

Transaction Optimization

For *PayPass* – M/Chip Terminals
07 July 2008

Number: 002582

Version: 2-0

Public

© 2008 MasterCard

Copyright

© 2008 MasterCard. The information contained in this manual is proprietary and confidential to MasterCard International Incorporated or one of its affiliated entities (collectively “MasterCard”) and its customer financial institutions.

This material may not be duplicated, published, or disclosed, in whole or in part, without the prior written permission of MasterCard.

Trademarks

Trademark notices and symbols used in this manual reflect the registration status of MasterCard trademarks in the United States. Please consult with the Customer Operations Services team or the MasterCard Law Department for the registration status of particular product, program, or service names outside the United States.

All third-party product and service names are trademarks or registered trademarks of their respective owners..

History

Version	Date	Comments	Author
0-1	20 Dec 06	First draft	DR / DG
0-2	27 Dec 06	Second draft	PSM
0-3	2 Jan 07	Minor changes	DR
0-4	15 Jan 07	DG changes	DR
1-0	19 Jan 07	Following review	DR
1-1	03 June 08	Updated following publication of performance requirements	DR
2-0	07 July 2008	Following review	DR

Preface

Objectives

The objective of this document is to present suggestions for optimising *PayPass* – M/Chip terminals in order to achieve fast transactions.

Scope

This document is intended for optimizing *PayPass* – M/Chip terminals. While the same ideas apply to *PayPass* – Mag Stripe terminals, *PayPass* – Mag Stripe is out of scope for this document.

This document is limited to optimizing the performance of terminals. Optimization of card performance is out of scope of this document.

Audience

Developers of *PayPass* – M/Chip terminals.

If you are reading this and recognize from the timing examples in chapter 6 that your product is at the slower end of the spectrum, you may want to check whether your terminal implements the tricks explained in this document.

If you are reading this and recognize that your product is at the faster end of the spectrum then it is unlikely that you will learn much from this document.

Related Documents

Title	Version	Reference
<i>PayPass</i> ISO/IEC 14443 Implementation Specification	1.1	[14443]
<i>EMV Contactless Communications Protocol</i>	2.0	[EMVCL]
<i>PayPass</i> – M/Chip Technical Specification	1.3	[PPMCHIP]
<i>PayPass</i> – Mag Stripe Technical Specification	3.3	[PPMAG]
<i>PayPass</i> Performance Measurement	1.3	[PPPERF]
<i>PayPass</i> Application Note # 2	-	[PPAN2]
EMV Book 3 – Application Specification	4.2	[EMVBOOK3]

Glossary

Term	Definition
AC	Application Cryptogram
ADF	Application Data File
AID	Application Identifier

Term	Definition
ATTRIB	PICC Selection, Type B
CDA	Combined DDA/AC Generation
DDA	Dynamic Data Authentication
FCI	File Control Information
GPO	GET PROCESSING OPTIONS
ICC	Integrated Circuit Card
PAN	Primary Account Number
PDOL	Processing Data Objects List
PK	Public Key
PPSE	PayPass Payment Systems Environment
RATS	Request Answer To Select (Type A)
RF	Radio Frequency
RSA	Rivest, Shamir & Adelman
S(WTX)	S-Block requesting Wait Time Extension
SDA	Static Data Authentication
SDAD	Signed Dynamic Application Data
SFI	Short File Identifier
SHA	Secure Hash Algorithm
SSAD	Signed Static Application Data
TC	Transaction Certificate

Assumptions

It is assumed that the reader is familiar with *PayPass – M/Chip* and EMV.

The timings quoted in this document are based to the requirements specified in Version 1.1 of the *PayPass* ISO 14443 Implementation Specification [14443]. A number of the parameters changed in Version 2.0 of the specifications [EMVCL] but not materially in regard of overall transaction times.

Table of Contents

1	Introduction.....	1
2	Polling & Activation sequence.....	2
3	Application selection	4
4	PayPass – M/Chip transaction.....	5
4.1	Introduction.....	5
4.2	Fixed behaviour.....	5
4.3	Overall transaction flow	6
4.4	Parallel processing.....	7
4.4.1	Communications chipset	7
4.4.2	Cryptographic processing.....	7
4.4.3	Command order.....	8
4.4.4	Minimal processing.....	8
5	Protocol Operation.....	9
5.1	Working time extensions.....	9
5.2	Chaining	9
5.3	Error handling & recovery.....	9
6	Timing examples	10
6.1	SELECT PPSE.....	10
6.2	SELECT AID _{MasterCard}	10
6.3	GET PROCESSING OPTIONS.....	10
6.4	READ RECORDS	10
6.5	1 st GENERATE AC.....	10

1 Introduction

Why are you reading this? Because MasterCard is keen to provide cardholders and merchants with a payment process that is fast, reliable and consistent. The performance of the card and the terminal are critical to this.

For an optimized experience, the terminal must give two indications as fast as it can:

- It must give a first indication after the end of 1st GENERATE AC command that the card may be removed from the magnetic field.
- It must give a second indication after completing its analysis of the transaction on success or failure of the transaction.

We have seen combinations of *PayPass*– M/Chip cards and terminals that give the first indication in less than 400 ms for SDA and less than 500 ms for CDA. We've also seen combinations taking three times as long. If your terminal is at the faster end of the spectrum in regard of performance, you are unlikely to learn much from this document.

This document concentrates on the terminal and explains how to optimize for giving the first indication. For this purpose, this document explains

- How to optimize the polling & activation sequence,
- How to optimize application selection and
- How to optimize the *PayPass*– M/Chip transaction flow.

Some of the observations about RSA processing are material to the time taken to the second indication.

MasterCard has published details on performance requirements [PPAN2] and measurement methods [PPPERF]. At the time of writing the time budgets for card and terminal are as follows for 96 byte key length CDA transactions:

	2008 target	2009 target
Card	500 ms	400 ms
Terminal	150 ms	100 ms

The time budget for the card includes card to terminal communications and the time budget for the terminal includes terminal to card communications.

[PPPERF] also indicates methods that must be used to measure the performance of a terminal.

2 Polling & Activation sequence

How to optimize the polling and activation sequence? It is governed by the prescribed sequence in the specification. The following text is based on [14443] rather than [EMVCL] but the basic principles remain the same.

On investigating the performance of a number of terminals, it has been noted that frequently, where there is specified a time for a function, many terminals wait longer than this even though any card that meets the specification will have responded within that time. To wait longer than this just wastes time.

Specifically the following are highlighted as places where the terminal implementation should strive to meet the specification but not wait any longer than this.

- The time between WUPA and WUPB commands during polling
- The Frame Delay Time request processing by the terminal should be close to 500 μ s
- The time between sending a HLTA command and the subsequent command
- The time between sending the REQA command and sending the WUPB command (assuming there is no response to the REQA command)
- The time between sending the REQB command and sending the WUPA command (assuming there is no response to the REQB command)
- The time from sending the WUPB command to sending the WUPA command if a Type A card is detected
- The time from sending the WUPA command to sending the WUPB command if a Type B card is detected

The performance of exception processing is particularly important during the polling and activation sequence as it is likely that communications errors will occur whilst the card is being introduced to the RF field. Therefore, the time taken to perform the following should be minimized as far as allowed within the specification:

- The duration of a RF field reset when a communications error is detected
- The timeout period allowed for a response to the HLTB and ATTRIB commands
- The timeout period allowed for the response to the RATS command
- The time taken to initiate exception processing once an error has been detected

A typical terminal at the upper end of the performance range should execute the activation sequence from the first Wake Up (WUPA or WUPB) command which

receives a response until the sending of the RATS or ATTRIB command in less than 35 ms. Slow terminals have been seen that take up to 250 ms. Note that the time taken for a Type B card will depend on the performance of the card and a slow card will take longer than the time for a typical card assumed here. Terminals have also been noted that perform other management activities during the activation sequence, sometimes adding 80 ms to the time taken.

3 Application selection

Application selection should look in the most efficient way for the AIDs that are supported by both card and terminal and identify the AID with the highest priority.

The application selection contains two steps:

- Sending the SELECT PPSE command to retrieve the FCI with the list of applications and their priority.
- Sending the SELECT AID command to open the application context.

There is little processing involved in analyzing the SELECT PPSE response; merely the parsing of the FCI data. Currently, most cardholder devices only contain one contactless application identifier (AID) in the PPSE. This situation may change with the introduction of NFC enabled mobile phones that contain multiple applications from different brands.

No assumptions or short cuts should be made when parsing the FCI of the PPSE.

In regard of the FCI of the payment application though, this is not the case. The aim is to issue the GET PROCESSING OPTIONS (GPO) command as quickly as possible. It is noted that the only important data element in the FCI record returned after the SELECT AID command is the PDOL. The PDOL determines the format of the next command i.e. the GPO command. More information on the PDOL can be found in section 4.

When parsing the FCI of the SELECT AID, the terminal therefore may focus initially on the presence of PDOL. The other data in the FCI has little impact on the transaction processing and parsing can wait, for example whilst the card is processing the GPO command.

4 PayPass – M/Chip transaction

4.1 Introduction

The *PayPass* – M/Chip transaction flow can be optimized using three assumptions:

1. The behavior of a *PayPass* – M/Chip card is fixed.
2. Most transactions will be successful and failed transactions can be discarded.

The first assumption allows pre-formatting of the commands GET PROCESSING OPTIONS, Read Record and 1st GENERATE AC.

The second assumption allows pre-population of the TVR (=Terminal Verification Results) and postponing the offline card authentication.

As a general rule, on top of these assumptions, the terminal should try to perform maximum parallel processing. RSA processing can be delegated to a crypto processor; the main processor can perform background tasks while a separate chipset controls the communication. A good example of both practices is given in section 4.4, during the READ RECORD command processing.

4.2 Fixed behaviour

For a *PayPass* application, the terminal can make the following assumptions, independent of whether the card supports *PayPass* – M/Chip or *PayPass* – Mag Stripe:

- The PDOL is empty/absent. This allows preparing the GET PROCESSING OPTIONS command upfront. If found to be present, then separate processing logic will need to be used.
- The data layout in the card is likely to follow one of two pre-defined layouts (but different between *PayPass* – M/Chip and *PayPass* – Mag Stripe).

The AIP in the response to the GET PROCESSING OPTIONS command indicates to the terminal whether the card is *PayPass* – M/Chip or *PayPass* – Mag Stripe.

If the card is *PayPass* – M/Chip, the terminal can make the following assumptions:

- The data required for running *PayPass* – Mag Stripe are located in SFI 1, Record 1 and doesn't have to be read for a *PayPass* – M/Chip transaction.
- The AFL is normally fixed (but different between *PayPass* – M/Chip SDA and *PayPass* – M/Chip CDA). This allows preparing the READ RECORD commands upfront.

The value of the AFL is normally '08 01 01 00 10 01 01 01 18 01 02 00' for *PayPass* – M/Chip SDA.

PayPass – M/Chip transaction

The value of the AFL is normally '08 01 01 00 10 01 01 01 18 01 02 00 20 01 02 00' for *PayPass* – M/Chip CDA.

If it is neither of these values then normal EMV processing as defined in [EMVBOOK3] will be necessary.

Based on the AIP and the correct value of the AFL, the following records should be read:

SFI	RECORD	SDA	CDA
2	1	✓	✓
3	1	✓	✓
3	2	✓	
4	1		✓
4	2		✓

- The CDOL1 is normally fixed. This allows preparing (part of) the 1st GENERATE AC command upfront. If it doesn't follow the fixed format, then generic code to handle it will need to be used. If the card has one of the two normal AFL values then the CDOL1 is located in SFI 2, Rec 1.

The CDOL1 for *PayPass* – M/Chip is usually equal to '9F0206 9F0306 9F1A02 9505 5F2A02 9A03 9C01 9F3704 9F3501 9F4502 9F4C08 9F3403'. This implies that the 1st GENERATE AC command may use a pre-defined template for the command data.

Note: On occasion, CDOL1 is extended with additional items after the normal set, but it is rare for the order of the specified set in the M/Chip 4 specification to be varied or not to form the first part of the command data.

- If the card is *PayPass* – M/Chip SDA with the normal AFL value, the terminal can assume that the data required for offline CAM is stored in SFI 3 Records 1 & 2.
- If the card is *PayPass* – M/Chip CDA with the normal AFL value, the terminal can assume that the data required for offline CAM is stored in SFI 3 Record 1 and SFI 4 Records 1 & 2.

4.3 Overall transaction flow

In previous releases of the the *PayPass* – M/Chip specifications [PPMCHIP] there are two models. This is changing and the next version of the specifications is likely to include just one.

In all cases upon receipt of the 1st GENERATE AC response, the terminal should indicate as quickly as possible that the card can be removed.

4.4 Parallel processing

Significant improvements in transaction time can be achieved by performing operations in parallel and by careful sequencing of the commands used to perform the transaction.

The general strategy should be built around keeping the delays between issuing READ RECORD commands and issuing the GENERATE AC command to a minimum whilst performing the maximum amount of additional processing that can be achieved while the terminal is waiting for the card to respond.

4.4.1 Communications chipset

Typical communications chipsets – e.g. the Philips RC531 – buffer the received data in a FIFO. This allows the main processor to be used to perform other tasks in parallel to the READ RECORD and GENERATE AC commands. Simple examples of parallel tasks are parsing the contents of the terminal file records, Processing Restrictions (Symbol 8 in section 3.1 of [PPMCHIP]) and parts of Symbols 9 and 10 (Terminal Risk Management and parsing the CVM list). It is also possible to perform part of Offline Data Authentication whilst awaiting responses from the card.

4.4.2 Cryptographic processing

Whilst Offline Data Authentication may be performed after the card has been removed from the RF field, it may also be performed in parallel with other processes to optimize the overall cardholder / merchant experience. This is especially true for time critical environments such as Transit. In this case care is needed in the design.

For offline card authentication, the terminal can assume that a *PayPass* – M/Chip card will only have one signed record, of typical size 180 bytes.

For SDA the cryptographic processes involved are as follows. Note that the exact sizes depend on key sizes; these examples are taken from the values used in the Type Approval profile 1 configuration.

Unwrap and check issuer key: One RSA operation with a short public exponent plus one SHA-1 hash of about 143 bytes

Unwrap and check SSAD: One RSA operation with with a short public exponent plus one SHA-1 hash of about 290 bytes.

For CDA the cryptographic processes involved are:

Unwrap and check issuer key: One RSA operation with a short public exponent plus one SHA-1 hash of about 143 bytes

Unwrap and check ICC PK Certificate: One RSA operation with a short public exponent plus one SHA-1 hash of about 300 bytes.

Unwrap and check SDAD: One RSA operation with a short public exponent plus two SHA-1 operations one of about 110 bytes and one of about 74 bytes.

Therefore for SDA we need two RSA operations and two SHA-1 hashes, which with the SHA-1 block size of 32 bytes¹ means in the region of 15 blocks of data to hash. For CDA we have three RSA operations and four SHA-1 operations totaling about 22 blocks of data. Variations in key size will alter this slightly.

Whilst the RSA operations can be handled well by a cheap coprocessor such as a smart card processor, care needs to be taken with the hashing. SHA-1 is by its nature a slow operation unless a dedicated hardware SHA-1 coprocessor is available. Asking a typical smart card processor to perform SHA-1 operations is likely to cost about 10 msec per hash, and passing a large amount of data over a slow interface will also take time. Therefore care must be exercised in regard of how and where SHA-1 operations are performed. If a smart card processor is used then consider asking it to start verification of the issuer key as soon as the data is available. If terminal file records are read in sequence, this would be after reading SFI3 record 1.

If a smartcard cryptographic coprocessor is used then it should support a high speed interface and the data required to be transferred between the terminal and the coprocessor should be minimized (e.g. Payment System public keys should be cached in the coprocessor rather than transferred on each transaction)

4.4.3 Command order

If the terminal has a separate crypto processor, it might make sense to read SFI3 record 1 first in order to determine the issuer certificate so that checking it can be started in parallel with sending the other READ RECORD commands.

4.4.4 Minimal processing

Obviously a terminal should not waste time performing processing which isn't strictly required by the specification. For example there is little benefit verifying that the Luhn check digit of the PAN is correct. The PAN will be signed in either the ICC PK Certificate or the SSAD anyway.

¹ SHA-1 is a block oriented algorithm that processes data in 32 byte blocks. Processing time is therefore related to how many 32 byte blocks must be processed plus an initial process independent of message length. Padding the final block does not materially affect the overall time.

5 Protocol Operation

5.1 Working time extensions

If a card requests a working time extension by sending an S(WTX) request then the terminal should send the S(WTX) response with a time delay as close to the minimum permitted by the Contactless specifications [14443, EMVCL] as possible. It is not unusual for many S(WTX) request/responses to be required to complete a transaction particularly during a *PayPass* – M/Chip CDA transaction.

5.2 Chaining

Terminals should be designed with sufficient buffer space that chaining is not required due to limitations of the terminal. When chaining is required because of limitations in the card's buffer space then the terminal should minimize the delay incurred by sending or requesting the next block as quickly as is permitted by the specification.

5.3 Error handling & recovery

As described in Section 2 the performance of exception processing during the polling and activation sequence is particularly important and it is likely that communications errors will occur whilst the card is being introduced to the RF field. The performance of exception processing during the remainder of the transaction may be equally important if the terminal is highly sensitive to card or environmental noise.

The performance of exception processing during the protocol operation phase of the transaction should be optimized to ensure that when errors occur these have the minimum impact on the overall transaction time. In particular the following should be minimized as far as allowed within the specification;

- The time between a transmission error being detected by the terminal and the sending of R(NACK) or R(ACK)
- The time between the expiry of the FWT indicated by the card (without a response being received by the terminal) and the terminal sending the R(NACK) or R(ACK)

In order to achieve the best possible transaction time even when the terminal is subjected to card and/or environmental noise it is important that the terminal can intelligently distinguish between a real communications error and noise. This ability to distinguish between genuine errors and noise enables the terminal to correctly initiate exception processing with the minimum of delay and avoid initiating exception processing prematurely. Terminal developers should pay particular attention to the user guidance provided by communications chipset providers as well as the *PayPass* specifications and associated Application Notes to optimize the terminal's immunity to noise whilst maintaining the fastest possible transaction time.

6 Timing examples

This section contains some examples of terminal and card timings. These figures should only be taken as guideline.

6.1 SELECT PPSE

Significant differences in performance have been noted between different terminals, for reasons that are not clear.

The time between the SELECT PPSE response from the card and the terminal starting the selection of the ADF has been seen to vary from less than 3 ms to over 25 ms for different terminal types.

6.2 SELECT AID_{MasterCard}

The spread of times noted in terminals to parse the ADF FCI data in the SELECT AID response before issuing the GET PROCESSING OPTIONS command is from less than 2 ms to over 26 ms.

6.3 GET PROCESSING OPTIONS

A typical card will take between 30 and 70 ms to respond to a GET PROCESSING OPTIONS command.

This “idle” time can be used for performing terminal housekeeping functions (e.g. future tasks such as those parts of Symbol 9 in [PPMCHIP] section 3.1 that do not rely on card provided data). As cards get faster, the above times are likely to be reduced. Therefore try to avoid presuming that time for more than about 20 ms of such processing will be available.

6.4 READ RECORDS

Reading records takes time. In general, with the densely packed records of a *PayPass*–M/Chip card, it is going to take over 20 ms for each one.

Terminal time delays between READ RECORDS have been seen in the range 2 to 5 ms. The time delay from the end of READ RECORDS to the start of 1ST GENERATE AC is very variable with some terminals taking as long as 250 ms, but ‘best of breed’ is less than 5 ms.

6.5 1st GENERATE AC

Given current card performances, it is unlikely that a card will respond with a TC in less than about 80 ms. It may take as long as 500 ms for a poor card or long keys with CDA. Typical CDA performance is likely to add 200 ms to the card processing. The time delay after GENERATE AC to the start of the audible / visual cues to remove the

card should be insignificant, but some terminals have been seen that add up to 80 ms of delay.

= *end of document* =