in kernel
paValue	**In**	NULL(only for length is 0) Hex	Value or NULL

Return	Meaning
0	Success.
0x00F0	Parameter error.
0x00F1	ICC Element can't be set.
0x00F2	Kernel ID is illegal.
0x00E1	Format error.
Other value	Other Error.

Programming Guide:

For example:

POS Enter mode setting:

uiRet = EMV_Kern_uiSetTLV(EMV_KERNELID_EMV, (uchar*)"\x9F\x39", 1, (uchar*)"\x91");

Master Card Mobile Support Indicator setting:

uiRet = EMV_Kern_uiSetTLV(EMV_KERNELID_MASTER, (uchar*)"\x9F\x7E", 1, (uchar*)"\x02");

Clear terminal serial number stored in kernel:

uiRet = EMV_Kern_uiSetTLV(EMV_KERNELID_EMV, (uchar*)"\x9F\x1E", 0, NULL);

6.11 EMV_Kern_uiSetTLVList

Prototype	uint EMV_Kern_uiSetTLVList(EMV_OBJECT tEMVObject, uchar ucKernelID, const uchar *pauData, uint uiDataLen)		
Function	TLV list setting		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	**In**	A point to an Application Object	
ucKernelID	**In**	Hex	Tag name defined with a prefix such as: EMV_TAG_XXXX (For EMV) C_TAG_XXXX (For PBOC) V_TAG_XXXX (For VISA) M_TAG_XXXX (For MASTER) A_TAG_XXXX (For AMEX) D_TAG_XXXX (For DISCOVER) J_TAG_XXXX (For JCB) DEF_TAG_XXXX (For DEFINE) Refer to Appendix A TAG Dictionary. **Note:** Tags defined by PBOC or VISA or MASTER, cannot be set into kernel at the same time. · If TLV list includes PBOC Tags, Kernel ID should be EMV_KERNELID_PBOC · If TLV list includes VISA Tags, Kernel ID should be EMV_KERNELID_VISA · If TLV list includes MASTER Tags, Kernel ID should be EMV_KERNELID_MASTER · If TLV list includes AMEX Tags, Kernel ID should be EMV_KERNELID_AMEX · If TLV list includes DISCOVER Tags, Kernel ID should be EMV_KERNELID_DISCOVER · If TLV list includes JCB Tags, Kernel ID should be EMV_KERNELID_JCB · If TLV list includes PBOC and VISA and MASTER Tags, Kernel ID should be EMV_KERNELID_EMV or EMV_KERNELID_DEFINE
pauData	**In**	Hex	TLV list
uiDataLen	**In**	>=1	TLV list length
Return	**Meaning**		
0	Success.		
Other value	Other Error.		

Programming Guide:

For example:

State 1: Only EMV Tags, such as: **EMV_TAG_TM_CNTRYCODE, EMV_TAG_TM_CURCODE**

auTLVlist= **"\x9F\x1A\x02\x08\x40\x5F\x2A\x02\x01\x56"**

ucRet = EMV_Kern_uiSetTLV(**EMV_KERNELID_EMV**, auTLVlist, 10);

State 2: Only DEFINE Tags, such as:
DEF_TAG_PSE_FLAG, DEF_TAG_QUERY_ICCLOG
auTLVlist= "\x**DF**\x **91**\x **81**\x **01**\x **01**\x **00**\x **DF**\x **91**\x **81**\x **03**\x **01**\x **01**"
ucRet = EMV_Kern_uiSetTLV(**EMV_KERNELID_DEFINE**, auTLVlist, 12);

State 3: Including EMV and/ or DEFINE Tags, such as: **EMV_TAG_TM_CNTRYCODE, EMV_TAG_TM_CURCODE, DEF_TAG_PSE_FLAG, DEF_TAG_QUERY_ICCLOG**
ucRet = EMV_Kern_uiSetTLV(**EMV_KERNELID_EMV**,……);

State 4: Including PBOC and EMV and /or DEFINE Tags, such as: **C_TAG_TM_9F7A, EMV_TAG_TM_CURCODE, DEF_TAG_PSE_FLAG, DEF_TAG_QUERY_ICCLOG**
ucRet = EMV_Kern_uiSetTLV(**EMV_KERNELID_PBOC**, ……);

State 5: Including VISA and EMV and /or DEFINE Tags, such as: **V_TAG_TM_9F66, EMV_TAG_TM_CURCODE, DEF_TAG_PSE_FLAG, DEF_TAG_QUERY_ICCLOG**
ucRet = EMV_Kern_uiSetTLV(**EMV_KERNELID_VISA**, ……);

State 6: Including MASTER and EMV and /or DEFINE Tags, such as:
M_TAG_TM_TRANS_LIMIT, EMV_TAG_TM_CURCODE,
DEF_TAG_PSE_FLAG, DEF_TAG_QUERY_ICCLOG
ucRet = EMV_Kern_uiSetTLV(**EMV_KERNELID_MASTER**,……);

6.12 EMV_Kern_uiGetTLV

Prototype	uint EMV_Kern_uiGetTLV(EMV_OBJECT tEMVObject, const uchar *pTag, uchar ucTagLen, uchar *pVal, uint *puiLen)		
Function	Obtain the value of TLV element stored in EMV kernel.		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	**In**	A point to an Application Object	
pTag	**In**	Hex	Tag
ucTagLen	**In**	1~3	Tag Length
pVal	**Out**	Value pointer	Value of the tag
puiLen	**Out**	Length pointer	Length of the value
Return	**Meaning**		
0	Success.		
0x01	Not existed.		
0xE1	Tag is undefined.		
0xE2	Tag is illegal.		
Other value	Other Error.		

Programming Guide:

For example:

Get ICC PAN from EMV Kernel:

ucRet = EMV_Kern_uiGetTLV((uchar*)"\x5A", 1, auValue, &uiLen);

6.13 EMV_kern_uiGetBalance

Prototype	uint EMV_kern_uiGetBalance(EMV_OBJECT tEMVObject, uchar *pauBalance, uchar*pucLen)		
Function	Obtain IC Card offline balance using APDU instruction		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	**In**	A point to an Application Object	
pauBalance	**Out**	Hex	Offline Balance
pucLen	**Out**	Length pointer	Length of Balance
Return	**Meaning**		
0	Success.		
Other value	Other Error.		

Programming Guide:

This API should be executed after Final Selection event has occurred.

6.14 EMV_kern_uiGetDataAPDU

Prototype	uint EMV_kern_uiGetDataAPDU(EMV_OBJECT tEMVObject, uchar ucP1, uchar ucP2, uchar*pauValue, ushort *pusValueLen)		
Function	Obtain ICC element using APDU instruction.		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	**In**	A point to an Application Object	
ucP1	**In**		First Byte of Tag.
ucP2	**In**		Second Byte of Tag.
pauValue	**Out**		The value of Tag.
pusValueLen	**Out**		Length of value.
Return	**Meaning**		
0	Success.		
Other value	Other Error.		

Programming Guide:

This API should be executed after Final Selection event has occurred.

For example:

Obtain ATC from IC Card:

uiRet=EMV_kern_uiGetDataAPDU(0x9F, 0x36, &auATC, &usATCLen);

6.15 EMV_Kern_uiGetICCLog

Prototype	uint EMV_Kern_uiGetICCLog(EMV_OBJECT tEMVObject, const EMV_tSelectAID*ptSelectedAID, uchar *pucLogNum, EMV_tICCLog *ptICCLog)		
Function	Obtain ICC transaction log.		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	**In**	A point to an Application Object	
ptSelectedAID	**In**	Structure EMV_tSelectAID Pointer	Refer to EMV_tSelectAID
pucLogNum	**In**	0-0xFF	If non zero, means max number of ICC log which application expected.
pucLogNum	**Out**	0-0xFF	The number of ICC log actually
ptICCLog	**Out**	Structure EMV_tICCLog list Pointer	Refer to EMV_tICCLog
Return	**Meaning**		
0	Success.		
0x01	IC card does not support this function.		
Other value	Other Error.		

Programming Guide:
This API should be executed after Application Selection event has occurred.

For example:
EMV_tICCLog atICCLog[10];
EMV_tSelectAID tSelectedAID;

tSelectedAID.ucAIDLen = AIDLen;
memcpy(tSelectedAID.auAID,
AID,AIDLen);

memset(atICCLog, 0, sizeof(atICCLog));
ucICCLogNum = 10;//Application expect to obtain 10 logs
ucRet = EMV_Kern_uiGetICCLog(&tSelectedAID,&ucICCLogNum, atICCLog);

6.16 EMV_Kern_uiGetECCLog

Prototype	uint EMV_Kern_uiGetECCLog(EMV_OBJECT tEMVObject, const EMV_tSelectAID*ptSelectedAID, uchar *pucLogNum, EMV_tECCLog *ptECCLog)		
Function	Obtain ICC load transaction log.		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	**In**	A point to an Application Object	
ptSelectedAID	**In**	Structure EMV_tSelectAID Pointer	Refer to EMV_tSelectAID
pucLogNum	**In**	0-0xFF	If it is non zero, it represents the max number of ICC log that application expects to obtain.
pucLogNum	**Out**	0-0xFF	The actually obtained number of ICC log
ptECCLog	**Out**	Structure EMV_tECCLog list Pointer	Refer to EMV_tECCLog
Return	**Meaning**		
0	Success.		
0x01	IC card does not support this function.		
Other value	Other Error.		

Programming Guide:

This API should be executed after Application Selection event has occurred.

It is only for the IC card which supports load transaction. For example, the PBOC 3.0 Ecash card.

Programming example:

```
EMV_tECCLogatECCLog[10];
EMV_tSelectAID tSelectedAID;
Uchar ucECCLogNum;

tSelectedAID.ucAIDLen = AIDLen;
memcpy(tSelectedAID.auAID,
AID,AIDLen);

memset(atECCLog, 0, sizeof(atECCLog));
ucECCLogNum = 10;//Application expect to obtain 10 logs
ucRet = EMV_Kern_uiGetICCLog(&tSelectedAID,&ucECCLogNum, atECCLog);
```

6.17 EMV_Kern_vSwitchDebug

Prototype	void EMV_Kern_vSwitchDebug(EMV_OBJECT tEMVObject, uchar ucMode);		
Function	Deactive or active kernel debug log function.		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	In	A point to an Application Object	
ucMode	In	0 – Deactivate 1 - Activate and obtain debug log after transaction finish 2 - Activate and obtain debug log during the transaction	Refer to Obtain Kernel Debug Log
Return	**Meaning**		
void	Success.		

Programming Guide:

Please refer to Obtain Kernel Debug Log.

6.18 EMV_kern_uiSetHandle

Prototype	uint EMV_kern_uiSetHandle(EMV_OBJECT tEMVObject, uchar ucFlag, EMVHandle hHandle)		
Function	Set device handle into EMV kernel		
Parameter	**Type**	**Value Range**	**Specification**
tEMVObject	In	A point to an Application Object	
ucFlag	In	0xA0 - Contact ICC device 0xA1 - Internal contactless ICC device 0xA2 - External contactless ICC device 0xB1 - PINpad	
hHandle	In	EMVHandle	Device handle
Return	**Meaning**		
uint	Success.		
Other value	Illegal flag.		

7. Structure Definition

7.1 EMV_Configuration

Structure Name	Register Configuration	
Type	Parameter Name	Parameter Specification
EMV_EXPAND_BASEFUN	**tExAPI_BASE**	Refer to EMV_EXPAND_BASEFUN
EMV_EXPAND_INTERFACE	**tExAPI_IFC**	Refer to EMV_EXPAND_INTERFACE
uint	**uiConfigDataLen**	Length of **uiConfigDataLen**
uchar	**auConfigData[600]**	ConfigData
uchar	**ucConfigID**	ConfigID for EMV
uchar	**ucDebugMode**	0 - Deactivate 1 - Activate and obtain debug log after transaction finish. 2 - Activate and obtain debug log during the transaction

7.2 EMV_tPKFILESTRU

Structure Name	RSA CA Public Key	
Type	Parameter Name	Parameter Specification
uchar	**auRid[5];**	RID
uchar	**ucIndex;**	Index
uchar	**ucModLen;**	Length of Mod
uchar	**auMod[256];**	Mod
uchar	**ucExpLen;**	Length of exponent: 1 or 3
uchar	**auExp[3];**	Exponent: \x03 or \x01\x00\x01
uchar	**auExpDate[4];**	Expiry, format as YYYYMMDD, BCD encoded.
uchar	**ucHashFlg;**	Hash verification flag: 1-support, 0-nonsupport
uchar	**auHash[20];**	Hash value.

7.3 EMV_tPKFILESTRU_SM

Structure Name	SM CA Public Key	
Type	Parameter Name	Parameter Specification
uchar	**auRid[5];**	RID
uchar	**ucIndex;**	Index
uchar	**ucKeyLen;**	Length of key
uchar	**auPubKey[256];**	key
uchar	**auExpDate[4];**	Expiry, format as YYYYMMDD, BCD encoded.
GROUP_PARA	**tGroupPara**	Always NULL

7.4 EMV_tSelectAID

Structure Name	AID being selected.	
Type	Parameter Name	Parameter Specification
uchar	**ucAIDLen;**	AID length
uchar	**auAID[EMV_LEN_MAX_AID];**	AID Data

7.5 EMV_tRecCert

Structure Name	Public Key Revocation	
Type	**Parameter Name**	**Parameter Specification**
uchar	**ucIndex;**	CA public key index.
uchar	**auRID[5];**	RID
uchar	**auSN[3];**	Serial number of certification

7.6 EMV_tICCLog

Structure Name	IC Card Transaction log	
Type	**Parameter Name**	**Parameter Specification**
uchar	**ucAmtFlg;**	If Amount Authorized existed? 0-No, 1-Yes
uchar	**auAmount[6];**	Amount, Authorized, BCD encoded.
uchar	**ucAmtOthFlg;**	If Amount other existed? 0-No, 1-Yes
uchar	**auAmountOth[6];**	Amount Other, BCD encoded.
uchar	**ucDateFlg;**	If Transaction Date existed? 0-No, 1-Yes
uchar	**auDate[3];**	Transaction Date, Format as YYMMDD, BCD encoded.
uchar	**ucTimeFlg;**	If Transaction Time existed? 0-No, 1-Yes
uchar	**auTime[3];**	Transaction Time, Format as HHMMSS, BCD encoded.
uchar	**ucCntCFlg;**	If Country Code existed? 0-No, 1-Yes
uchar	**auCntCode[2];**	Country Code(9F1A)
uchar	**ucCurCFlg;**	If Currency Code existed? 0-No, 1-Yes
uchar	**auCurCode[2];**	Currency Code(5F2A)
uchar	**ucATCFlg;**	If ATC existed? 0-No, 1-Yes
uchar	**auATC[2];**	ATC (9F36)
uchar	**ucSevFlg;**	If Service Type existed? 0-No, 1-Yes
uchar	**ucServeType;**	Service Type(9C)
uchar	**ucMchFlg;**	If Merchant Name existed? 0-No, 1-Yes
char	**szMchName[30];**	Merchant Name(9F4E)
uchar	**ucTLVLen;**	TLVlist Length
uchar	**auTLV[256];**	TLVlist, stored other TLV elements which without listed above.

7.7 EMV_tECCLog

Structure Name	IC Card load Transaction log	
Type	**Parameter Name**	**Parameter Specification**
uchar	**auECTag[2];**	Tag name of IC Card offline Balance.
uchar	**auPreValue[6];**	Balance value before modify
uchar	**auAftValue[6];**	Balance value after modify
uchar	**ucDateFlg;**	If Transaction Date existed? 0-No, 1-Yes
uchar	**auDate[3];**	Transaction Date, Format as YYMMDD, BCD encoded.
uchar	**ucTimeFlg;**	If Transaction Time existed? 0-No, 1-Yes
uchar	**auTime[3];**	Transaction Time, Format as HHMMSS, BCD encoded.
uchar	**ucCntCFlg;**	If Country Code existed? 0-No, 1-Yes
uchar	**auCntCode[2];**	Country Code(9F1A)
uchar	**ucATCFlg;**	If ATC existed? 0-No, 1-Yes
uchar	**auATC[2];**	ATC (9F36)
uchar	**ucMchFlg;**	If Merchant Name existed? 0-No, 1-Yes
char	**szMchName[30];**	Merchant Name(9F4E)
uchar	**ucTLVLen;**	TLVlist Length
uchar	**auTLV[50];**	TLVlist, store other TLV elements not listed above.

7.8 EMV_tCandAIDInfo

Structure Name	Candidate AID information.	
Type	**Parameter Name**	**Parameter Specification**
uchar	**ucAIDLen;**	Length of AID
uchar	**auAID[EMV_LEN_MAX_AID];**	AID
uchar	**ucLabelLen;**	Length of Application Label
uchar	**auAppLabel[16];**	Application Label
uchar	**ucAPNLen;**	Length of Application Preferred Name
uchar	**auAPN[16];**	Application Preferred Name
uchar	**ucAPIDFlag;**	If Application priority existed? 0-No, 1-Yes
uchar	**ucAPID;**	Application priority
uchar	**ucLangPrefLen;**	Length of Prefer language list
uchar	**auLangPref[8];**	Prefer language list
uchar	**ucIssCTIndexFlag;**	If Issuer Code Table Index existed? 0-No, 1-Yes
uchar	**ucIssCTIndex;**	Issuer Code Table Index
uchar	**ucKernelIDLen;**	Kernel ID length
uchar	**auKernelID[8];**	kernel ID(9F2A)

7.9 EMV_tAIDCandList

Structure Name	Candidate AID information list	
Type	**Parameter Name**	**Parameter Specification**
uchar	**ucReSelectFlag;**	If Re do Application selection flag? 0-No, 1-Yes
uchar	**ucCandAIDNum;**	The Number of candidate AID
EMV_tCandAIDInfo	**patCandList**	Candidate AID information list.

7.10 EMV_tFinalData

Structure Name	Final selection AID information	
Type	**Parameter Name**	**Parameter Specification**
uchar	**ucAIDLen;**	Final selected AID length
uchar	**auAID[EMV_LEN_MAX_AID];**	Final selected AID Data
uchar	**ucKernelID;**	Macro define Kernel ID, refer to Macro Definition.
uchar	**ucPIDLen;**	Program ID length
uchar	**auPID[16];**	Program ID(9F5A), only for VISA

7.11 EMV_tRecordData

Structure Name	Read ICC Record Return Data	
Type	Parameter Name	Parameter Specification
uchar	**ucAIDLen;**	AID length
uchar	**auAID[EMV_LEN_MAX_AID];**	AID
uchar	**ucPanLen;**	PAN length
uchar	**auPan[10];**	PAN, BCD encoded.
uchar	**ucPanSNFlag;**	If PANSequenceNumber existed? 0-No, 1-Yes
uchar	**ucPanSN;**	PANSequenceNumber
uchar	**auExpiry[4];**	Application Expiration Date, Format as YYYYMMDD BCD encoded.
uchar	**ucAlgorithmID;**	ODA Algorithm ID: 00-RSA 01-SM
uchar	**ucPubKIndex;**	CA Public key index.
uchar	**ucFlowType;**	Flow type of transaction. Refer to Macro Definition.
uchar	**auECIAC[6];**	ECash transaction issuer authorized code.
uchar	**ucSFI11;**	SFI existed flag, 0-No, 1-Yes Only for flow of VISA Paywave2
uchar	**RFULen;**	Length of RFU
uchar	**RFU[256];**	RFU

7.12 EMV_tCVM

Structure Name	Cardholder Verification Method	
Type	Parameter Name	Parameter Specification
uchar	**ucCVM;**	CVM Flag, refer to Macro Definition.
uchar	**ucPINTimes;**	For offline PIN CVM: PIN enter remaining times. If 0 means ICC didn't return.
uchar	**ucCertType;**	For Certificate CVM: Certificate type. (Certificate Type: 00-Identity Card, 01-certificate of officer, 02-passport, 03-entry permit, 04-temporary Identity card, 05- others)
uchar	**ucCertLen;**	For Certificate CVM: length of certificate number
uchar	**auCert[40];**	For Certificate CVM: Certificate number

7.13 EMV_tDisplayMsg

Structure Name	Display Message Information	
Type	Parameter Name	Parameter Specification
uchar	**ucMsgID;**	Message ID, refer to Macro Definition.
uchar	**ucCurrency ;**	Currency flag: 0-RMB, 1-Dollar
uchar	**ucDataLen;**	Message length
uchar	**auData[30];**	Message content

7.14 EMV_tErrorID

Structure Name	EMV Error Location	
Type	**Parameter Name**	**Parameter Specification**
uchar	**ucL1;**	Level 1 error location , refer to Macro Definition.
uchar	**ucL2;**	Level2 error location, refer to Macro Definition.
uchar	**ucL3;**	Level3 error location, refer to Macro Definition.
uchar	**ucMsgID;**	Message ID, refer to Macro Definition.
ushort	**usSW12;**	APDUStatus Code

7.15 EMV_tTransData

Structure Name	Transaction Result Data	
Type	**Parameter Name**	**Parameter Specification**
uchar	**ucACType;**	Application Cryptograph: 0 - Declined 1 - Approved 2 - Online Requested
uchar	**ucCVM;**	CVM Flag, refer to Macro Definition.
uchar	**ucPanLen;**	Length of PAN
uchar	**auPan[10];**	PAN
uchar	**ucPanSnFlag;**	If PanSn existed? 0-No, 1-Yes
uchar	**ucPanSn;**	PAN Sequence Number
uchar	**auExpiry[3];**	Application Expiry
uchar	**ucFlowType;**	Transaction Flow, refer to Macro Definition.
uchar	**ucECIACFlag;**	If ECIAC existed? 0-No, 1-Yes
uchar	**auECIAC[6];**	ECash transaction issuer authorized code.
uchar	**ucBalanceFlag;**	If balance existed? 0-No, 1-Yes
uchar	**auBalance[6];**	ICC Offline Balance, BCD encoded.
uchar	**ucCID;**	Cryptograph Information Data
uchar	**ucSRLength;**	Length of Stript Result, 0-Did't executed issuer script
uchar	**auScriptResult[100];**	Issuer Script Result.
uchar	**ucMSDT1Len;**	Track 1 of MSD1T1Data.
uchar	**auMSDT1Data[200];**	Track 1 of Magnetic Stripe Data
uchar	**ucMSDT2Len;**	Length of MSDT2Data
uchar	**auMSDT2Data[100];**	Track 2 of Magnetic Stripe Data
uint	**uiTLVLen;**	Length of TLV list.
uchar	**auTLVData[300];**	TLV list.

7.16 EMV_EXPAND_BASEFUN

Structure Name	Callback Function List from local driver API	
Type	**Parameter Name**	**Parameter Specification**
Callback	**EX_API_ucICCIO**	Reader executive exchange APDU data with card
Callback	**EX_API_ucListener**	Kernel executive to listen data
Callback	**EX_API_ucGetRandomData**	Kernel get a random data
Callback	**EX_API_ucRSACal**	Executive RSA calculation
Callback	**EX_API_ucCalculateHash**	Executive Hash/SHA1 calculation
Callback	**EX_API_ucCalculateHash_SM**	Executive Hash_SM calculation
Callback	**EX_API_ucVerifySign_SM**	Executive Verify Signature_SM
Callback	**EX_API_ucVerifyOfflinePIN**	Executive Verify Offline PIN

Callback Functions Prototype:

uchar (***EX_API_ucICCIO**)(EMV_tICCDev tICCDevice, uint uiSendLen, const void* pvDataIn, uint *puiRecLen, void* pvDataOut);

uchar (***EX_API_ucListener**)(uchar ucFlag, EMVHandle hMagDevice);//Flag:1-Magstripe 2-Contact Chip

uchar (***EX_API_ucGetRandomData**)(uint uiLen, uchar *pauData);

uchar (***EX_API_ucRSACal**)(uint uiInLen, const uchar *pauInData, const EMV_tPKFILESTRU *ptPubKey, uint *puiOutLen, uchar *pauOutData);

uchar (***EX_API_ucCalculateHash**)(uchar ucHashFlag, uint uiLen, const uchar *pauInData, uchar *pauHash);

uchar (***EX_API_ucCalculateHash_SM**)(uint uiLen, const uchar *pauInData, const EMV_tPKFILESTRU_SM *ptPubKey, uchar *pauHash);

uchar (***EX_API_ucVerifySign_SM**)(const uchar *pauHash, uint uiLen, const uchar *pauInData, const EMV_tPKFILESTRU_SM *ptPubKey);

uchar (***EX_API_ucVerifyOfflinePIN**)(uchar ucFlag, const uchar *pauRandom, const EMV_tPKFILESTRU *ptPubKey, ushort *pauSW12);Flag:0-Plaintext Offline PIN 1-Enciphered offline PIN

Refer to EMV_Configuration.

7.17 EMV_EXPAND_INTERFACE

Structure Name	Callback Function List	
Type	**Parameter Name**	**Parameter Specification**
Callback	**EXEP_ucWaitCard**	Requesting cardholder to show contactless card.
Callback	**EXEP_ucAppSelection**	Requesting cardholder to select IC card application.
Callback	**EXEP_ucFinalSlt**	Supply a time slot for application to adjust parameter according to finial selected AID.
Callback	**EXEP_ucReadRecord**	Supply a time slot for application to deal with messages form card records.
Callback	**EXEP_ucCardHolderVerify**	Requesting cardholder to make verification according to CVM which is indicated by EMV kernel.
Callback	**EXEP_ucOnlineProcess**	Requesting online authorization.
Callback	**EXEP_vEndProcess**	Inform transaction is finished.
Callback	**EXEP_vSendOut**	Send out some messages classify by kernel instruction (refer to Macro Definition) from EMV kernel. (Response is not needed)
Callback	**EXEP_vObtain**	Obtain some messages from application by EMV kernel. (Need response by application)

Callback Functions Prototype:

//Requesting cardholder to show contactless card.
uchar (***EXEP_ucWaitCard**)(uchar **ucFlag**);
//Requesting cardholder to select IC card applicaton.
Uchar (***EXEP_ucAppSelection**)(const **EMV_tAIDCandList** *ptDCandList~~, EMV_tSelectAID *ptSelectedAID~~[4]);
//Supply a time slot for application to adjust parameter according to finial selected AID.
uchar (***EXEP_ucFinalSlt**)(const **EMV_tFinalData** *ptFinalData~~, EMV_tGPOParam *ptDataBack~~[5]);
//Supply a time slot for application to deal with messages form card records.
uchar (***EXEP_ucReadRecord**)(const **EMV_tRecordData** *ptRecordData~~, EMV_tTRManage *ptTRManage~~[6]);
//Requesting cardholder to make verification according to CVM which is indicated by EMV kernel.
uchar (***EXEP_ucCardHolderVerify**)(const **EMV_tCVM** *ptCVM~~, EMV_tCHVData *ptCHVData~~[7]);
//Requesting online authorization.
uchar (***EXEP_ucOnlineProcess**)(const **EMV_tTransData** *ptTransData~~, EMV_tHostData *ptHostData~~[8]);
//Inform transaction is finished.
void (***EXEP_vEndProcess**)(uint **uiResult**, const **EMV_tTransData** *ptTransData);
//Send out some messagesclassifiy by kernel instruction from EMV kernel. (Response not needed)
void (***EXEP_vSendOut**)(uchar ucINS, uint uiDataLen, const uchar *pauData);
//Obtain some messages from application by EMV kernel. (Need response by application)
void (***EXEP_vObtain**)(uchar ucINS, uint uiDataLen, const uchar *pauData);

Specification:

In the callback function list above, when implementing these callback functions, the parameters rendered**Green** are income parameters for the callback functions. Application should store these parameters and return successfully without any redundant operation. Refer to EMV_Configuration.

~~The paramers rendered**Yellow** are only used for reference when application developer programmes.~~[9]

[4]Multiple Application Object have deleted it.
[5]Multiple Application Object have deleted it.
[6]Multiple Application Object have deleted it.
[7]Multiple Application Object have deleted it.
[8]Multiple Application Object have deleted it.

Programming Guide:
EMV_EXPAND_INTERFACE is mandatory for starting a transaction. Each element of this structure is a pointer of call back function. Application will implement these call back functions without any delay and must be non-blocking.

For example, when EMV Kernel generates an event marked with R1, the implement of callback function should be simple and follow these steps:
1. Store the parameters which are returned with event R1.
2. Set the event flag with R1.
3. Return success after end processing

Programming example:

```
ucharEG_ucSignal;                    //event flag
uchar EG_auSignalData[1000];         //event parameter
uint  EG_uiSignalDataLen;            //length of event parameter

//EXEP_ucFinalSltcallback implementation as follow:
uchar EXEP_ucReadRecord(const EMV_tRecordData *ptRecordData, EMV_tTRManage
*ptTRManage
{
        EG_uiSignalDataLen=sizeof(EMV_tRecordData);
        memcpy(EG_auSignalData, ptRecordData, EG_uiSignalDataLen);
        EG_ucSignal=SIGNAL_READRECORD;
        return0;
}
```

8. Obtain Kernel Debug Log

During the development or maintenance, if there is any problem caused by EMV kernel, application developer should attempt to obtain debug log from EMV kernel, and then send the email attached description of issues and the kernel debug log to the EMV expert, which is great helpful for expert to analyze the reasons of issues.

The following indicates the steps of how to obtain debug log from EMV kernel:

1. Application should call the API of **EMV_Kern_vSwitchDebug** to active the function of debug log writing before starting a transaction. There are three kinds of debug mode. Refer to API specification. Choose Debug Mode 1 or 2 will active the function of debug log (Debug Mode 1 – Obtain debug log after transaction finished, Debug Mode 2 – Obtain debug log during transaction process).

2. If you set Debug Mode as “1 (Obtain debug log after transaction finished)”, EMV kernel will send out the debug log after a transaction finished or terminated by callback function of EMV_EXPAND_INTERFACE.EXEP_vSendOut. The application should determine the value of the first parameter “ucINS”. If the value is EMV_INS_DBLOG, the third parameter is the log from the kernel. All messages of debug log are encoded with ASCII. The application should store the messages through writing file, sending them to PC by serial port or other means and receive the ASCII character info through the serial port tool on the PC side to complete the EMV log acquisition.

3. If you set Debug Mode as “2 (Obtain debug log during transaction processing)”, EMV kernel will send out the debug log during the transaction processing by callback function of EMV_EXPAND_INTERFACE.EXEP_vSendOut. The application should determine the value of the first parameter “ucINS”. If the value is EMV_INS_DBLOG, the third parameter is the log from the kernel. All messages of debug log are encoded with ASCII. The application should store the messages through writing file, sending them to PC by serial port or other means and receive the ASCII character info through the serial port tool on the PC side to complete the EMV log acquisition.

Appendix A: TAG Dictionary

A.1 TAG of EMV_KERNELID_EMV

<table>
<tr><th>Kernel ID</th><th colspan="5">EMV_KERNELID_PBOC</th></tr>
<tr><th>Tag Name</th><th>Definition</th><th colspan="4">Description</th></tr>
<tr><td rowspan="4">EMV_TAG_TM_TERMTYPE</td><td rowspan="4">Tag: 9F35
Length: 1
Format: BCD
Kernel: EMV</td><td colspan="4">Terminal Type indicates the environment of the terminal, its communication capability, and its operational control.</td></tr>
<tr><td rowspan="2">Environment</td><td colspan="3">Operator</td></tr>
<tr><td>Financial Institution</td><td>Merchant</td><td>Cardholder</td></tr>
<tr><td>Attendant:
Online only
Online & Offline
Offline only</td><td>
11
12
13</td><td>
21
22
23</td><td>
-
-
-</td></tr>
<tr><td>Self-help:
Online only
Online & Offline
Offline only</td><td>
14
15
16</td><td>
24
25
26</td><td>
34
35
36</td></tr>
<tr><td>EMV_TAG_TM_CAP</td><td>Tag: 9F33
Length: 3
Format: b
Kernel: EMV</td><td colspan="4">Terminal Capabilities
indicates the card data input, CVM, and security capabilities of the terminal.</td></tr>
<tr><td>EMV_TAG_TM_CAP_AD</td><td>Tag: 9F40
Length: 5
Format: b
Kernel: EMV</td><td colspan="4">Additional Terminal Capabilities
indicates the data input and output capabilities of the terminal.</td></tr>
<tr><td>EMV_TAG_TM_CNTRYCODE</td><td>Tag: 9F1A
Length: 2
Format: b
Kernel: EMV</td><td colspan="4">Terminal Country Code
indicates the country of the terminal, represented according to ISO 3166.</td></tr>
<tr><td>EMV_TAG_TM_CURCODE</td><td>Tag: 5F2A
Length: 2
Format: b
Kernel: EMV</td><td colspan="4">Transaction Currency Code
indicates the currency code of the transaction according to ISO 4217.</td></tr>
<tr><td>EMV_TAG_TM_FLOORLMT</td><td>Tag: 9F1B
Length: 4
Format: b
Kernel: EMV</td><td colspan="4">Terminal Floor Limit
indicates the floor limit in the terminal in conjunction with the AID.</td></tr>
<tr><td>EMV_TAG_TM_AID</td><td>Tag: 9F06
Length: 5-16
Format: b
Kernel: EMV</td><td colspan="4">Application Identifier (AID) –
Terminal identifies the application as described in ISO/IEC 7816-5</td></tr>
<tr><td>EMV_TAG_TM_AUTHAMNTN</td><td>Tag: 9F02
Length: 6
Format: BCD
Kernel: EMV</td><td colspan="4">Amount, Authorized (Numeric)
Authorized amount of the transaction (excluding adjustments)</td></tr>
<tr><td>EMV_TAG_TM_OTHERAMNTN</td><td>Tag: 9F03
Length: 6
Format: BCD
Kernel: EMV</td><td colspan="4">Amount, Other (Numeric)
Secondary amount associated with the transaction representing a cashback amount</td></tr>
<tr><td>EMV_TAG_TM_TRANSDATE</td><td>Tag: 9A
Length: 3
Format: BCD
Kernel: EMV</td><td colspan="4">Transaction Date
Local date that the transaction was authorized, format as YYMMDD</td></tr>
<tr><td>EMV_TAG_TM_TRANSTIME</td><td>Tag: 9F21
Length: 3
Format: BCD</td><td colspan="4">Transaction Time
Local time that the transaction was authorized, format as HHMMSS</td></tr>
</table>

	Kernel: EMV	
EMV_TAG_TM_TRSEQCNTR	Tag: 9F41 Length:2- 4 Format: BCD Kernel: EMV	Transaction Sequence Counter Counter maintained by the terminal that is incremented by one for each transaction
EMV_TAG_TM_ARC	Tag: 8A Length: 2 Format: an Kernel: EMV	Authorization Response Code Code that defines the disposition of a message
EMV_TAG_TM_AUTHCODE	Tag: 89 Length:6 Format: b Kernel: EMV	Authorization Code Value generated by the authorization authority for an approved transaction
EMV_TAG_TM_APPVERNO	Tag: 9F09 Length:2 Format: b Kernel: EMV	Version number assigned by the Issuer for the application. For AMEX3.1 this specification the Application Version Number must always be '0001'.
EMV_TAG_TM_TRANSTYPE	Tag: 9C Length:1 Format: b Kernel: EMV	Transaction Type: 0x00: Goods/Service 0x09: CashBack 0x01: Cash 0x20: Refund
EMV_TAG_TM_ACQID	Tag: 9F01 Length: 6 Format:n6-11 Kernel: EMV	Uniquely identifies the acquirer within each payment system.
EMV_TAG_TM_MCHNAMELOC	Tag: 9F4E Length: Var. Format: ans Kernel: EMV	Indicates the name and location of the merchant
EMV_TAG_TM_MCHCATCODE	Tag: 9F15 Length: 2 Format: n4 Kernel: EMV	Classifies the type of business being done by the merchant, represented according to ISO 8583:1993 for Card Acceptor Business Code

A.2 TAG of EMV_KERNELID_PBOC

<table>
<tr><th>Kernel ID</th><th colspan="3">EMV_KERNELID_PBOC</th></tr>
<tr><th>Tag Name</th><th>Definition</th><th colspan="2">Description</th></tr>
<tr><td>C_TAG_TM_9F7A</td><td>Tag: 9F7A
Length: 1
Format: b
Kernel: PBOC</td><td colspan="2">Indicate supporting Ecash or not.
0 - No
1 - Yes</td></tr>
<tr><td>C_TAG_TM_DF69</td><td>Tag: DF69
Length: 1
Format: b
Kernel: PBOC</td><td colspan="2">Indicate supporting PBOC SM algorithm or not.
0 - No
1 - Yes</td></tr>
<tr><td>C_TAG_TM_9F66</td><td>Tag: 9F66
Length: 4
Format: b
Kernel: PBOC</td><td colspan="2">Terminal Transaction Qualifiers
Indicate terminal capabilities, requirements, and preferences to the card.
<table>
<tr><th>Byte</th><th>Bit</th><th>Definition</th></tr>
<tr><td rowspan="4">1</td><td>8</td><td>Reserved</td></tr>
<tr><td>7</td><td>1 - Support contactless debit/credit app
0 - Not support contactless debit/credit app</td></tr>
<tr><td>6</td><td>1 - Support qBOC
0 - Not support qBOC</td></tr>
<tr><td>5</td><td>1 - Support contact debit/credit app</td></tr>
</table></td></tr>
</table>

				0 - Not support contact debit/credit app
			4	1 - Terminal supports offline only 0 - Terminal has online capability
			3	1 - Support online PIN 0 - Not support online PIN
			2	1 - Support signature 0 - Not support signature
			1	Reserved
		2	8	1 - Require online cipher text 0 - No require online cipher text
			7	1 - Require CVM 0 - No require CVM
			6-1	Reserved
		3	8-1	Reserved
		4	8	1 - Terminal supports fDDA with version 01 0 - Terminal supports fDDA with version 00 only
			7-1	Reserved
C_TAG_TM_9F7B	Tag: 9F7B Length: 6 Format: BCD Kernel: PBOC	Ecash Floor Limit If authorized amount is higher than Ecash floor limit, transaction will request online authorization.		
C_TAG_TM_TRANS_LIMIT	Tag: DF8124 Length: 6 Format: BCD Kernel: PBOC	Contactless Transaction Limit If authorized amount is higher than transaction limit, transaction will be terminated.		
C_TAG_TM_CVM_LIMIT	Tag: DF8126 Length: 6 Format: BCD Kernel: PBOC	Contactless CVM Required Limit If authorized amount is higher than CVM limit, transaction will be requested CVM.		
C_TAG_TM_FLOOR_LIMIT	Tag: DF8123 Length: 6 Format: BCD Kernel: PBOC	Contactless Floor Limit If authorized amount higher than floor limit, transaction will be requested online authorization.		
C_TAG_TM_RD_RCP	Tag: DF06 Length: 2 Format: b Kernel: PBOC	Reader Configuration Parameters B1b8: Status Check enabled/disabled (1b=enabled, 0b=disabled) B1b7: Amount, Authorized of Zero Check enabled/disabled (1b=enabled, 0b=disabled) B1b6: Amount, Authorized of Zero Option (1b=Option 1, 0b=Option 2, this bit is only applicable when the reader is online capable) B1b5: Reader Contactless Transaction Limit Check enabled/disabled (1b=enabled, 0b=disabled) B1b4: Reader CVM Required Limit Check enabled/disabled (1b=enabled, 0b=disabled) B1b3: Reader Contactless Floor Limit Check enabled/disabled (1b=enabled, 0b=disabled) B1b2: Exception file enabled/disabled (1b=enabled, 0b=disabled) All other bits are RFU		

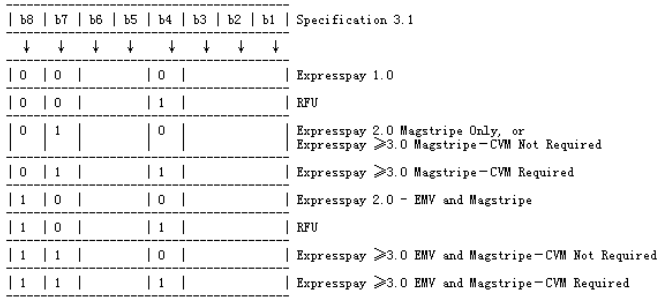
A.3 TAG of EMV_KERNELID_VISA

Kernel ID	EMV_KERNELID_VISA	
Tag Name	**Definition**	**Description**
V_TAG_RD_RCP	Tag: DF06 Length: 2 Format: b Kernel: VISA	Reader Configuration Parameters Byte 1: b8=Status Check b7=Amount, Authorized of Zero Check b6=Amount, Authorized of Zero Option (1b=Option 1, 0b=Option 2, this bit is only applicable when the reader is online-capable) (Discover-Zero Amount Allowed Flag) b5=Reader Contactless Transaction Limit Check b4=Reader CVM Required Limit Check b3=Reader Contactless Floor Limit Check b2=Exception File b1=Certification Revocation List Byte 2: b8-b1=RFU(0)
V_TAG_TM_9F66	Tag: 9F66 Length: 4 Format: b Kernel: VISA	Terminal Transaction Qualifiers indicates terminal capabilities, requirements, and preferences to the card. Byte 1: bit 8: 1=MSD supported bit 7: RFU (0) bit 6: 1=qVSDC supported bit 5: 1=EMV contact ship supported bit 4: 1=Offline-only reader bit 3: 1=Online PIN supported bit 2: 1=Signature supported bit 1: 1=Offline Data Authentication (ODA) for Online Authorizations supported ***Note:*** Readers compliant to this specification set TTQ byte 1 bit 1 to 0b. Byte 2: bit 8: 1=Online cryptogram required bit 7: CVM required bit 6: 1=(Contact Chip) Offline PIN supported bits 5-1: RFU (00000) Byte 3: bit 8: 1=Issuer Update Processing supported bit 7: 1=Mobile functionality supported (Consumer Device CVM) bits 6-1: RFU (00000) Byte 4: RFU ('00')
V_TAG_TM_TRANS_LIMIT	Tag: DF8124 Length: 6 Format: BCD Kernel: VISA	Contactless Transaction Limit If authorized amount is **greater than or equal to** the transaction limit, transaction will be terminated. It is used in conjunction with **V_TAG_RD_RCP** (DF06).
V_TAG_TM_CVM_LIMIT	Tag: DF8126 Length: 6 Format: BCD Kernel: VISA	Contactless CVM Required Limit If authorized amount is **greater than or equal to** the CVM limit, transaction will be requested CVM. It is used in conjunction with **V_TAG_RD_RCP** (DF06).
V_TAG_TM_FLOOR_LIMIT	Tag: DF8123 Length: 6 Format: BCD Kernel: VISA	Contactless Floor Limit If authorized amount is **greater than** the floor limit, transaction will be requested online authorization. It is used in conjunction with **V_TAG_RD_RCP** (DF06). If the Amount, Authorized is greater than either the Reader Contactless Floor Limit or (if the Reader Contactless Floor Limit is not present) the applicable Terminal Floor Limit (tag '9F1B'), then the reader shall indicate Online Cryptogram Required (set TTQ byte 2 bit 8 to 1b).

A.4 TAG of EMV_KERNELID_MASTER

Kernel ID	EMV_KERNELID_MASTER	
Tag Name	**Definition**	**Description**
M _TAG_TM_TRANS_LIMIT	Tag: DF8124 Length: 6 Format: BCD Kernel: MASTER	Contactless Transaction Limit If authorized amount is higher than transaction limit, transaction will be terminated.
M_TAG_TM_TRANS_LIMIT_CDV	Tag: DF8125 Length: 6 Format: BCD Kernel: MASTER	Contactless Transaction Limit while CDV support indicates the transaction amount above which the transaction is not allowed, when on-device cardholder verification is supported.
M _TAG_TM_CVM_LIMIT	Tag: DF8126 Length: 6 Format: BCD Kernel: MASTER	Contactless CVM Required Limit If authorized amount is higher than CVM limit, transaction will be requested CVM.
M _TAG_TM_FLOOR_LIMIT	Tag: DF8123 Length: 6 Format: BCD Kernel: MASTER	Contactless Floor Limit If authorized amount is higher than floor limit, transaction will be requested online authorization.
M_TAG_TM_9F7C	Tag: '9F7C' Length: 20 Format: b Kernel: MASTER	Proprietary merchant data that may be requested by the Card.
M_TAG_TM_9F53	Tag: '9F53' Length: 1 Format: an Kernel: MASTER	This is a data object defined by MasterCard which indicates the type of transaction being performed, and which may be used in card risk management.
M_TAG_TM_9F6D	Tag: '9F6D' Length: 2 Format: b Kernel: MASTER	Version number assigned by the payment system for the specific mag-stripe mode functionality of the Kernel.

A.5 TAG of EMV_KERNELID_AMEX

Kernel ID	EMV_KERNELID_AMEX	
Tag Name	**Definition**	**Description**
A_TAG_TM_9F6D	Tag: 9F6D Length:1 Format: b Kernel: AMEX	Contactless Reader Capabilities  '00' = Expresspay 1.0 '40' = Expresspay 2.0Magstripe only '48' = Expresspay 2.0Magstripe – Mobile CVM Required '80' = Expresspay 2.0EMV and Magstripe 'C0' = ExpresspayMobile (XPM) 'C8' = ExpresspayMobile (XPM) – Mobile CVM Required

A_TAG_TM_9F6E	Tag: 9F6E Length:4 Format: b Kernel: AMEX	Enhanced Contactless Reader Capabilities Byte 1: b8==AEIPS contact mode supported b7==Expresspay Magstripe Mode supported b6==Expresspay EMV full online mode supported b5==Expresspay EMV partial online mode supported b4==Expresspay Mobile Supported b3-b1==RFU Byte 2: b8==Mobile CVM supported b7==Online PIN supported b6==Signature b5==Plaintext Offline PIN b4-b1==RFU Byte 3: b8==Terminal is offline only b7==CVM Required b6-b1==RFU Byte 4: b8-b1==RFU
A_TAG_TM_TRANS_LIMIT	Tag: DF8124 Length:6 Format: BCD Kernel: AMEX	Terminal Contactless Transaction Limit If the Amount, Authorized is **exceeds** the Reader Contactless Transaction Limit, the transaction shall be terminate.
A_TAG_TM_FLOOR_LIMIT	Tag: DF8123 Length:6 Format: BCD Kernel: AMEX	Terminal Contactless Floor Limit If authorized amount is **exceeds** floor limit, transaction will be requested online authorization.
A_TAG_TM_CVM_LIMIT	Tag: DF8126 Length:6 Format: BCD Kernel: AMEX	Terminal CVM Required Limit If the Amount, Authorized is **equal to or exceeds** the Reader CVM Required Limit, then CVM processing is required.
A_TAG_PREAGAIN	Tag: DF8130 Length:1 Format: b Kernel: AMEX	Indicate whether a TryAgain needed or not: 0x00- Not Try Again 0x01- Try Again
A_TAG_TM_IN_CARD_BIN_RANGE	Tag: DF8127 Length:1 Format: b Kernel: AMEX	Indicate if the CardBin in the white CardBin list or not: 0xA0 - Out of CardBin list, Transaction shall be declined 0x00 - In the CardBin list, Transaction continue process

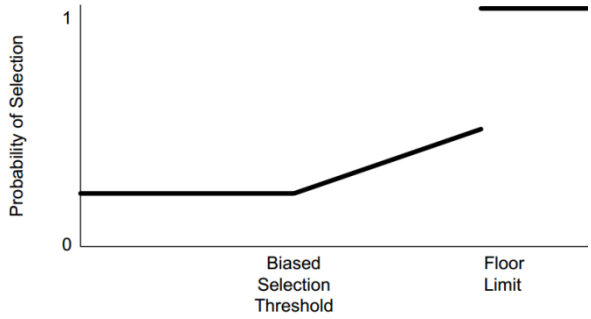
A.6 TAG of EMV_KERNELID_DISCOVER

Kernel ID	EMV_KERNELID_DISCOVER	
Tag Name	**Definition**	**Description**
D_TAG_TM_RD_RCP	Tag: DF06 Length:2 Format: b Kernel: DISCOVER	Reader Configuration Parameters Byte 1: DF Descriptions (1b=Enabled/Present, 0b=Disabled/Not present) b8==Status Check (Value: 1-Support, 0-Not support) b7=='Zero Amount Allowed' flag is present or not (Flag: 1-Present, 0-Not present), using it in conjunction with B1b6 b6==Zero Amount Allowed (Value:1-Allowed, 0-Not allowed) b5==Reader Contactless Transaction Limit Check (Flag: 1-Present, 0-Not present) b4==Reader CVM Required Limit Check (Flag: 1-Present, 0-Not present) b3==Reader Contactless Floor Limit Check (Flag: 1-Present, 0-Not present) b2==Exception File (Flag: 1-Enabled, 0-Disabled) b1==Certification Revocation List (Flag: 1-Enabled, 0-Disabled) Byte 2: b8=='Status Check' flag is present or not (Flag: 1- Present, 0-Not present), using it in conjunction with B1b8 b7-b1==<RFU
D_TAG_TM_9F66	Tag: 9F66 Length:4 Format: b Kernel: DISCOVER	Terminal Transaction Qualifiers indicates terminal capabilities, requirements, and preferences to the card. Byte1: b8==Magnetic stripe mode supported b7==RFU b6==EMV Mode supported b5==EMV contact chip supported b4==Offline-only reader b3==Online PIN supported b2==Signature supported b1==RFU Byte2: b8==Online Cryptogram required b7==CVM Required b6==(Contact chip) Offline PIN supported b5-b1==RFU Byte3: b8==Issuer Update Processing supported b7==Consumer Device CVM supported b6-b1==RFU Byte4: b8-b1==RFU
D_TAG_TM_TRANS_LIMIT	Tag: DF8124 Length:6 Format: BCD Kernel: DISCOVER	Terminal Contactless Transaction Limit If the Amount, Authorized is **equal to or exceeds** the Reader Contactless Transaction Limit, the transaction shall be terminate. It is used in conjunction with **D_TAG_TM_RD_RCP** (DF06).
D_TAG_TM_FLOOR_LIMIT	Tag: DF8123 Length: 6 Format: BCD Kernel: DISCOVER	Terminal Contactless Floor Limit If authorized amount is **exceeds** floor limit, transaction will be requested online authorization. It is used in conjunction with **D_TAG_TM_RD_RCP** (DF06).

D_TAG_TM_CVM_LIMIT	Tag: DF8126 Length:6 Format: BCD Kernel: DISCOVER	Terminal CVM Required Limit If the Amount, Authorized is **equal to or exceeds** the Reader CVM Required Limit, then CVM processing is required. It is used in conjunction with **D_TAG_TM_RD_RCP** (DF06).

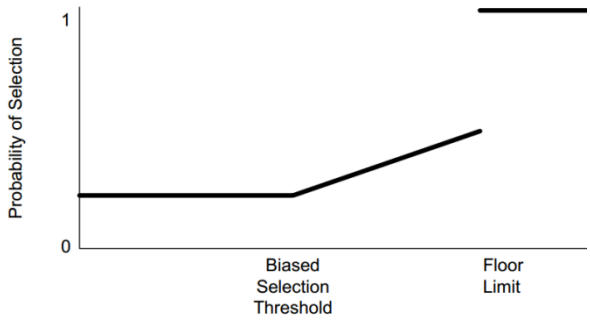
A.7 TAG of EMV_KERNELID_JCB

<table>
<tr><th>Kernel ID</th><th colspan="2">EMV_KERNELID_JCB</th></tr>
<tr><th>Tag Name</th><th>Definition</th><th>Description</th></tr>
<tr><td>DEF_TAG_J_COMB_OPTION</td><td>Tag: DF918404
Length:2
Format: b
Kernel: JCB</td><td>Reader Configuration Parameters
Combination Options Byte 1 (Leftmost)
<table>
<tr><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>Meaning</th></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>RFU</td></tr>
<tr><td></td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td>Status Check supported</td></tr>
<tr><td></td><td></td><td>1</td><td></td><td></td><td></td><td></td><td></td><td>Offline Data Authentication supported</td></tr>
<tr><td></td><td></td><td></td><td>1</td><td></td><td></td><td></td><td></td><td>Exception File Check required[11]</td></tr>
<tr><td></td><td></td><td></td><td></td><td>1</td><td></td><td></td><td></td><td>Random Transaction Selection supported</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td>0</td><td></td><td></td><td>RFU</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td></td><td>EMV Mode Supported[12]</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td>Legacy Mode Supported[13]</td></tr>
</table>
Combination Options Byte 2 (Rightmost)
<table>
<tr><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>Meaning</th></tr>
<tr><td>0x</td><td>0x</td><td>0x</td><td>0x</td><td>0x</td><td>0x</td><td>0x</td><td>0x</td><td>Each bit RFU</td></tr>
</table></td></tr>
<tr><td>DEF_TAG_J_TIP</td><td>Tag: DF918408
Length:3
Format: b
Kernel: JCB</td><td>Terminal Interchange Profile
TIP Byte 1 (Leftmost)
<table>
<tr><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>Meaning</th></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>CVM required by reader / N/A[14]</td></tr>
<tr><td></td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td>Signature supported</td></tr>
<tr><td></td><td></td><td>1</td><td></td><td></td><td></td><td></td><td></td><td>Online PIN supported</td></tr>
<tr><td></td><td></td><td></td><td>1</td><td></td><td></td><td></td><td></td><td>On-Device CVM supported</td></tr>
<tr><td></td><td></td><td></td><td></td><td>0</td><td></td><td></td><td></td><td>RFU</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td>1</td><td></td><td></td><td>Reader is a Transit Reader</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td></td><td>EMV contact chip supported</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td>(Contact Chip) Offline PIN supported</td></tr>
</table>
TIP Byte 2
<table>
<tr><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>Meaning</th></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>Issuer Update supported[15]</td></tr>
<tr><td></td><td>0x</td><td>0x</td><td>0x</td><td>0x</td><td>0x</td><td>0x</td><td>0x</td><td>Each bit RFU</td></tr>
</table>
TIP Byte 3 (Rightmost)
<table>
<tr><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>Meaning</th></tr>
<tr><td>0x</td><td>0x</td><td>0x</td><td>0x</td><td>0x</td><td>0x</td><td>0x</td><td>0x</td><td>Each bit RFU</td></tr>
</table></td></tr>
<tr><td>DEF_TAG_J_TRANS_LIMIT</td><td>Tag: DF918402
Length:6
Format: BCD
Kernel: JCB</td><td>Terminal Contactless Transaction Limit
If the Amount, Authorized is equal to or exceeds the Reader Contactless Transaction Limit, the transaction shall be terminate.</td></tr>
<tr><td>DEF_TAG_J_FLOOR_LIMIT</td><td>Tag: DF918401
Length:6
Format: BCD
Kernel: JCB</td><td>Terminal Contactless Floor Limit
If authorized amount is greater than floor limit, transaction will be requested online authorization.</td></tr>
</table>

DEF_TAG_J_CVM_LIMIT	Tag: DF918403 Length:6 Format: BCD Kernel: JCB	Terminal CVM Required Limit If the Amount, Authorized is equal to or exceeds the Reader CVM Required Limit, then CVM processing is required.
DEF_TAG_J_RS_MAX_PERCENT	Tag: DF918405 Length:1 Format: b Kernel: JCB	Maximum Target Percentage (0-99), larger than Target Percentages.
DEF_TAG_J_RS_TARGET_PERCENT	Tag: DF918406 Length:1 Format: b Kernel: JCB	Target Percentages (0-99)
DEF_TAG_J_RS_THRESH_VALUE	Tag: DF918409 Length:4 Format: b Kernel: JCB	BiasedSelectionThreshold, if authorized amount is higher than BiasedSelectionThreshold, the chance of online authorization will be increased. The relationship between online authorization probability and floor limit and BiasedSelectionThreshold as follow figure:  Reference: 10.6.2 Random Transaction Selection of EMV specification Book 3
DEF_TAG_J_ONLINE_TWOPRE	Tag: DF918410 Length:1 Format: b Kernel: JCB	Online Issuer update in two ways: Present and hold card or Two present card. By judging the online callback function transaction data EMV_tTransData, it is determined whether the label DEF_TAG_J_ONLINE_TWOPRE exists in the label list auTLVData. If yes, perform a re-swinging operation; otherwise, hold the card.

A.8 TAG of EMV_KERNELID_DEFINE

Kernel ID	EMV_KERNELID_DEFINE	
Tag Name	**Definition**	**Description**
DEF_TAG_PSE_FLAG	Tag: DF918101 Length: 1 Format :b Kernel: All	Application selection way: 0 - PSE selection first and then AID selection 1 - Only PSE selection 2 - Only AID selection 3 - Only PPSE selection 4 - PPSE First, AID selection Second (Discover ZIP Mode)
DEF_TAG_GAC_CONTROL	Tag: DF918102 Length: 1 Format: b Kernel: All	Generate AC control flag 0 - Normal 1 - Force offline 2 - Force online 3 - Force decline
DEF_TAG_QUERY_ICCLOG	Tag: DF918103 Length: 1 Format: b Kernel: All	Indicate if start a ICC log query transaction: 0 - No 1 - Yes
DEF_TAG_SERVICE_TYPE	Tag: DF918104 Length: 1 Format: b Kernel: All	Service Type, refer to Macro Definition.
DEF_TAG_START_RECOVERY	Tag: DF918105 Length:1 Format: b Kernel: PBOC	Indicate if start a recovery transaction for a torn transaction: 0 –Normal transaction process 1 - Current torn recovery process 2 - All torn recovery process
DEF_TAG_PAN_IN_BLACK	Tag: DF918106 Length:1 Format: b Kernel: All	Indicate if the PAN listed in exception file: 0 - No 1 - Yes
DEF_TAG_ACCUMULATE_AMOUNT	Tag: DF918107 Length:6 Format: BCD Kernel: All	Serial offline approved amount accumulate for the same PAN.
DEF_TAG_CHV_STATUS	Tag: DF918108 Length:1 Format: b Kernel: All	Indicate the status of operation of cardholder verification: 0 - Non execute 1 - Has executed 2 - Executed fail 3 - Exceed pin try times 4 - Bypass pin
DEF_TAG_ONLINE_STATUS	Tag: DF918109 Length:1 Format: b Kernel: All	Indicate the status of online communication: 0 - Online success 1 - Online failed
DEF_TAG_AUTHORIZE_FLAG	Tag: DF91810A Length:1 Format: b Kernel: All	Indicate the result online authorization: 0 - Declined 1 - Approved
DEF_TAG_HOST_TLV DATA	Tag: DF91810B Length: Var Format: b Kernel: All	Hostresponse data while online authorization, such as issuer script, which is TLV format.
DEF_TAG_RAND_SLT_THRESHOLD	Tag: DF91810C Length:6 Format: BCD Kernel: All	BiasedSelectionThreshold, if authorized amount is higher than BiasedSelectionThreshold, the chance of online authorization will be increase. The relationship between online authorization probability and Floor limit and BiasedSelectionThreshold as follow figure:

		 Reference: 10.6.2 Random Transaction Selection of EMV specification Book 3
DEF_TAG_RAND_SLT_PER	Tag: DF91810D Length:1 Format: b Kernel: All	Target Percentages (0-99)
DEF_TAG_RAND_SLT_MAXPER	Tag: DF91810E Length:1 Format: b Kernel: All	Maximum Target Percentage (0-99), larger than Target Percentages.
DEF_TAG_TAC_DEFAULT	Tag: DF918110 Length:5 Format: b Kernel: All	Terminal Action Code, For Default
DEF_TAG_TAC_DECLINE	Tag: DF918111 Length:5 Format: b Kernel: All	Terminal Action Code, For Denial
DEF_TAG_TAC_ONLINE	Tag: DF918112 Length:5 Format: b Kernel: All	Terminal Action Code, For Online
DEF_TAG_BALANCE_BEF_GAC	Tag: DF918113 Length:6 Format: BCD Kernel: PBOC\MASTER	ICC balance before Generate AC command.
DEF_TAG_BALANCE_AFT_GAC	Tag: DF918114 Length:6 Format: BCD Kernel: PBOC\MASTER	ICC balance After Generate AC command.
DEF_TAG_TORN_SUPPORT	Tag: DF918115 Length:1 Format: b Kernel: ALL	Indicate application support torn transaction or not. 0 - No 1 - Yes
DEF_TAG_M_TRANS_MODE	Tag: DF918201 Length:1 Format: b Kernel: MASTER	Kernel Configuration for PayPass flow 0 - Mag-stripe Flow only 1 - EMV Flow only 2 - Mag-Stripe and EMV Both
DEF_TAG_M_BALANCE_SUP	Tag: DF918202 Length:1 Format: b Kernel: MASTER	Balance obtain flag, before or after the GENERATE AC: 0 – don't support any one 1 - only support before Generate AC 2 - only support after Generate AC 3 - Support Both
DEF_TAG_M_CDV_SUP	Tag: DF918204 Length:1 Format: b Kernel: MASTER	Card holder device CVM verification for PayPass 0-unsupport 1-support
DEF_TAG_M_REQ_CVM	Tag: DF918205 Length:1	CVM Capability – CVM Required Indicates the CVM capability of the Terminal and Reader when

<table>
<tr><td></td><td>Format: b
Kernel: MASTER</td><td>the transaction amount is greater than the Reader CVMRequired Limit.
CVM Capability – CVM Required
<table><tr><td>b8</td><td>Plaintext PIN for ICC verification</td></tr><tr><td>b7</td><td>Enciphered PIN for online verification</td></tr><tr><td>b6</td><td>Signature (paper)</td></tr><tr><td>b5</td><td>Enciphered PIN for offline verification</td></tr><tr><td>b4</td><td>No CVM required</td></tr><tr><td>b3-1</td><td>RFU</td></tr></table></td></tr>
<tr><td>DEF_TAG_M_REQ_NOCVM</td><td>Tag: DF918206
Length:1
Format: b
Kernel: MASTER</td><td>CVM Capability – No CVM Required
indicates the CVM capability of the terminal and Reader when the transaction amount is less than or equal to the Reader CVM Required Limit.
CVM Capability – No CVM Required
<table><tr><td>b8</td><td>Plaintext PIN for ICC verification</td></tr><tr><td>b7</td><td>Enciphered PIN for online verification</td></tr><tr><td>b6</td><td>Signature (paper)</td></tr><tr><td>b5</td><td>Enciphered PIN for offline verification</td></tr><tr><td>b4</td><td>No CVM required</td></tr><tr><td>b3-1</td><td>RFU</td></tr></table></td></tr>
<tr><td>DEF_TAG_M_MAG_REQ_CVM</td><td>Tag: DF918207
Length:1
Format: b
Kernel: MASTER</td><td>Mag-stripe CVM Capability – CVM Required
indicates the CVM capability of the terminal in the case of a mag-stripe mode transaction when the Amount authorized (Numeric) is greater than the Reader CVM Required Limit
Mag-stripe CVM Capability – CVM Required
<table><tr><td>b8-5</td><td>CVM<table><tr><td>0000: NO CVM</td></tr><tr><td>0001: OBTAIN SIGNATURE</td></tr><tr><td>0010: ONLINE PIN</td></tr><tr><td>1111: N/A</td></tr><tr><td>Other values: RFU</td></tr></table></td></tr><tr><td>b4-1</td><td>RFU</td></tr></table></td></tr>
<tr><td>DEF_TAG_M_MAG_REQ_NOCVM</td><td>Tag: DF918208
Length:1
Format: b
Kernel: MASTER</td><td>Mag-stripe CVM Capability – No CVM Required
indicates the CVM capability of the terminal in the case of a mag-stripe mode transaction when the Amount authorized (Numeric) is less than or equal to the Reader CVM Required Limit.
Mag-stripe CVM Capability – No CVM Required
<table><tr><td>b8-5</td><td>CVM<table><tr><td>0000: NO CVM</td></tr><tr><td>0001: OBTAIN SIGNATURE</td></tr><tr><td>0010: ONLINE PIN</td></tr><tr><td>1111: N/A</td></tr><tr><td>Other values: RFU</td></tr></table></td></tr><tr><td>b4-1</td><td>RFU</td></tr></table></td></tr>
<tr><td>DEF_TAG_M_MSG_HOLDTIME</td><td>Tag: DF918209
Length:3
Format: BCD
Kernel: MASTER</td><td>Message Hold Time
indicates the default delay for the processing of the next MSG signal. The Message Hold Time is an integer in units of 100ms.</td></tr>
<tr><td>DEF_TAG_M_RF_HOLDTIME</td><td>Tag: DF91820A
Length:1
Format: b
Kernel: MASTER</td><td>Hold Time Value
indicates the time that the field is to be turned off after the transaction is completed if requested to do so by the cardholder device. The Hold Time Value is in units of 100ms.</td></tr>
<tr><td>DEF_TAG_D_ISSUERSCRIPT_EXCUTIVE</td><td>Tag: DF918215
Length:1</td><td>Indicate whether an Issuer Script Process needed or not:
0x00- Not Needed (Default)</td></tr>
</table>

	Format: b Kernel: DISCOVER	0x01- Needed It should be set to 0x01 when Online Process the Issuer return 71 or 72 issuer script.
DEF_TAG_RESELECT_CONDITION	Tag: DF928103 Length:5 Format: b Kernel: All	Indicate which condition will cause failure on current AID and request application selection on next AID: Byte 1: b8-Final selection response with SW12 unequal 9000 b7-Final selection response with the data didn't include PDOL b6- Final selection response with the data include PDOL but didn't request 9F66. Byte2-Byte5 RFU
DEF_TAG_PPSE_6A82_TURNTO_AIDLIST	Tag: DF918155 Length:1 Format: b Kernel: All	If PPSE response 6A82, turn to AID list selection. 0x01-Indicates turn to AID List Selection when PPSE returns 6A82 0x00-Do nothing
DEF_TAG_CHECK_CAPK_INDEXLIST	Tag: DF928104 Length:1 Format: b Kernel: PBOC/VISA	When card returned approval, Whether check CAPK index or not before read the last record. 0x00-Not check 0x01-check
DEF_TAG_CTL_AS_CB_FLAG	Tag: DF928105 Length:1 Format: b Kernel: All	Whether execute app select callback function or not in contactless transaction 0x00-Not execute 0x01-Execute
DEF_TAG_OBTAIN_FLAG	Tag: DF928101 Length:1 Format: b Kernel: All	Indicate whether 'OBTAIN' callback function is to be output after ending of a step 0x00-No need 0x01-Executive OBTAIN after end of ODA step
DEF_TAG_OBTAIN_RETURN_DATA	Tag: DF928102 Length:1 Format: b Kernel: All	The output data of OBTAIN callback function after ending Of ODA step 0x00-ODA success 0x01-ODA failed
DEF_TAG_ONLIE_ODA_FAIL_FLOW_TYPE	Tag: DF918163 Length:1 Format: b Kernel: All	Indicate the transaction is online or declined when online ODA execute fail. 0x00-Online 0x01-Declined
DEF_TAG_RESULT_CODE	Tag: DF91810F Length:4 Format: b Kernel: All	Addition Result Code, refer to Appendix C.
DEF_TAG_ALLOW_DUP_ICC_SAMEVALUE	Tag: DF918140 Length:1 Format: b Kernel: All	Indicate that if allow ICC element repeated with same value: 0-Not allow 1-Allow (Except the following tags: 5A,57,5F24,9F07,9F32,8C come from PBOC requirements)
DEF_TAG_ERROR_TYPE	Tag: DF91815A Length:1 Format: b Kernel: All	Indicate which error type when the transaction was terminated. 0x01-Try Another Interface 0x02-Try Another Payment 0x03-Use Another Card
DEF_TAG_TRYGAIN_SPECIAL_RETCODE	Tag: DF918162 Length:1 Format: b Kernel: All	Indicate Try Again using Special Return Code, not using WaitCard (Flag) Event Callback Function 0x01: Using Special Return Code 0x00 or not set: Using on WaitCard (Flag) Event, Flag=**EMV_FLAG_EXECUTE_CDCVM**
DEF_TAG_FORCE_ONLINE_ALL	Tag: DF91815C Length:1 Format: b Kernel: All	Indicate all of the transaction independent kernel ID will request online authorization (only situation TC turn to ARQC). 0x00 - Never force online 0x01 - Force online with TVR Byte 4 bit 4 set. 0x02 - Force online without TVR set.

Appendix B: Macro Definition

Macro Name	Value	Indication
Signal Type		
EMV_SIGNAL_ACT	0x01	Activate a new transaction
EMV_SIGNAL_NEXT	0x02	Inform kernel to continue process
EMV_SIGNAL_STOP	0xF0	Terminate the transaction
EMV_SIGNAL_CLEAN	0xC0	Kernel housekeeping
Kernel ID		
EMV_KERNELID_EMV	0x00	EMV Contact
EMV_KERNELID_EMVCTLess	0x01	EMV Contactless
EMV_KERNELID_MASTER	0x02	Master Card
EMV_KERNELID_VISA	0x03	VISA
EMV_KERNELID_AMEX	0x04	AMEX
EMV_KERNELID_JCB	0x05	JCB
EMV_KERNELID_DISCOVER	0x06	DISCOVER
EMV_KERNELID_PBOC	0x07	PBOC
EMV_KERNELID_RUPAY	0x0D	RUPAY
EMV_KERNELID_NSICC	0xDA	Indonesian
EMV_KERNELID_DEFINE	0xDE	Kernel Defined
Flow Type		
EMV_FLOWTYPE_EMV	0x01	EMV/PBOC Contact Level2
EMV_FLOWTYPE_ECASH	0x03	PBOC ECash
EMV_FLOWTYPE_QPBOC	0x11	qPBOC
EMV_FLOWTYPE_PBOC_CTLESS	0x12	PBOC Contactless Level2
EMV_FLOWTYPE_MSD	0x13	VISA MSD
EMV_FLOWTYPE_MSD_LEGACY	0x14	VISA MSD Legacy
EMV_FLOWTYPE_QVSDC	0x21	VISA qVSDC
EMV_FLOWTYPE_WAVE2	0x22	VISA PayWave2
EMV_FLOWTYPE_M_CHIP	0x31	MASTER Card PayPass-Chip
EMV_FLOWTYPE_M_STRIPE	0x32	MASTER Card PayPass-Stripe
EMV_FLOWTYPE_J_EMV	0x33	JCB EMV Mode
EMV_FLOWTYPE_J_MAG	0x34	JCB Magstripe Mode
EMV_FLOWTYPE_J_LEGACY	0x35	JCB Legacy mode
EMV_FLOWTYPE_A_XP2_MS	0x41	AMEX ExpressPay Card Magstripe Mode
EMV_FLOWTYPE_A_XP2_EMV	0x42	AMEX ExpressPay Card EMV Mode
EMV_FLOWTYPE_A_XPM_MS	0x43	AMEX ExpressPay Mobile Magstripe Mode
EMV_FLOWTYPE_A_XPM_EMV	0x44	AMEX ExpressPay Mobile EMV Mode
EMV_FLOWTYPE_D_DPAS_MS	0x51	Discover D-PAS Magstripe Mode
EMV_FLOWTYPE_D_DPAS_EMV	0x52	Discover D-PAS EMV Mode
EMV_FLOWTYPE_D_ZIP	0x53	Discover ZIP Mode
EMV_FLOWTYPE_R_LEGACY	0x61	RuPay EMV Legacy Mode
EMV_FLOWTYPE_R_NONLEGACY	0x62	RuPay EMV Non-Legacy Mode
Service Type		
EMV_SERVETYPE_GOOD	0x00	Goods
EMV_SERVETYPE_SERVICE	0x00	Service
EMV_SERVETYPE_CASH	0x01	Cash
EMV_SERVETYPE_CASHBACK	0x09	Cashback

EMV_SERVETYPE_REFUND	0x20	Refund
CVM Flag		
EMV_CVMFLAG_NOCVM	0x00	No CVM Verification
EMV_CVMFLAG_OFFLINEPIN	0x01	Offline PIN
EMV_CVMFLAG_ONLINEPIN	0x02	Online PIN
EMV_CVMFLAG_SIGNATURE	0x03	Signature
EMV_CVMFLAG_OLPIN_SIGN	0x04	Online PIN and Signature
EMV_CVMFLAG_CDV	0x05	Consumer Device Verification (qVSDC/qPBOC)
EMV_CVMFLAG_CCV	0x06	Confirmation Code Verified (PayPass)
EMV_CVMFLAG_CERTIFICATE	0x11	Certificate Verification (Certificate Type: 00-Identity Card, 01-certificate of officer, 02-passport, 03-entry permit, 04-temporary Identity card, 05- others)
EMV_CVMFLAG_ECASHPIN	0x21	ECash Change PIN
AC Type		
EMV_ACTION_AAC	0x00	Declined
EMV_ACTION_TC	0x01	Approved
EMV_ACTION_ARQC	0x02	Request Online Authorization
Action Flag		
EMV_FLAG_ADD	0x01	Add
EMV_FLAG_DELETE	0x02	Delete
EMV_FLAG_CLEAR	0x03	Clear
Kernel Instruction		
EMV_INS_SET_TORN	0xA1	Send out torn transaction record
EMV_INS_DEL_TORN	0xD1	Inform application to delete torn records expired
EMV_INS_DISPLAY	0xA2	Send out display information, format reference structure: EMV tDisplayMsg
EMV_INS_TLVDATA	0xA3	Send out TLV data
EMV_INS_CLOSERF	0xA4	Inform application to close contactless interface
EMV_INS_DBLOG	0xDB	Send out debug logs
EMV_INS_GET_TORN	0xB1	Obtain torn records specified by kernel
EMV_INS_SEND_DISC	0xE1	Send out discretionary data (PayPass)
EMV_INS_APPSELECT_DATA	0xC1	Send out application selection data
EMV_INS_SET_FAIL_WATER	0xA5	Inform app save failed transaction water
EMV_INS_DEL_FAIL_WATER	0xA6	Inform app delete fail transaction water
Message ID		
EMV_MSGID_CARD_READ_OK	0x17	Read card finished
EMV_MSGID_TRY_AGAIN	0x21	Try again
EMV_MSGID_APPROVED	0x03	Transaction approved
EMV_MSGID_APPROVED_SIGN	0x1A	Transaction approved and requesting signature
EMV_MSGID_DECLINED	0x07	Transaction declined
EMV_MSGID_ERR_OTH_CARD	0x1C	Transaction error, please try other card.
EMV_MSGID_INSERT_CARD	0x1D	Please insert IC card
EMV_MSGID_SEE_PHONE	0x20	Please check cell phone.
EMV_MSGID_AUTH_WAIT	0x1B	Waiting authorization
EMV_MSGID_CLEAR_DISPLAY	0x1E	Clear screen display
EMV_MSGID_ICC_ACCOUNT	0x1F	ICC Account
EMV_MSGID_PCII	0xF1	Display message according PCII
EMV_MSGID_UNMATCH_PAN	0xF2	In current tron recovery process, unmatch pan

EMV_MSGID_READ_CARD_FAIL	0xF3	In All flash tron recovery process, read card fail
EMV_MSGID_ONLINE_ODA_RESULT	0xF4	In Online ODA process, the result of ODA
Error Location		
EMV_L1_ERR_TIMEOUT	0x01	ICC APDU communicate time out
EMV_L1_ERR_TRANSMISSION	0x02	ICC APDU transmission error
EMV_L1_ERR_PROTOCOL	0x03	ICC transmit protocol error
EMV_L2_ERR_ICC_DATA_MISS	0x01	ICC Data missing
EMV_L2_ERR_CAM_FAILED	0x02	CAM Fail
EMV_L2_ERR_ICC_STATUS	0x03	APDU Status Error
EMV_L2_ERR_PARSING	0x04	ICC data parsing error.
EMV_L2_ERR_MAX_EXCEEDED	0x05	Exceed max limit.
EMV_L2_ERR_ICC_DATA	0x06	ICC data error.
EMV_L2_ERR_MAG_NOT_SUP	0x07	Don’t support magnetic.
EMV_L2_ERR_NO_PPSE	0x08	Don’t support PPSE.
EMV_L2_ERR_PPSE_FAULT	0x09	PPSE fault
EMV_L2_ERR_NO_CAND_AID	0x0A	Candidate AID list is empty.
EMV_L2_ERR_TERM_DATA	0x0F	Terminate parameter error.
EMV_L3_ERR_TIMEOUT	0x01	Time out
EMV_L3_ERR_STOP	0x02	Transaction being terminated
EMV_L3_ERR_AMOUNT	0x03	Amount is absent
WaitCard Callback Parameter Flag		
EMV_FLAG_NORMAL	0x00	Normal transaction wait card flag
EMV_FLAG_SHOW_CARD_AGAIN	0x01	Show card again
EMV_FLAG_ISS_SCRIPT_UPDATE	0x02	Show card again, don't show amount
EMV_FLAG_EXECUTE_CDCVM	0x03	CDCVM wasn't executed, prompt related information and show card again

Appendix C: Transaction Return Code

Transaction Return Code (TRC) indicates the current transaction's status and error type, which comes out from the first parameter of the “transaction finished event callback” API “EXEP_vEndProcess”，as follows:

void (*EXEP_vEndProcess)(uint **uiResult**, const EMV_tTransData *ptTransData)

TRC is divided into three categories:

1. Transaction finished normally

The only return code that identifies the normal end of the transaction:

EMV_RESULT_NORMAL

2. Common Error Code

Coded with 2 bytes, format: 0xEX XX

There are some common features for this kind of error type, which appears frequently and is independent with transaction flow. These kind of error codes have been defined as macro for being recognized easily by developer.

3. Other Error Code

Coded with 4 bytes, format: 0xFN XX YY ZZ

If the fourth byte ZZ, has a value 0xFX，then still need an additional result code for help to position an error exactly. This additional result code had been defined as Tag DEF_TAG_RESULT_CODE（DF91810F）.

Programming example:

```
void EXEP_vEndProcess(uint uiResult, const EMV_tTransData *ptTransData)
{
    Uint uiLen;
    Uchar auResultCode[10];
        switch(uiResult)
        {
        case EMV_RESULT_NORMAL:
                Log("Transaction finisedand the status of transaction refer
EMV_tTransData");Break;
        case EMV_RESULT_NOAPP:
                Log("There is no application match both by terminal and card");Break;
        case EMV_RESULT_NOPUBKEY:
                Log("Quick pass transaction, CA public key missing, transaction
terminate");Break;
        case EMV_RESULT_EXPIRY:
                Log("Card application had expired, transaction terminate.");Break;
        case EMV_RESULT_STOP:
                Log("Transacton had been stoped manually ");Break;
        case EMV_RESULT_REPOWERICC:
```

```
                Log("Please repower the card and try again");Break;
        case EMV_RESULT_REFUSESERVICE:
                Log("IC card refuse the service.");Break;
        case EMV_RESULT_CARDLOCK:
                Log("IC card had been locked(SW=6A81)");Break;
        case EMV_RESULT_APPLOCK:
                Log("Card application had been locked(SW=6283)");Break;
        case EMV_RESULT_EXCEED_CTLMT:
                Log("Amount larger than contactless limit, transaction terminate");Break;
        case EMV_RESULT_APDU_ERROR:
                Log("APDU error, Please try again");Break;
        case EMV_RESULT_APDU_STATUS_ERROR:
                Log("APDU status error , please try again.");Break;
   default:
                uiLen=0;
                memset(auResultCode, 0, sizeof(auResultCode))
                if ((uiResult&0xF0)==0xF0)
                {//need to get additional result codd from DEF_TAG_RESULT_CODE.
                EMV_Kern_uiGetTLV(tEMVObject,DEF_TAG_RESULT_CODE,4,auResultCode,
&uiLen);
                }
                if(!uiLen)
                {       Log("Other error , transaction terminate. Transaction result
code:%04X",uiResult); }
                else
                {Log("Other error , transaction terminate. Transaction result
         code:%04X+%04X",uiResult, auResultCode); }
                break;
        }
}
```

Transaction Finished Normal		
EMV_RESULT_NORMAL	0x0000	Transaction completed
Common Error Code		
EMV_RESULT_BUSY	0xEE01	EMV Kernel is busy. Please wait and try again later.
EMV_RESULT_NOAPP	0xEE02	No AID matched between terminal and card.
EMV_RESULT_NOPUBKEY	0xEE03	For qPBOC and qVSDC transaction, Terminal can't supply CA public key Indicated by the card.
EMV_RESULT_EXPIRY	0xEE04	ICC application is expiry.
EMV_RESULT_FLASHCARD	0xEE06	Flash card has occurred.
EMV_RESULT_STOP	0xEE07	EMV kernel is terminated by application.
EMV_RESULT_REPOWERICC	0xEE08	Communicate between ICC and Terminal.
EMV_RESULT_REFUSESERVICE	0xEE09	ICC refused to provide the service.
EMV_RESULT_CARDLOCK	0xEE0A	Card locked (SW=6A81).
EMV_RESULT_APPLOCK	0xEE0B	Application locked (SW=6283).
EMV_RESULT_EXCEED_CTLMT	0xEE0C	Amount exceed contactless limit.

EMV_RESULT_APDU_ERROR	0xEE0D	APDU Exchange error.
EMV_RESULT_APDU_STATUS_ERROR	0xEE0E	Card response APDU SW12!=9000
EMV_RESULT_ALL_FLASH_CARD	0xEE0F	Return all flash card to be deal with process.
Other undefined codes	XXXXXXXX	Other reason caused the transaction terminated.
Other Error Code		
FN XX YY ZZ	If value of ZZ >=0xF0, need additional result code DF91810F.	
DEF_TAG_RESULT_CODE	Additional result code, Hex format, 4 bytes.	