



beyond  
payment

# **MasterCard PayPass 3.x Kernel overview and migration help**

**ICO-OPE-00230-V1-EN**

# Contents

<b>1. Introduction</b>	<b>4</b>
1.1. Purpose	4
1.2. What is PayPass 3?	4
1.3. Related documents	4
1.4. Software architecture	5
<b>2. Major points helping migrating from PayPass 2.x</b>	<b>6</b>
2.1. New functions	6
2.2. Input tags	6
2.2.1. Terminal capabilities	6
2.2.2. Contactless Transaction Limit	7
2.3. Obsolete tags	7
2.3.1. M/Stripe indicator	7
2.3.2. Transaction CVM	8
2.3.3. Transaction outcome	8
2.3.4. Lists of Application Version Numbers	8
2.3.5. Refund transaction flow	8
2.4. Output tags	8
2.4.1. Overview	8
2.4.2. Outcome Parameter Set	9
2.4.3. Data Record	9
2.4.4. Discretionary Data (DD)	10
2.4.5. Error Indication (EI)	11
2.4.6. User Interface Request Data (UIRD)	12
2.4.7. Examples	12
2.4.7.1. M/Chip transaction is offline approved, with Signature	12
2.4.7.2. M/Chip transaction is sent online for authorisation, no CVM	13
2.4.7.3. M/Chip transaction is terminated because of a card error	13
2.4.8. Status returned	13
2.5. Other steps where UIRD is returned	14
2.5.1. Remove Card sequence	14
2.5.2. Message for double tap	14

2.6. Transaction flow	15
2.7. Other changes	15
2.7.1. Concept of “Empty Tags”	15
2.7.2. Proprietary tags	16
2.7.3. Default kernel database values	16
<b>3. New features overview</b>	<b>17</b>
3.1. Mobile transactions	17
3.2. Torn recovery	17
3.3. Data Exchange	18
3.4. Data Storage	19
3.4.1. SDS (Standalone Data Storage)	19
3.4.2. IDS (Integrated Data Storage)	19

## 1. Introduction

---

### 1.1. Purpose

---

The aim of this document is to give a first overview of the PayPass 3 new kernel. This would be useful for to understand what the new features are, and would help developers to migrate from PayPass 2.x to PayPass 3.x. However, this document does not constitute a migration note (as the API between PayPass 2.x and PayPass 3.x are really different, especially on the transaction flow and the data used, and a detailed reading of the kernel documentation is really important).

### 1.2. What is PayPass 3?

---

PayPass 3 is the latest release of MasterCard PayPass Terminal specifications. It allows to leverage the new features of MasterCard M/Chip Advance new card generation which includes MasterCard virtual cards embedded in mobile phones.

This new specification provides three new major features:

- PayPass Mobile for high value transactions (including double tap), to use the PASS code (or m-PIN) on the mobile phone as CVM above PayPass limit.
- Data Storage acceptance to access the open services to be hosted by the new MasterCard M/Chip Advance cards.
- Recovery of Torn transactions to secure the payment processing of pre-paid/pre-authorised PayPass cards with Offline balance.

### 1.3. Related documents

---

The kernel present in this package is compliant with:

- PayPass M/Chip – Reader Card Application Interface Specification, version 3.0.1, April 2012.
- PayPass M/Chip – Reader Card Application Interface Specification - Errata, version 1.4, June 2012.

The payment application developers now have to use the following specification:

- PayPass – M/Chip Requirements (December 2011).

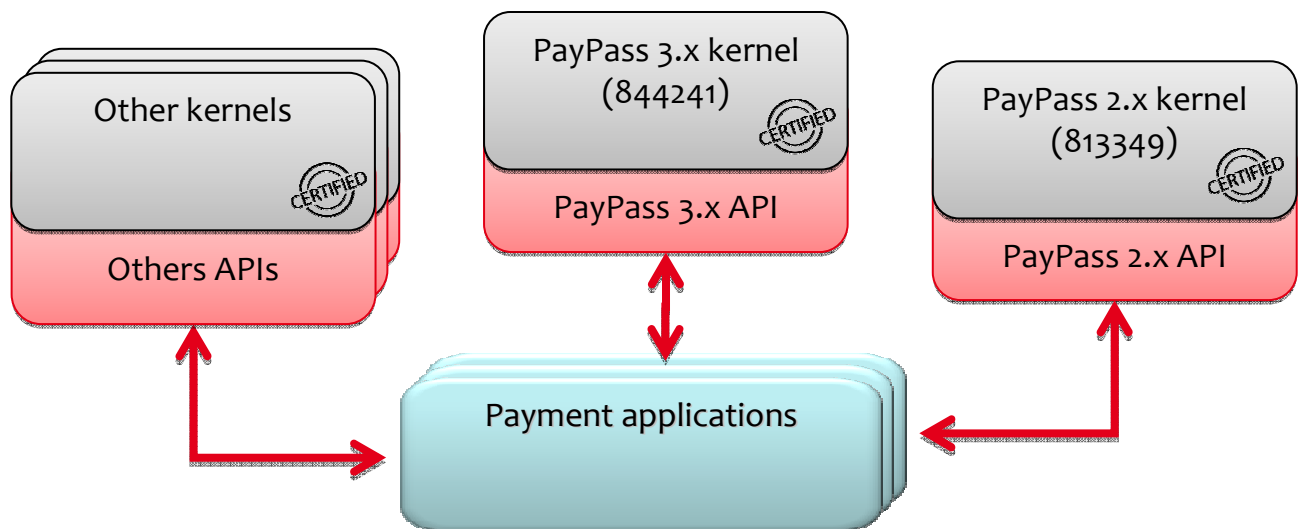
This replaces the PayPass M/Chip – Acquirer Implementation Requirements, previously used.

## 1.4. Software architecture

The PayPass 3.x kernel is not considered as an evolution of the PayPass 2.x, but rather as a new kernel. So, we decided to create a new kernel identifier and to use a new application type for this kernel:

- Identifier is **844241**
- Application type is **ACD1**

So, this means that PayPass 3.x kernel can cohabitate with a PayPass 2.x within the same terminal.



### IMPORTANT

The PayPass 2.x and PayPass 3.x kernels are very different and an application developed with PayPass 2.x is not compliant with the new PayPass 3.x kernel. The necessary work to rewrite a PayPass 3.x application will also depend on the features you need to implement.

## 2. Major points helping migrating from PayPass 2.x



As said previously, PayPass 2.x and PayPass 3.x kernel are very different, so the aim of this section is to help developers to migrate from PayPass 2.x to PayPass 3.x. However, the information provided here cannot be exhaustive, so you shall refer to the detailed documentation for your developments (a dedicated sample custom application is available to highlight the interface to the PayPass 3.x kernel).

### 2.1. New functions

The new PayPass 3 kernel API functions are now named **PayPass3\_xxx()**. All the functions have been kept, and one new function has been created for specific new feature (Data Exchange). Here is the list of the available functions:

- `int PayPass3_DoTransaction (T_SHARED_DATA_STRUCT *pDataStruct)`
- `int PayPass3_ResumeTransaction (T_SHARED_DATA_STRUCT *pDataStruct)`
- `int PayPass3_GetAllData (T_SHARED_DATA_STRUCT *pDataStruct)`
- `int PayPass3_GetData (T_SHARED_DATA_STRUCT *pDataStruct)`
- `int PayPass3_LoadData (T_SHARED_DATA_STRUCT *pDataStruct)`
- `int PayPass3_Cancel (void)`
- `int PayPass3_DebugManagement (T_SHARED_DATA_STRUCT *pDataStruct)`
- `int PayPass3_Clear (void)`
- `int PayPass3_GetInfos (T_SHARED_DATA_STRUCT *pDataStruct)`
- `int PayPass3_DETSignal (T_SHARED_DATA_STRUCT *pDataStruct)`

As you can see, this is very similar to the PayPass 2.x API. However, the PayPass3\_DoTransaction data (input and output) are different from PayPass 2.x.

### 2.2. Input tags

This section indicates what the major changes are with the input data provided with the PayPass3\_DoTransaction() function, between PayPass 2.x and PayPass 3.x. As said before, this is not exhaustive and some additional data can be necessary with PayPass 3.

#### 2.2.1. Terminal capabilities

With PayPass 2.x kernel, two values of terminal capabilities are provided, on 3 bytes:

- **TAG\_PAYPASS\_TERMINAL\_CAPABILITIES\_CVM\_REQ**: Terminal Capabilities to be used if amount is above the CVM limit.
- **TAG\_PAYPASS\_TERMINAL\_CAPABILITIES\_NO\_CVM\_REQ**: Terminal Capabilities to be used if amount is below the CVM limit.

Note that these two tags are now obsolete with PayPass 3.x and will not be managed if provided to the kernel.

With the PayPass 3.x kernel: each byte is provided separately, and the kernel builds the Terminal Capabilities, depending on the CVM limit and the card type:

- Byte 1 = `TAG_PAYPASS_CARD_DATA_INPUT_CAPABILITY`
- Byte 2 =
  - `TAG_PAYPASS_MCHIP_CVM_CAPABILITY_CVM_REQUIRED`: if M/Chip card and amount above CVM Limit. Format is the same as EMV Terminal Capabilities Byte 2.
  - `TAG_PAYPASS_MCHIP_CVM_CAPABILITY_CVM_NOT_REQUIRED`: if M/Chip card and amount below CVM Limit. Format is the same as EMV Terminal Capabilities Byte 2.
  - `TAG_PAYPASS_MSTRIPE_CVM_CAPABILITY_CVM_REQUIRED`: if M/Stripe card and amount above CVM Limit. The format is different from the EMV one, and it is possible to only indicate one CVM (please refer to the tag documentation for more information).
  - `TAG_PAYPASS_MSTRIPE_CVM_CAPABILITY_CVM_NOT_REQUIRED`: if M/Stripe card and amount below CVM Limit. The format is different from the EMV one, and it is possible to only indicate one CVM (please refer to the tag documentation for more information).
- Byte 3 = `TAG_PAYPASS_SECURITY_CAPABILITY`

### 2.2.2. Contactless Transaction Limit

With the PayPass 2.x kernel, this limit is given to the application selection module and analysed during the pre processing. If the amount is greater than this limit, then the AID cannot be chosen for final selection.

With the PayPass 3.x kernel, this is different. This limit is not provided to the application selection and this is the role of the kernel to compare the amount to this limit. Moreover, there exist two different Contactless Transaction Limits:

- `TAG_PAYPASS_CLESS_TRANSACTION_LIMIT_NO_DCV` for cards.
- `TAG_PAYPASS_CLESS_TRANSACTION_LIMIT_DCV` for mobiles.

Depending on the cardholder device presented to the reader (card or mobile), the kernel will automatically set the `TAG_EP_CLESS_TRANSACTION_LIMIT` tag with one of the values defined before.

So, it is mandatory to provide `TAG_PAYPASS_CLESS_TRANSACTION_LIMIT_NO_DCV` and `TAG_PAYPASS_CLESS_TRANSACTION_LIMIT_DCV` to the kernel, and no provide `TAG_EP_CLESS_TRANSACTION_LIMIT` (as this one is automatically managed by the kernel).

## 2.3. Obsolete tags

Here is the list of tags that are used with PayPass 2.x kernel but that are obsolete with PayPass 3.x kernel. For more details about this, please refer to the documentation. Note that obsolete input tags will be ignored and not stored if provided by an application and obsolete output tags will no longer be returned by the kernel.

### 2.3.1. M/Stripe indicator

`TAG_PAYPASS_MSTRIPE_INDICATOR` and `TAG_PAYPASS_INT_FORCE_MSTRIPE_FLOW` are obsolete and now covered by the new `TAG_PAYPASS_KERNEL_CONFIGURATION`.

### 2.3.2. Transaction CVM

`TAG_PAYPASS_TRANSACTION_CVM` is obsolete and now covered by `TAG_PAYPASS_OUTCOME_PARAMETER_SET` (“CVM” field).

### 2.3.3. Transaction outcome

`TAG_PAYPASS_TRANSACTION_OUTCOME` is obsolete and now covered by `TAG_PAYPASS_OUTCOME_PARAMETER_SET` (“Status” field).

### 2.3.4. Lists of Application Version Numbers

`TAG_PAYPASS_INT_MSTRIPE_TERMINAL_AVN_LIST` is obsolete and there is no any AVN checking for PayPass 3.x in the M/Stripe flow. However, you have to indicate an application version number directly through tag `TAG_PAYPASS_MSTRIPE_APPLI_VERSION_NUMBER_TERM` (as it is part of the Data Record returned by the kernel).

`TAG_PAYPASS_INT_MCHIP_TERMINAL_AVN_LIST` is now obsolete and it is not possible to provide a list of application version number to the kernel. They are now replaced by `TAG_EMV_APPLI_VERSION_NUMBER_TERM` (EMV tag), directly indicating the AVN to be used for the M/Chip transaction.

### 2.3.5. Refund transaction flow

With PayPass 3.x, there is no more specific transaction flow for refund. So tag `TAG_PAYPASS_INT_USE_REFUND_FLOW` is now obsolete and this is the role of the application to correctly provide parameters to meet the refund requirements (force AAC on Generate AC, etc).

## 2.4. Output tags

With the new PayPass 3.x kernel, the data returned by the `PayPass3_DoTransaction()` function are different from the ones returned by the PayPass 2.x kernel. This section describes the major data returned by the PayPass 3.x kernel.

### 2.4.1. Overview

The PayPass 3.x kernel now returns the following data in the `PayPass3_DoTransaction()` function:

- **Outcome Parameter Set (OPS)**: Indicates the transaction outcome, the CVM to apply (if applicable), with the different data present. This is always present on every kernel output.
- **Data Record (DR)**: List of TLV encoded data objects returned with the Outcome Parameter Set on the completion of transaction processing.
- **Discretionary Data (DD)**:
  - List of Kernel-specific data objects sent to the Terminal as a separate field.
  - This also contains the **Error Indication (EI)** containing information regarding the nature of the error that has been encountered during the transaction processing.
- Up to two **User interface Request Data (UIRD)**: Information of a GUI state.



### 2.4.2. Outcome Parameter Set

This tag is a structure containing the following fields:

- **“Status”**: Indicates the status of the transaction
  - N/A: Initial setting.
  - APPROVED: offline approved.
  - DECLINED: offline declined.
  - ONLINE REQUEST: to be sent online for authorisation.
  - END APPLICATION: transaction is terminated (could be a restart in some cases).
  - SELECT NEXT: Continue the transaction and select the next available AID.
  - TRY ANOTHER INTERFACE: conducted over another interface.
  - TRY AGAIN: silent restart of the transaction.
- **“Start”**: Indicates how the transaction needs to be restarted, if necessary
  - N/A: No restart required (initial setting).
  - START B: Transaction needs to be restarted from the beginning (i.e. as a « retry »).
  - START C: The transaction needs to be restarted using the next available AID, if available.
- **“CVM”**: indicates the CVM to be used
  - N/A: CVM not applicable (initial setting).
  - NO CVM: No CVM required (almost all the cases).
  - OBTAIN SIGNATURE: Signature required.
  - ONLINE PIN: Online PIN required.
  - CONFIRMATION CODE VERIFIED: Mobile code has been required.
- **“Presence”**: indicates what are the other data present
  - UIR on outcome: always set to ‘No’.
  - UIR on restart: generally set to ‘Yes’ when START B. If set to ‘Yes’, an UIRD is present.
  - Data Record: when set to ‘Yes’, the Data Record is returned by the kernel as well.
  - Discretionary Data: always set to ‘Yes’ as DD is always provided.
  - Receipt: indicates if a receipt has to be printed or not.
- **“Field Off Request”**: Indicates if the field needs to be powered off, with the indicated delay (to be performed by the application after the power off call)
  - N/A: field off not required.
  - X: Hold time in units of 100 ms.
- **“Online Response Data”**: always set to “N/A”.
- **“Alternate Interface Preference”**: always set to “N/A”.
- **“Removal Timeout”**: always set to “0”.

### 2.4.3. Data Record

This is a constructed tag, containing the transaction data. The Data Record content is different for M/Chip and M/Stripe transactions. Note that the data below are only returned if they are present in the kernel database.

M/Stripe Data Record Content	Tags
Application Label	TAG_EMV_APPLICATION_LABEL
Application Preferred Name	TAG_EMV_APPLI_PREFERRED_NAME
DF Name	TAG_EMV_DF_NAME
Interface Device Serial Number	TAG_EMV_IFD_SERIAL_NUMBER
Issuer Code Table Index	TAG_EMV_ISSUER_CODE_TABLE_INDEX
Mag-Stripe application version number (Reader)	TAG_PAYPASS_MSTRIPE_APPLI_VERSION_NUMBER_TERM
Track 1 Data	TAG_PAYPASS_TRACK1_DATA
Track 2 Data	TAG_PAYPASS_TRACK2_DATA

M/Chip Data Record Content	Tags
Amount, Authorized (numeric)	TAG_EMV_AMOUNT_AUTH_NUM
Amount, Other (numeric)	TAG_EMV_AMOUNT_OTHER_NUM
Application Cryptogram	TAG_EMV_APPLICATION_CRYPTOGRAM
Application Expiration Date	TAG_EMV_APPLI_EXPIRATION_DATE
Application Interchange Profile	TAG_EMV_AIP
Application Label	TAG_EMV_APPLICATION_LABEL
Application PAN	TAG_EMV_APPLI_PAN
Application PAN Sequence Number	TAG_EMV_APPLI_PAN_SEQUENCE_NUMBER
Application Preferred Name	TAG_EMV_APPLI_PREFERED_NAME
Application Transaction Counter	TAG_EMV_ATC
Application Version Number (Reader)	TAG_EMV_APPLI_VERSION_NUMBER_TERM
Cryptogram Information Data	TAG_EMV_CRYPTOGRAM_INFO_DATA
CVM Results	TAG_EMV_CVM_RESULTS
DF Name	TAG_EMV_DF_NAME
Interface Device Serial Number	TAG_EMV_IFD_SERIAL_NUMBER
Issuer Application Data	TAG_EMV_ISSUER_APPLI_DATA
Issuer Code Table Index	TAG_EMV_ISSUER_CODE_TABLE_INDEX
Terminal Capabilities	TAG_EMV_TERMINAL_CAPABILITIES
Terminal Country Code	TAG_EMV_TERMINAL_COUNTRY_CODE
Terminal Type	TAG_EMV_TERMINAL_TYPE
Terminal Verification Results	TAG_EMV_TVR
Track 2 Equivalent Data	TAG_EMV_TRACK_2_EQU_DATA
Transaction Category Code	TAG_PAYPASS_TRANSACTION_CATEGORY_CODE
Transaction Currency Code	TAG_EMV_TRANSACTION_CURRENCY_CODE
Transaction Date	TAG_EMV_TRANSACTION_DATE
Transaction Type	TAG_EMV_TRANSACTION_TYPE
Unpredictable Number	TAG_EMV_UNPREDICTABLE_NUMBER

#### 2.4.4. Discretionary Data (DD)

This is a constructed tag, containing the discretionary data. The DR content is different for M/Chip and M/Stripe transactions. Note that the data below are only returned if they are present and not empty in the kernel database.

M/Stripe Discretionary Data Content	Tags
DD Card (Track 1)	TAG_PAYPASS_DD_CARD_TRACK1
DD Card (Track 2)	TAG_PAYPASS_DD_CARD_TRACK2
Error indication	TAG_PAYPASS_ERROR_INDICATION
Third Party Data	TAG_PAYPASS_THIRD_PARTY_DATA

M/Chip Discretionary Data Content	Tags
Application Currency Code	TAG_EMV_APPLI_CURRENCY_CODE
Balance Read After Gen AC	TAG_PAYPASS_BALANCE_READ_BEFORE_GENAC
Balance Read Before Gen AC	TAG_PAYPASS_BALANCE_READ_AFTER_GENAC
DS Summary 3	TAG_PAYPASS_INT_DS_SUMMARY_3
DS Summary Status	TAG_PAYPASS_INT_DS_SUMMARY_STATUS
Error indication	TAG_PAYPASS_ERROR_INDICATION
Post-Gen AC Put Data Status	TAG_PAYPASS_POST_GENAC_PUT_DATA_STATUS
Pre-Gen AC Put Data Status	TAG_PAYPASS_PRE_GENAC_PUT_DATA_STATUS
Third Party Data	TAG_PAYPASS_THIRD_PARTY_DATA
Torn Record	TAG_PAYPASS_TORN_RECORD Or TAG_PAYPASS_OLD_TORN_RECORD

### 2.4.5. Error Indication (EI)

Note that this tag is systematically returned in the Discretionary Data structure. This tag contains the following fields:

- “L1 Error”
  - OK: No problem detected.
  - TIME OUT ERROR: Card has been removed from the field or did not respond on time.
  - TRANSMISSION ERROR: Transmission error occurred.
  - PROTOCOL ERROR: Protocol error occurred.
- “L2 Error”
  - OK: No problem occurred.
  - CARD DATA MISSING: Card data missing detected.
  - CAM FAILED: CAM error detected (generally due to ODA error).
  - STATUS BYTES: Card returned an unexpected SW.
  - PARSING ERROR: A parsing error occurred.
  - MAX LIMIT EXCEEDED: The amount exceeds « Contactless Transaction limit ».
  - CARD DATA ERROR: A card data error has been detected.
  - MAGSTRIPE NOT SUPPORTED: M/Stripe transaction is not supported.
  - IDS READ ERROR: An error occurred during the IDS read process.
  - IDS WRITE ERROR: An error occurred during the IDS write process.
  - IDS DATA ERROR: An IDS data error has been detected.
  - IDS NO MATCHING AC: The AC does not match during the IDS process.
  - TERMINAL DATA ERROR: A terminal data error has been detected.
- “L3 Error”
  - OK: No problem detected.
  - TIME OUT: A timeout occurred (related to Data Exchange).
  - STOP: Transaction has been stopped following a `PayPass3_Cancel()`.
  - AMOUNT NOT PRESENT: Transaction cannot be completed because amount is not present.
- “Status Word”: SW returned by the card if L2 = STATUS BYTE.
- “Msg on error”: Indicates the message to be displayed.

## 2.4.6. User Interface Request Data (UIRD)

This is a tag representing a request from the kernel to display a GUI state (including the LEDs). This contains the following information:

- “**Message Identifier**”: message to be displayed
  - **CARD READ OK**: Card has been read and can be removed.
  - **TRY AGAIN**: Cardholder needs to present his card again.
  - **APPROVED**: Transaction is approved.
  - **APPROVED – SIGN**: Transaction is approved and a signature is required.
  - **DECLINED**: Transaction is declined.
  - **ERROR - OTHER CARD**: Transaction is terminated because of an error.
  - **INSERT CARD**: Card needs to be inserted.
  - **SEE PHONE**: Check the phone for instructions.
  - **AUTHORISING – PLEASE WAIT**: Transaction is sent online for authorisation.
  - **CLEAR DISPLAY**: Display shall be cleared.
- “**Status**”: audio/visual indicators status
  - **NOT READY**: not ready to read a card.
  - **READY TO READ**: Ready to read a card (first LED turned on).
  - **CARD READ SUCCESSFULLY**: card removal sequence.
- “**Hold Time**”: time the message needs to be displayed.
- “**Language preference**”: language preference provided by the card.
- “**Value qualifier**”: nature of the additional data to be displayed
  - **NONE**: no additional value to be displayed.
  - **AMOUNT**: Amount needs to be displayed.
  - **BALANCE**: Card balance needs to be displayed.
- “**Value**”: value for « value qualifier ».
- “**Currency Code**”: to display the currency associated to « Value Qualifier ».

## 2.4.7. Examples

### 2.4.7.1. M/Chip transaction is offline approved, with Signature

<b>OUTCOME PARAMETER SET</b> Status: APPROVED Start: N/A Online Response Data: N/A CVM: OBTAIN SIGNATURE Presence: UIR on Outcome: No. UIR on Restart: No. Data Record: Yes. Discretionary Data: Yes. Receipt: Yes. Alternate Interface Preference: N/A. Field Off Request: N/A. Removal Timeout: 0.	<b>USER INTERFACE REQUEST DATA</b> Message ID: APPROVED - SIGN Status: NOT READY Hold Time: 000013 Language Pref: 0000000000000000 Value Qualifier: NONE Value: 000000000000 Currency Code: 0000
<b>DISCRETIONARY DATA</b> Error indication. All the transaction discretionary data...	<b>ERROR INDICATION (in Discretionary Data)</b> L1 Error: OK L2 Error: OK L3 Error: OK SW: 0000 Message on error: N/A
	<b>DATA RECORD</b> All the transaction data for record...

### 2.4.7.2. M/Chip transaction is sent online for authorisation, no CVM

#### OUTCOME PARAMETER SET

Status: ONLINE REQUEST  
 Start: N/A  
 Online Response Data: N/A  
 CVM: NO CVM  
 Presence:  
     UIR on Outcome: No.  
     UIR on Restart: No.  
     Data Record: Yes.  
     Discretionary Data: Yes.  
     Receipt: No.  
 Alternate Interface Preference: N/A.  
 Field Off Request: N/A.  
 Removal Timeout: 0.

#### DISCRETIONARY DATA

Error indication.  
 All the transaction discretionary data...

#### USER INTERFACE REQUEST DATA

Message ID: AUTHORISING PLEASE WAIT  
 Status: NOT READY  
 Hold Time: 000000  
 Language Pref: 656E000000000000  
 Value Qualifier: NONE  
 Value: 000000000000  
 Currency Code: 0000

#### ERROR INDICATION (in Discretionary Data)

L1 Error: OK  
 L2 Error: OK  
 L3 Error: OK  
 SW: 0000  
 Message on error: N/A

#### DATA RECORD

All the transaction data for record...

### 2.4.7.3. M/Chip transaction is terminated because of a card error

#### OUTCOME PARAMETER SET

Status: END APPLICATION  
 Start: N/A  
 Online Response Data: N/A  
 CVM: N/A  
 Presence:  
     UIR on Outcome: No.  
     UIR on Restart: No.  
     Data Record: No.  
     Discretionary Data: Yes.  
     Receipt: No.  
 Alternate Interface Preference: N/A.  
 Field Off Request: N/A.  
 Removal Timeout: 0.

#### DISCRETIONARY DATA

Error indication.  
 All the transaction discretionary data...

#### USER INTERFACE REQUEST DATA

Message ID: ERROR OTHER CARD  
 Status: NOT READY  
 Hold Time: 000013  
 Language Pref: 0000000000000000  
 Value Qualifier: NONE  
 Value: 000000000000  
 Currency Code: 0000

#### ERROR INDICATION (in Discretionary Data)

L1 Error: OK  
 L2 Error: CARD DATA ERROR  
 L3 Error: OK  
 SW: 0000  
 Message on error: ERROR OTHER CARD

### 2.4.8. Status returned

As with PayPass 2.x kernel, a status is returned by the `PayPass3_DoTransaction()` function. However, this return is now considered to be used for debug only, and the status of the transaction is now indicated in the OPS Status (see before).

## 2.5. Other steps where UIRD is returned

### 2.5.1. Remove Card sequence

As you will see in the sample application, we still continue to customise the following transaction steps to execute the “removal sequence”:

- STEP\_PAYPASS\_MCHIP\_REMOVE\_CARD.
- STEP\_PAYPASS\_MSTRIPE\_REMOVE\_CARD.

Generally, the UIRD provided on these steps is:

**USER INTERFACE REQUEST DATA**  
Message ID: CLEAR DISPLAY  
Status: CARD READ SUCCESSFULLY  
Hold Time: 000000  
Language Pref: 0000000000000000  
Value Qualifier: NONE  
Value: 000000000000  
Currency Code: 0000

This means that no message has to be displayed and only the LEDs and buzzer sequence is launched. Moreover, this GUI has to be processed within a task, in order the kernel processing can continue during the card read sequence is executed (please refer to the sample to see an example of implementation).

Take card that in some specific cases, a message could be required to be displayed on this step (i.e. Message ID different from “CLEAR DISPLAY”).

### 2.5.2. Message for double tap

When a mobile phone is presented and a double tap is about to be required, the following transaction steps are called:

- STEP\_PAYPASS\_MCHIP\_SEND\_PHONE\_MSG.
- STEP\_PAYPASS\_MSTRIPE\_SEND\_PHONE\_MSG.

A “User Interface Request Data” (UIRD) is provided on each of these customisation steps. The Message ID value depends on several transaction data (Phone Message Table, POS Cardholder Interaction Information, etc).

Here is an example

**USER INTERFACE REQUEST DATA**  
Message ID: SEE PHONE  
Status: NOT READY  
Hold Time: 000013  
Language Pref: 0000000000000000  
Value Qualifier: NONE  
Value: 000000000000  
Currency Code: 0000

Generally, as output of `PayPass3_DoTransaction()` function in this case, you would have:

<b>OUTCOME PARAMETER SET</b> Status: END APPLICATION Start: START B Online Response Data: N/A CVM: N/A Presence: UIR on Outcome: No. UIR on Restart: Yes. Data Record: Yes. Discretionary Data: Yes. Receipt: No. Alternate Interface Preference: N/A. Field Off Request: N/A. Removal Timeout: 0.	<b>USER INTERFACE REQUEST DATA</b> Message ID: SEE PHONE Status: READY TO READ Hold Time: 000000 Language Pref: 0000000000000000 Value Qualifier: NONE Value: 000000000000 Currency Code: 0000
<b>DISCRETIONARY DATA</b> Error indication. All the transaction discretionary data...	<b>ERROR INDICATION (in Discretionary Data)</b> L1 Error: OK L2 Error: OK L3 Error: OK SW: 0000 Message on error: N/A  <b>DATA RECORD</b> All the transaction data for record...

Note that the OPS indicates a restart from the beginning (START B) and the UIRD is the same as in the customisation step, but the Status indicates “READY TO READ” in order the GUI changes to indicate the reader is waiting for the mobile presentation.

## 2.6. Transaction flow

With PayPass 3.x, there is no more difference in the transaction flow as we previously had with PayPass 2.x (where refund transactions had a specific flow). So, whatever the transaction type, the same transaction steps will be executed.

So, if specific configuration is necessary for a kind of transaction, this is the role of the application to provide the adequate parameters (e.g. provide the correct parameters to force an AAC for a refund transaction, as this will not be done automatically by the kernel).

## 2.7. Other changes

### 2.7.1. Concept of “Empty Tags”

With the PayPass 3.x kernel, MasterCard introduced the concept of Empty Tags. An empty tag is considered as present within the database, but with a zero length. It means that if the card returns a zero length tag, it will be stored as empty in the kernel database (this kind of tags was ignored in PayPass 2.x). This concept is applicable for every kind of tags. We recommend not using this if you don’t use Data Storage or Data Exchange features.



## 2.7.2. Proprietary tags

As with PayPass 2.x, it is possible to declare some proprietary tags. With PayPass 3.x, each proprietary tag will be set to “empty” in the database (i.e. present with length set to 0). That is why each tag shall be defined with the TI\_OPTIONS\_RFU\_MASK\_EMPTY\_TAG\_ALLOWED in the RFU option field.

When Data Exchange is supported, please ensure that all terminal sourced proprietary tags are updated to include PAYPASS\_SOURCE\_DET in "Source" as indicated in the documentation.



Please refer to the documentation for more information about this.

## 2.7.3. Default kernel database values

To perform a transaction, the kernel needs a certain amount of mandatory data. With PayPass 3.x kernel, if some mandatory data are missing (i.e. not present), then the kernel will use default values for these missing tags. This concerns the payment and the data storage as well.

Here is the list of default values:

Data Object	Tag	Default
Additional Terminal Capabilities	TAG_EMV_ADD_TERMINAL_CAPABILITIES	'0000000000'
Application Version Number (Reader)	TAG_EMV_APPLI_VERSION_NUMBER_TERM	'0002'
Card Data Input Capability	TAG_PAYPASS_CARD_DATA_INPUT_CAPABILITY	'00'
CVM Capability – CVM Required	TAG_PAYPASS_MCHIP_CVM_CAPABILITY_CVM_REQUIRED	'00'
CVM Capability – No CVM Required	TAG_PAYPASS_MCHIP_CVM_CAPABILITY_CVM_NOT_REQUIRED	'00'
Default UDOL	TAG_PAYPASS_DEFAULT_UDOL	'9F6A04'
Hold Time Value	TAG_PAYPASS_HOLD_TIME_VALUE	'0D'
Kernel Configuration	TAG_PAYPASS_KERNEL_CONFIGURATION	'00'
Mag-stripe Application Version Number (Reader)	TAG_PAYPASS_MSTRIPE_APPLI_VERSION_NUMBER_TERM	'0001'
Mag-stripe CVM Capability – CVM Required	TAG_PAYPASS_MSTRIPE_CVM_CAPABILITY_CVM_REQUIRED	'Fo'
Mag-stripe CVM Capability – No CVM Required	TAG_PAYPASS_MSTRIPE_CVM_CAPABILITY_CVM_NOT_REQUIRED	'Fo'
Max Lifetime of Torn Transaction Log Record	TAG_PAYPASS_MAX_LIFETIME_OF_TORN_TXN_LOG_RECORD	'012C'
Max Number of Torn Transaction Log Records	TAG_PAYPASS_MAX_NUMBER_OF_TORN_TXN_LOG_RECORDS	'00'
Message Hold Time	TAG_PAYPASS_DEFAULT_HOLD_TIME	'000013'
Reader Contactless Floor Limit	TAG_EP_CLESS_FLOOR_LIMIT	'000000000000'
Reader Contactless Transaction Limit (No On-device CVM)	TAG_PAYPASS_CLESS_TRANSACTION_LIMIT_NO_DCV	'000000000000'
Reader Contactless Transaction Limit (On-device CVM)	TAG_PAYPASS_CLESS_TRANSACTION_LIMIT_DCV	'000000000000'
Reader CVM Required Limit	TAG_EP_CLESS_CVM_REQUIRED_LIMIT	'000000000000'
Security Capability	TAG_PAYPASS_SECURITY_CAPABILITY	'00'
Terminal Action Code – Default	TAG_EMV_INT_TAC_DEFAULT	'CC00000000'
Terminal Action Code – Denial	TAG_EMV_INT_TAC_DENIAL	'0000000000'
Terminal Action Code – Online	TAG_EMV_INT_TAC_ONLINE	'CC00000000'
Terminal Type	TAG_EMV_TERMINAL_TYPE	'00'
Time Out Value	TAG_PAYPASS_TIME_OUT_VALUE	'01F4'
Transaction Type	TAG_EMV_INT_TRANSACTION_TYPE and TAG_EMV_TRANSACTION_TYPE	'00'

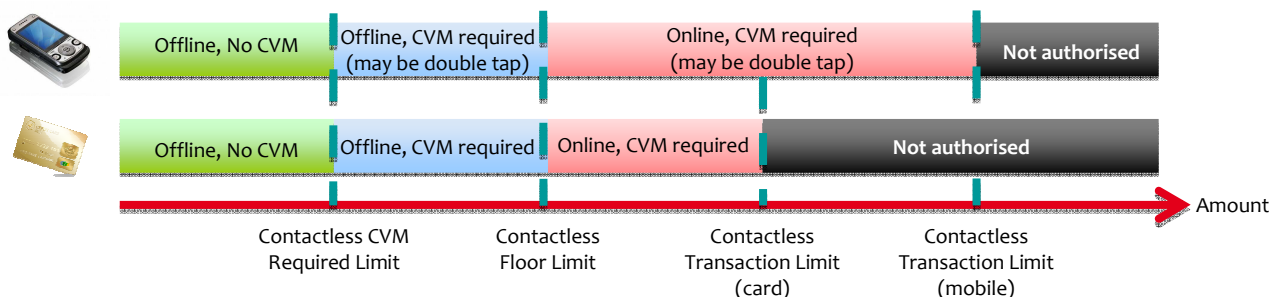


## 3. New features overview

### 3.1. Mobile transactions


MasterCard introduced a new Contactless Transaction Limit dedicated to Mobile (i.e. Cardholder devices capable of performing CVM):

- `TAG_PAYPASS_CLESS_TRANSACTION_LIMIT_NO_DCV` for cards.
- `TAG_PAYPASS_CLESS_TRANSACTION_LIMIT_DCV` for mobiles.



The PayPass 3.x kernel has dedicated mobile in the transaction flow, base on some transaction data. Two customisation steps are dedicated to mobile GUI processing as well:

- `STEP_PAYPASS_MCHIP_SEND_PHONE_MSG`.
- `STEP_PAYPASS_MSTRIPE_SEND_PHONE_MSG`.

 Please refer to the documentation for more information about how to implement this new feature.

### 3.2. Torn recovery

The aim of this feature is to be able to recover a torn transaction (i.e. when a contactless card/device is removed from the reader field during the last GENAC exchange. The subsequent torn transaction recovers this abnormal card removal. So it is only applicable for M/Chip transactions).

A torn transaction is stored in a torn record, within a torn log:

- Number of entries is configurable, through the following tag:
  - `TAG_PAYPASS_MAX_NUMBER_OF_TORN_TXN_LOG_RECORDS`.
- Record lifetime is configurable, through the following tag:
  - `TAG_PAYPASS_MAX_LIFETIME_OF_TORN_TXN_LOG_RECORD`.

The storage of the « torn records » is managed by the application:

- During the transaction (on specific dedicated transaction steps), the kernel can call the payment application to access the torn records:
  - Identify a record through step `STEP_PAYPASS_MCHIP_IS_TORN_RECORD`.
  - Add a record through step `STEP_PAYPASS_MCHIP_ADD_TORN_RECORD`.
  - Remove a record through step `STEP_PAYPASS_MCHIP_REMOVE_TORN_RECORD`.
- On a regular basis (outside of a transaction processing), expired records shall be removed from the log (depending on `TAG_PAYPASS_MAX_LIFETIME_OF_TORN_TXN_LOG_RECORD`).
- Note the torn record is volatile data as it shall be erased on terminal reset.

From the kernel perspective, the Torn Recovery feature is supported if:

- o Terminal supports it, i.e. TAG\_PAYPASS\_MAX\_NUMBER\_OF\_TORN\_TXN\_LOG\_RECORDS indicating the number of entries, is present and greater than zero.
- o Card supports it, i.e. it returns tag 0x9F51 (TAG\_PAYPASS\_DRDOL).

Torn Record Content	Tags
Amount, Authorized (numeric)	TAG_EMV_AMOUNT_AUTH_NUM
Amount, Other (numeric)	TAG_EMV_AMOUNT_OTHER_NUM
Application PAN	TAG_EMV_APPLI_PAN
Application PAN Sequence Number	TAG_EMV_APPLI_PAN_SEQUENCE_NUMBER
Balance Read Before Gen AC	TAG_PAYPASS_BALANCE_READ_AFTER_GENAC
CDOL1 Related Data	TAG_KERNEL_CONSTRUCTED_CDOL1_VALUE
CVM Results	TAG_EMV_CVM_RESULTS
DRDOL Related Data	TAG_PAYPASS_CONSTRUCTED_DRDOL_VALUE
DS Summary 1	TAG_PAYPASS_DS_SUMMARY_1
IDS Status	TAG_PAYPASS_INT_DS_IDS_STATUS
Interface Device Serial Number	TAG_EMV_IFD_SERIAL_NUMBER
PDOL Related Data	TAG_KERNEL_CONSTRUCTED_PDOL_VALUE
Reference Control Parameter	TAG_PAYPASS_INT_MCHIP_GENAC_REF_CTRL_PARAMETER
Terminal Capabilities	TAG_EMV_TERMINAL_CAPABILITIES
Terminal Country Code	TAG_EMV_TERMINAL_COUNTRY_CODE
Terminal Type	TAG_EMV_TERMINAL_TYPE
Terminal Verification Results	TAG_EMV_TVR
Transaction Category Code	TAG_PAYPASS_TRANSACTION_CATEGORY_CODE
Transaction Currency Code	TAG_EMV_TRANSACTION_CURRENCY_CODE
Transaction Date	TAG_EMV_TRANSACTION_DATE
Transaction Time	TAG_EMV_TRANSACTION_TIME
Transaction Type	TAG_EMV_TRANSACTION_TYPE
Unpredictable Number	TAG_EMV_UNPREDICTABLE_NUMBER

The content of a torn record is (some of these tags may be not returned if they are not present):

 Please refer to the documentation for more information about how to implement this new feature.

### 3.3. Data Exchange

The Data Exchange (DE) is a mechanism allowing the exchange of data between the kernel and the application in parallel with payment transaction processing. This processing is executed in parallel of the payment transaction processing, in a dedicated kernel task, during the card processing (i.e. during the terminal « free time »).

The developers may choose to use or not use this mechanism. For standard payment with mobile, this mechanism would be useless. This is generally used with the Data Storage mechanism.

The tag data involved in Data Exchange implementations may vary between regions.

 Please refer to the documentation for more information about how to implement this new feature.

## 3.4. Data Storage

The Data Storage feature provides an access (read / write) of proprietary data stored within card slots during a payment transaction. There exist two kinds of Data Storages. Data Storage implementations will vary from country to country and acquirer to acquirer and slot information / tags required will need clarified by the region.

### 3.4.1. SDS (Standalone Data Storage)

The SDS is standalone and need extra command in the transaction flow to read / write data within the card. It uses 10 special tags.

The write sequence uses PUT DATA commands:

- 5 tags without a MAC (9F75-9F79), can be written by anyone.
- 5 tags with a MAC (9F70-9F74), can be written by the Issuer.

The read access is open to anyone and uses the GET DATA commands.

 Please refer to the documentation for more information about how to implement this new feature.

### 3.4.2. IDS (Integrated Data Storage)

The IDS is integrated into the EMV transaction flow (i.e. no additional commands needed). The read operation is performed on the GPO command; the write operation is performed during the Generate AC command. The IDS contains enhanced security and is designed to work with CDA. The slot structure supports multiple operators.

One of the advantages of this solution is that it is transparent to non DS terminals.

 Please refer to the documentation for more information about how to implement this new feature.