

Magensa Payment Protection Gateway (MPPGv3)

**EMV (chip card) Acceptance
Integration Guide**

June 12, 2020

Document Part Number:
D998200349-30

REGISTERED TO ISO 9001:2015

INFORMATION IN THIS PUBLICATION IS SUBJECT TO CHANGE WITHOUT NOTICE AND MAY CONTAIN TECHNICAL INACCURACIES OR GRAPHICAL DISCREPANCIES. CHANGES OR IMPROVEMENTS MADE TO THIS PRODUCT WILL BE UPDATED IN THE NEXT PUBLICATION RELEASE. NO PART OF THIS DOCUMENT MAY BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, FOR ANY PURPOSE, WITHOUT THE EXPRESS WRITTEN PERMISSION OF MAGTEK, INC.

MagTek® is a registered trademark of MagTek, Inc.
MagnePrint® is a registered trademark of MagTek, Inc.
Magenta™ is a trademark of MagTek, Inc.
MagneSafe® is a registered trademark of MagTek, Inc.
DynaPro™ and DynaPro Mini™, are trademarks of MagTek, Inc.
ExpressCard 2000™ is a trademark of MagTek, Inc.
IPAD® is a trademark of MagTek, Inc.
IntelliStripe® is a registered trademark of MagTek, Inc.

AAMVA™ is a trademark of AAMVA.
American Express® and EXPRESSPAY FROM AMERICAN EXPRESS® are registered trademarks of American Express Marketing & Development Corp.
D-PAYMENT APPLICATION SPECIFICATION® is a registered trademark to Discover Financial Services CORPORATION
MasterCard® is a registered trademark and PayPass™ and Tap & Go™ are trademarks of MasterCard International Incorporated.
Visa® and Visa payWave® are registered trademarks of Visa International Service Association.

MAS-CON® is a registered trademark of Pancon Corporation.
Molex® is a registered trademark and PicoBlade™ is a trademark of Molex, its affiliates, related companies, licensors, and/or joint venture partners

ANSI®, the ANSI logo, and numerous other identifiers containing "ANSI" are registered trademarks, service marks, and accreditation marks of the American National Standards Institute (ANSI).
EMVCo™ and EMV™ are trademarks of EMVCo and its licensors.
ISO® is a registered trademark of the International Organization for Standardization.
PCI Security Standards Council® is a registered trademark of the PCI Security Standards Council, LLC.
UL™ and the UL logo are trademarks of UL LLC.
The Bluetooth® word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by MagTek is under license.

All other system names and product names are the property of their respective owners.

Table A - Revisions

Rev Number	Date	Notes
10	11/6/2019	Initial Release
20		<ul style="list-style-type: none">• Updated Table C, Chase (Orbital) - Retail / REFUND information to reflect the use of transaction ID or token + auth code• Updated information about REFUND operation in section 2.2.3.1, under Amount heading• Updated information in section 2.2.3.2, under PaymentMode / For eDynamo readers to better reflect how to handle eDynamo readers• Added information about handling tips under section 2.2.3.3• Added cautionary notice under section 4.1.2 regarding REFUND test• Added reference to Chase under section 4.1.2.2 retail REFUND test, as Chase can do refund by reference ID or token• Removed Chase from section 4.2.3.1 and added Chase to section 4.2.3.2 REFUND testing, as Chase only does REFUND for restaurant by token• Added information about fault code 602• Added more information about fault code 712• Added information about fault code 721• Added information about fault code 749
30	5/26/2020	<ul style="list-style-type: none">• Added information for Heartland to Table B, Table C, section 2.2.3.3, and section 4• Added information about static device IDs for TSYS accounts to the beginning of section 2.2.3.3• Added more information about fault code 749

Table of Contents

Table of Contents	4
1 Introduction	6
2 Operational Overview	8
2.1 Commanding the Reader	8
2.1.1 Command Parameters	8
2.1.1.1 Card Type	8
2.1.1.2 Quick Chip Mode	9
2.1.1.3 Additional Command Arguments	9
2.1.1.4 FALLBACK	9
2.1.2 Reader Output	10
2.2 Processing to MPPG	12
2.2.1 Transaction Types	13
2.2.2 Methods	14
2.2.3 Method Inputs	14
2.2.3.1 Shared Input Data	14
2.2.3.2 Method-Dependent Input Data	15
2.2.3.3 Processor-Dependent Input Data	16
2.2.4 Method Outputs	18
2.2.4.1 Data from the Card-Issuing Financial Institution	18
2.2.4.2 Data from the Merchant Processor	18
2.2.4.3 Data from the MPPG Service	18
2.3 Receipts	19
3 Integration and Testing Preparation	22
3.1 Integration Preparation	22
3.2 Testing Preparation	22
4 Certification	23
4.1 Retail Certification	23
4.1.1 SALE (Chase/Elavon/First Data/Heartland/TSYS/WorldPay)	23
4.1.2 REFUND	24
4.1.2.1 REFUND By Reference ID (Chase/Heartland/TSYS/WorldPay)	24
4.1.2.2 REFUND By Token (Chase/Elavon/First Data)	24
4.2 Restaurant Certification	25
4.2.1 AUTHORIZE (Chase/First Data/Heartland/TSYS/WorldPay)	25
4.2.2 CAPTURE	26
4.2.2.1 CAPTURE By Reference ID (Heartland/TSYS/WorldPay)	26

4.2.2.2	CAPTURE By Token (Chase/First Data).....	26
4.2.3	REFUND	27
4.2.3.1	REFUND By Reference ID (Heartland/TSYS/WorldPay)	27
4.2.3.2	REFUND By Token (Chase/First Data).....	27
5	Reporting Test Results.....	28
Appendix A	29
A.1	Fault Error Codes.....	29
A.1.1	FaultCode: 602 (KSN is required).....	29
A.1.2	FaultCode: 603 (CustomerCode is required)	29
A.1.3	FaultCode: 604 (Username is required)	29
A.1.4	FaultCode: 605 (Password is required)	29
A.1.5	FaultCode: 619 (Amount is not valid).....	29
A.1.6	FaultCode: 701 (Access Denied)	29
A.1.7	FaultCode: 708 (EncryptionType is required)	30
A.1.8	FaultCode: 712 (Unknown Error)	30
A.1.9	FaultCode: 721 (Payload post failed)	30
A.1.10	FaultCode: 738 (Invalid EMVSRED Data)	30
A.1.11	FaultCode: 749 (Error in <token method>)	30
	Error in TokensCallRedeemTokenWebService An unexpected error occurred.....	30
	Error in TokensCallRedeemTokenWebService Token verification failed	30
A.1.12	FaultCode: 5000 (Unknown Error).....	30
A.2	Other Errors	31
A.2.1	FaultCode: DeserializationFailed	31

1 Introduction

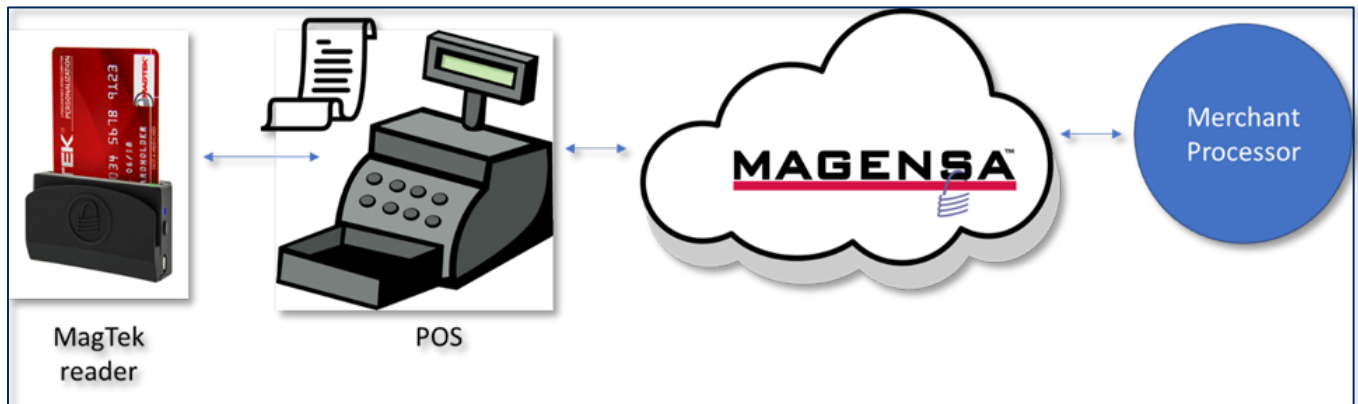
The Magensa Payment Protection Gateway (MPPG¹), a Magensa service², provides to POS integrators an interface to various merchant processors for magstripe and EMV (chip card) transactions using MagTek readers. This guide outlines the steps to integrate and certify to MPPG for EMV processing. The user of this guide is assumed to be a POS integrator, termed “user”. In order to use MPPG, the user must first complete the following items:

- 1) Execute a Magensa Service Agreement, which outlines pricing for the service and responsibilities of the user.
- 2) Obtain a MagTek reader capable of EMV processing. Please see www.magtek.com for the current list of certified devices, operation modes, merchant processors and card brands supported. Most EMV devices will also support magstripe. EMV devices must be ordered to be compatible with Magensa, which will include a compatible EMV tag load. Generally, all readers are provided with the Magensa Public Key installed. However, other encryption keys can be supported. Please see your MagTek sales representatives for more details.
- 3) Have a POS application (created by the user or obtained by a third-party) that has the following functions³:
 - a) Control of the MagTek reader by a host device, such as a Windows PC, mobile device, or an integrated processor. MagTek readers offer a variety of interface methods including USB, IP, Bluetooth, etc. Depending upon the interface chosen, there are several command format options as well, which vary from byte-level commands and OS interfaces via an SDK, to middleware and browser options provided by Magensa’s MagneFlex.
 - b) Posting to and reception from, via the Internet, the MPPG service. Documentation for the MPPG interface, as well as sample inputs, are available to assist you in making the MPPG connection.
 - c) Cardholder receipt generation, based on data returned by MPPG.
- 4) Test the POS application using MPPG Pilot credentials. These credentials will allow you to test your integration to the reader and MPPG without needing a live merchant account at the desired merchant processor.
- 5) Complete the MPPG certification script outlined at the end of this document.
- 6) Onboard merchant account(s) with a supported merchant processor and obtain MPPG merchant credentials for each.

¹ Currently versioned: MPPGv3

² Magensa LLC is a subsidiary of MagTek, Inc., providing payment processing and data security services.

³ If the user does not have a POS application, Magensa’s QwickPAY product may be used. Please see your sales representative for more details.



2 Operational Overview

To properly use an EMV-capable reader with MPPG, the POS application must establish a process flow of interaction with the reader and with MPPG that adheres to the operational attributes of the EMV standard. The following section outlines these steps beginning with the reader and ending with MPPG. Refer to the Programmer's Reference Manual for detailed information concerning a particular reader's commands and general function.

2.1 Commanding the Reader

The basic interaction flow between the reader and the POS application is as follows:

- 1) The POS application commands the reader to prepare for an EMV transaction. Included in the command are a variety of parameters that control the behavior of the reader during the interaction with the card.
- 2) The reader interacts with the card, and, depending on reader type, may also interact with the cardholder via a screen or through Notification Messages from the reader that the POS application displays to the cardholder. Such interaction is controlled by the reader.
- 3) The reader returns encrypted card data (nominally referred to as an 'ARQC'), or error data.

MagTek readers are designed to support the EMV standard through three payment card interfaces:

- **SWIPE** – The swipe of a magnetic stripe across a read head.
 - All readers, other than eDynamo, treat **SWIPE** as an EMV function when an EMV transaction has been requested, and output **SWIPE** data in a manner similar to **INSERT** and **TAP**.
 - eDynamo **SWIPE** behaves in the same manner as MagTek magstripe-only readers. Processing of magstripe-only **SWIPE** data to MPPG is not covered in this document. See the MPPG Programmer's Reference Manual **ProcessCardSwipe** method for instructions in processing magstripe-only **SWIPE**.
- **INSERT** – The insertion of a chip card into a read slot such that the reader makes an electrical connection with the card's contact pad.
- **TAP** – The placement of a chip card (or mobile phone) against the reader such that it can be read via a wireless interface.


MagTek readers will support these interfaces in a variety of combinations, depending upon the type of reader.

2.1.1 Command Parameters

There are several parameters that must be passed to the reader when commanding the start of an EMV transaction:

2.1.1.1 Card Type

This parameter expresses which card payment interfaces the reader will accept (and *not* accept) during the transaction. Regardless of command format option, the enumerated values of card type are as follows:

- 1 – **SWIPE** only (do not use in conjunction with this specification, as this card type will suppress EMV operation)
- 2 – **INSERT** (insertion of a chip card)
- 3 – **SWIPE** or **INSERT**
- 4 – **TAP** (tap of a contactless card) 
- 5 – **SWIPE** and **TAP**
- 6 – **INSERT** and **TAP**
- 7 – **SWIPE**, **INSERT** or **TAP**

Some MagTek readers support all interfaces, while others only two or even one.

2.1.1.2 Quick Chip Mode

MPPG is designed to use the mode of processing EMV transactions termed “Quick Chip”. This mode greatly reduces processing time. It also does not require a true transaction amount to be known before commanding the reader, thus allowing a cardholder to present their card in advance of the POS application knowing the final amount. Quick Chip is activated by an argument included in the EMV command to the reader. Please see the Programmer’s Reference Guide for further information. Processing EMV data to MPPG without Quick Chip activated will likely result in unexpected or unwanted transaction behavior at MPPG or the Merchant Processor.

2.1.1.3 Additional Command Arguments

Generally, the following additional arguments are included in the command. Please see the particular reader’s Programmer’s Reference Manual for formatting.

- *Transaction Flow Time* - Specifies the maximum time, in seconds, for user interaction events to complete before the reader times out.
- *Amount Authorized* – Under the Quick Chip mode described above, this argument is not the amount that is to be authorized for the transaction. Instead, a plug amount should be used. The value is discretionary but must be greater than \$0.00 (assumed US Dollars) and should be less than \$10, so as to not trigger card CVM limits.
- *Transaction Type* – Set this argument to the enumeration for “Purchase”.
- *Transaction Currency Code* – Set this argument to 0x0840, US Dollars. MPPG does not process in currency other than US Dollars.

Other fields not referenced here may be set to 0x00 or default, as appropriate. Cashback at the point of sale is not currently supported by MPPG.

2.1.1.4 FALLBACK

MagTek anticipates that cardholders may be unaware of the options they have to present a card, especially the existence of chip card options such as **INSERT** or **TAP**. In addition, cardholders may be unaware that, in certain cases, they may **SWIPE** if **INSERT** fails. Therefore, MagTek readers support, to varying degrees, **FALLBACK**. This operational behavior manifests in different ways, depending on the reader:

- **Magstripe FALLBACK** – The reader will prompt the cardholder (via screen or Notification Message, as applicable) to **INSERT** if the cardholder performed a **SWIPE** and the card supports **INSERT**. All readers support this behavior *except* for eDynamo. In the case of eDynamo, if **SWIPE** occurs, the POS application may inspect the masked track 2 data returned for the first digit of the card's service code (found immediately after the expiration date). If it is a '2' or '6', then the card is capable of **INSERT**. The POS application can then, if desired, restart the transaction and inform the cardholder to **INSERT**.
- **Technical FALLBACK** – If **INSERT** fails to read properly, the cardholder (via screen or Notification Message, as applicable) will be prompted to re-attempt **INSERT**. If three attempts are unsuccessful, the cardholder will then be prompted to **SWIPE**. In the case of eDynamo, if **INSERT** fails, the reader will decline the transaction by producing a Notification Message containing error data. The POS application can inspect this data for EMV⁴ tag DFDF1A. If the value is "FF", which indicates chip failure, the POS application can inform the cardholder to **SWIPE** (upon which magstripe **FALLBACK** should be ignored).

It is important to note that automatic **FALLBACK** behavior *will not* occur if the card type chosen initially is equal to 2, 4 or 6. EMV-style liability protection is extended to Technical **FALLBACK** for certain processors. Please see Table B: Input Parameters.

2.1.2 Reader Output

Upon a successful interaction with the cardholder (normal or **FALLBACK**), the reader will respond with a TLV-formatted data container termed the 'ARQC'. (Except eDynamo, where **SWIPE** should be treated as a Magstripe-only reader.) This container should be TLV-parsed into its constituent data elements.

⁴ Data output from EMV-capable readers is formatted using BER-TLV as described in ISO 7816, X.690, EMV and other specifications. The Internet contains a variety of resources and code samples for parsing these data objects.

2 - Operational Overview

Table B: Input Parameters

Merchant Processor	Restaurant*	Amount**	Partial Approval Key***	Technical Fallback	ProcessorName	PIN
Chase (Orbital)	BaseAmt+Tip	cents	partialAuthInd	No liability protection	Chase - [Pilot Production]	Not supported
Elavon (Converge)	Not supported	cents	PartialAuthIndicator	Liability protection (except eDynamo)	Elavon - [Pilot Production]	Not supported
First Data	BaseAmt+Tip	cents	PartAuthrznApprvlCapabl	Liability protection (except eDynamo)	Rapid Connect v3 - [Pilot Production]	Supported
Heartland	BaseAmt+Tip	dollars/cents	(supported, but disabled by default)	No liability protection for eDynamo, liability protection for other readers	Heartland - [Pilot Production]	Not supported
TSYS (MultiPass)	Tip only	both	(Always on)	No liability protection	TSYS - [Pilot Production]	Not supported
WorldPay	BaseAmt+Tip	dollars/cents	PartialApprovedFlag	Liability protection	VantivExpress - [Pilot Production]	Not supported

* How the final amount is presented at CAPTURE

** Supported amount format is either cents (e.g. "12345"), dollars and cents (e.g. "123.45"), or both

*** Key for key/value pair to use for setting partial approval amount

Table C: Input Methods

Merchant Processor	SALE/AUTH	CAPTURE	VOID	REFUND
Chase (Orbital) - Retail	Supported	Token + AuthCode	Not supported	TransactionID or Token + AuthCode
Chase (Orbital) - Restaurant	Supported	Token + AuthCode	Not supported	Token + AuthCode
Elavon (Converge)	SALE only	Not supported	Token*	Token
First Data (Nashville, Omaha, North)	Supported	Response** + Token + MCC***	Response** + Token**** + MCC***	Token + MCC***
Heartland	Supported	TransactionID	TransactionID	TransactionID
TSYS (MultiPass)	Supported	TransactionID	TransactionID	TransactionID
WorldPay	Supported	TransactionID	TransacctionID	TransactionID

* Only used to VOID an immediately preceding SALE transaction in the same processing day

** The verbatim response from the processor must be provided as an input

*** Merchant Category Code; see your First Data representative

**** Must be processed within 15 minutes of original transaction

2.2 Processing to MPPG

Once the POS application has collected reader data, it must process it to MPPG to complete the transaction. MPPG follows a mostly standardized model used by the payments industry for processing card-based payment transactions. This model is based on two, distinct transaction messages that occur at the national card acceptance brands that MPPG supports (Visa, MasterCard, American Express, Discover):

Authorization Request: A request sent to the card-issuing financial institution by a merchant to hold funds in the cardholder's account. This transaction, if approved, is a conditional promise made by the financial institution to the merchant to honor a future request for the payment of funds requested by the authorization. The representation of this promise is the "Authorization (sometimes: Approval) Code" returned by the financial institution. If the authorization is approved, the financial institution may hold the funds in the cardholder's account for a period of time (varies by financial institution). That is, make them unavailable to be used by the cardholder. The cardholder may become aware of this through notifications sent by the financial institution, or by referring to their online banking app (often referred to as "processing" or "pending").

In some cases, it is possible the financial institution may approve a transaction but for an amount less than requested. This is termed a *Partial Authorization*. If this occurs, it is the merchant's responsibility to collect the remaining funds directly from the cardholder via another means, including another card transaction with a different card (where *Partial Authorization* could occur again, continuing the cycle). *Partial Authorization* may be set on or off at transaction time for some processors. See Table B: Input Parameters for details.

Settlement Request: A request sent by the merchant to be paid the funds that were originally requested by an Authorization Request. Until this request is made, funds will NOT be paid. Payment of funds to the merchant can take a business day or more to complete. When making this request, there are two scenarios that must be considered for settlements requests supported by MPPG:

- 1) The amount of the settlement matches the authorization request. This scenario is by far the most common and can be found across a wide variety of merchant types.
- 2) The amount of the settlement exceeds the authorization request. In general, this is only supported for merchants defined as "restaurants" by their Merchant Category Code (MCC) at their merchant processor. In this case, the additional funds over the authorization amount represent a tip. In general, an amount equal to or less than 120% of the originally authorized amount may be requested as a final amount (base plus tip), as outlined by rules promulgated by the acceptance brands. Please refer to the processor or acceptance brands for more detail. An amount greater than 120% may be requested, though the financial institution may refuse to pay those funds or may process a chargeback later. Not all merchant processors supported by MPPG allow for restaurant MCCs. See Table B: Input Parameters for more detail.

The messages outlined above are not provided to the user of MPPG directly. Rather, they are initiated by an MPPG *method* modified with a *transaction type*. Methods are the SOAP or REST actions that initiate processing at MPPG. Transaction types indicate how a user wishes to initiate an authorization or settlement request.

2.2.1 Transaction Types

- **SALE** – Instructs the merchant processor to perform an authorization request, and, if successful, also perform a settlement request *automatically* for the same amount. This transaction type is recommended for most merchants, unless a tip needs to be collected *after* authorization. Settlements are generally not processed immediately but are held and then batch-processed at a cutoff time each business day by the merchant processor. Generally, merchants may request the merchant processor adjust this time to one that meets their needs. MPPG will return data immediately indicating if the authorization request was approved or declined. The settlement will be processed automatically without response data. This transaction is subject to *Partial Authorization*, depending on the merchant processor.
- **AUTHORIZE** – Instructs the merchant processor to perform an authorization request *only*. MPPG will return data immediately indicating if the authorization request was approved or declined. This transaction type WILL NOT result in settlement or the payment of funds to the merchant. A follow-on **CAPTURE** transaction must be used as described below. **AUTHORIZE** is generally chosen when the amount of the settlement request is anticipated to be different from the authorization request, such as when a tip is added. This transaction type is only recommended when a tip must be collected after the presentation of a payment card. Using **AUTHORIZE-CAPTURE** instead of **SALE** can lead to additional fees or penalties if misused. The card acceptance brands generally recommend that tips be collected before a card is presented. In that case, **SALE** can be used instead, generally leading to better outcomes and lower fees. This transaction is subject to *Partial Authorization*, depending on merchant processor.
- **CAPTURE** – A request that funds authorized via **AUTHORIZE** be paid to the merchant. **CAPTURE** should only be used once to settle an outstanding authorization request. **CAPTURE** SHOULD NOT be used when **SALE** was used for the original authorization request. Doing so may result in unpredictable behavior or multiple charging of the cardholder's account. Once a **CAPTURE** is processed, it cannot be modified or canceled. While settlement will not occur immediately, MPPG will return data indicating if the settlement request was accepted by the merchant processor. However, under special circumstances, the issuing institution may still reject the settlement request. Please contact the merchant processor if this occurs.
- **VOID** – A request that a previously processed authorization request via a **SALE** or **AUTHORIZE**(where the **CAPTURE** has not yet been processed) be *reversed* and funds held by the cardholder's issuing institution be released. This transaction will generally only work before the cutoff time established by the merchant processor, or in some cases, even earlier. See Table C: Input Methods. This transaction type is provided to allow the POS application to immediately cancel a transaction if an error occurred or the cardholder requests it. MPPG will return data immediately indicating if the reversal request was accepted. If an amount is requested in the transaction, provide the *full* amount of the original **SALE** or **AUTHORIZE**.
- **REFUND** – A request that funds received by the merchant from a previously processed **SALE** transaction or **AUTHORIZE-CAPTURE** pair be deducted from the merchant and returned to the cardholder. The amount of funds can be equal to or less than what was originally paid to the merchant. This transaction is generally not available before the settlement has taken place. In the past, this transaction effectively caused a settlement request (without an authorization message) to be processed with a negative amount. However, the card acceptance brands have now required that an authorization request be processed first with the negative amount, and then the settlement request. (As in **SALE**.) However, the transition to this model in the industry is not complete. This transition to the new model does NOT affect how this transaction type is used by the POS application. However, some cardholders may see an immediate return of their funds. MPPG will return data immediately indicating if the **REFUND** request has been accepted (and in some cases, as noted, authorized).

MPPG does not support **REFUNDS** that are not preceded by a **SALE** transaction or **AUTHORIZE-CAPTURE** pair (sometimes referred to as “unreferenced” refunds).

2.2.2 Methods

Methods are the SOAP or REST actions that initiate processing at MPPG for a particular transaction type. They accept reader data and control fields as arguments. Generally, methods vary by the type of reader data being presented.

Five methods are provided by MPPG:

- 1) **ProcessCardSwipe**: This method should be used only with eDynamo to process technical **FALLBACK** (use CardInputCode: 9). See the MPPG Programmer’s Reference Manual for details on this method.
- 2) **ProcessEMVSRED** (to be replaced with **ProcessDATA** in MPPG V4): Use this method to process card data obtained from **SWIPE** (except eDynamo), **INSERT** or **TAP** (as noted earlier, the reader must be commanded for EMV). For **SALE** and **AUTHORIZE**.
- 3) **ProcessReferenceID**: Use this method to process **VOID**, **REFUND** or **CAPTURE** for certain processors. See Table C: Input Methods. This method does not require any interaction with or data from a reader.
- 4) **ProcessToken**: Use this method to process **VOID**, **REFUND** or **CAPTURE** for certain processors. See Table C: Input Methods. This method does not require any interaction with or data from a reader.

2.2.3 Method Inputs

This section discusses the user-provided inputs required for MPPG when EMV is employed. As some processors have significantly different input structures than others, please refer to the sample MPPG inputs by processor and transaction type found in MPPG Input Samples. All fields are mandatory, unless otherwise noted.

2.2.3.1 Shared Input Data

All methods have the following input fields in common:

- **Amount** – The amount being requested. See Table B: Input Parameters for format information.
 - *TSYS only, REFUND only (strongly recommended):*
 - If a **REFUND** transaction will be for the full amount of the referenced finalized transaction, then do not include the <amount> element in the request message. If the amount element is not included, then the TSYS backend will assume that the **REFUND** transaction will be refunding the full amount of the referenced finalized transaction and set up the transaction accordingly. In addition, the transaction will automatically be run as a **VOID** transaction if the referenced finalized transaction has not been settled before the **REFUND** request is received.
 - If a **REFUND** transaction will be for a partial refund that is less than or equal to the amount of the original **AUTHORIZE** transaction, the <amount> element must be included and the <tip> element omitted.

- If a REFUND transaction will be for a partial refund that is greater than the amount of the original AUTHORIZE transaction but less than the amount of the CAPTURE transaction (authorized amount plus tip), the <amount> element must be set to the amount of the original AUTHORIZE transaction and the <tip> element must be set for the remaining amount to refund.
- **Username** – Assigned to a specific merchant by Magensa corresponding to that merchant’s account at a processor.
- **Password** – Assigned to a specific merchant by Magensa corresponding to that merchant’s account at a processor.
- **CustomerCode** – Assigned to a specific merchant by Magensa corresponding to that merchant’s account at a processor.
- **CustomerTransactionID** – a user-defined number that will be echoed by the merchant processor through the MPPG response.
- **ProcessorName** - The name of the merchant processor. See Table B: Input Parameters. “Production” is for production use. “Pilot” may be used when testing on the MPPG Pilot system. See the Certification and Testing section.

2.2.3.2 Method-Dependent Input Data

The following methods have specific requirements for input fields:

- ProcessEMVSRED
 - **EMVSREDDData** – ARQC data⁵ returned from the reader, or EMV tag DFDF59 (Encrypted Data Primitive).
 - **EncryptionType** – EMV tag DFDF57 (encryption type)
 - **KSN** - EMV tag DFDF56 (Key Serial Number)
 - **NumberOfPaddedBytes** - EMV tag DFDF58 (Number of Bytes of Padding in the Encrypted Data Primitive)
 - **PaymentMode**
 - For all readers other than eDynamo:
 - If EMV tag DFDF53 is 0x00 and the reader was configured to read EMV data as primary and MSR data as fallback, use “EMV”.
 - If this tag is 0x00 and the reader was configured to read MSR data as primary, use “MagStripe”.
 - If this tag is 0x01 (technical fallback) or 0x81 (MSR fallback), use “MagStripe”.
 - For eDynamo readers:
 - When reading EMV data, use “EMV”.
 - When reading MSR data, you cannot use the ProcessEMVSRED method – please disregard this document, reference the information for the ProcessCardSwipe method in the MPPG API reference, and use the ProcessCardSwipe method instead..
 - **TransactionType** - “SALE” or “AUTHORIZE”.
- ProcessReferenceID

⁵ Using ARQC is the recommended data to use for this element, as EMV tag DFDF59 may be missing some data that is required by some processors.

- **TransactionID⁶** – The TransactionID returned from the original transaction.
 - *WorldPay only (required):* If processing a refund, provide the transaction ID from the SALE transaction or the transaction ID from the CAPTURE transaction of an AUTHORIZE/CAPTURE transaction pair.
- **TransactionType** - “CAPTURE”, “VOID” or “REFUND”.
- **ProcessToken**
 - **Token** – The string returned from an earlier MPPG transaction containing card data required to process **VOID**, **REFUND** or **CAPTURE** for certain processors. See Table C: Input Methods.
 - **TransactionType** - “CAPTURE”, “VOID”, or “REFUND”.

2.2.3.3 Processor-Dependent Input Data

The following additional data is either required, recommended, or optional for certain processors. In each case, the data is added as a key/value pair in the <TransactionInputDetails> section of the SOAP request. The listed identifier is placed in a <key> element, and the <data> element is populated with the appropriate value.

- **deviceID** – *TSYS only, SALE/AUTHORIZE ONLY.* If your account has not been set up with a static device ID to be automatically applied to every transaction, or if you want to override the default device ID with another value, then you will need to specify the device ID to be used with your transaction.
- **priorAuthCd** – *Chase only (required), CAPTURE ONLY* – The authorization code from the previous corresponding AUTHORIZE transaction.
- **CardInputCode** – *WorldPay only (required).*
 - **SALE/AUTHORIZE (ProcessEMVSRED)**
 - If the EMV tag DFDF52 (card type) is 0x01 (card swiped), then set this value to “2”.
 - If the tag is 0x05 (EMV card was inserted), then set this value to “6”.
 - If the tag is 0x06 or 0x08 (EMV contactless card was tapped), then set this value to “7”.
 - If a technical fallback read is being performed (please see section 2.1.1.4 FALLBACK as well as information about PaymentMode in section 2.2.3.2 Method-Dependent Input Data), then set this value to “9”.
 - **AUTHORIZE (ProcessReferenceID)** – *Visa and Mastercard cards only.* You can perform an incremental authorization against an initial authorization by sending an AUTHORIZE request using the ProcessReferenceID method and setting the <CardInputCode> value to the value that was set for <CardInputCode> in the original AUTHORIZE request. When you use this method to request an AUTHORIZE transaction, the amount you specify will be added to the original authorized amount. For example, if the original authorized amount was \$10.00 and you want to increase the authorized amount to \$15.00, use this AUTHORIZE method with an amount of \$5.00 to increase the authorization amount.
If you attempt to perform this method of authorization against a card that is not from Visa or Mastercard, the request will be declined. If the new authorization amount goes higher than the available credit on the account, the request will be declined.

⁶ The Transaction ID can be referenced for up to 30 to 90 days from the date of the original transaction, depending on the processor. For the specific limitations for your processor, please contact your processor’s representative.

- CAPTURE – If EMV tag DFDF52 in the original authorization transaction to be captured was 0x01 then set this value to “2”; if the tag was 0x05, then set this value to “6”; if the tag was 0x06 or 0x08, then set this value to “7”; if the original transaction was a technical fallback, then set this value to “9”.
- **LastRecordNumber** – *Elavon only (required)*. An incremental, numeric value that indicates the position of a transaction on the Elavon log for the current processing day. (See your Elavon representative for a definition of the current processing day.) Calculate the value as followed, based on the transaction type being executed:
 - SALE/REFUND – For the first transaction of the day, use ‘0’; otherwise, use the <RecordNumber> returned in the processor response from the previous transaction.
 - VOID – Use the <RecordNumber> returned in the processor response from the previous transaction, which *must* be a SALE transaction in the same processing day.
- **Merchant Category Code (MCC)** – *First Data only (strongly recommended)*. By default, the MCC value is set to retail if no MCC information is included as part of the message. If the merchant’s category is something other than retail (e.g. restaurant), then the message needs to include the MCC that was assigned to the merchant in the merchant’s VAR sheet.
- **Merchant soft descriptors** – *First Data/WorldPay only, SALE/AUTHORIZE ONLY (optional)*. Soft descriptors allow a merchant to specify the merchant information a cardholder will see in the transaction description on their credit card statement. The following keys and their respective values should be added in the TransactionInputDetails block for any information that is to be changed from the default values:
 - First Data
 - MerchName – Merchant name, 38 characters max
 - MerchAddr – Merchant address, 30 characters max
 - MerchCity – Merchant city, 20 characters max
 - MerchState – Merchant state, 2 characters max
 - MerchPostalCode – Merchant ZIP code, 9 characters max, alphanumeric and spaces only
 - WorldPay
 - MerchantDescriptor – Merchant name, 25 characters max
 - MerchantDescriptorCity – Merchant city, 13 characters max
 - MerchantDescriptorState – Merchant state, 2 characters max
- **Partial Authorization** – *Chase/Elavon/First Data/WorldPay only, SALE/AUTHORIZE only (optional)*. If you wish to change from the default value used for your processor, then you will need to insert one of the following sets of additional tags:
 - Chase: key=<pay:partialAuthInd>, value=“N” for off (default) or “Y” for on
 - Elavon: key=<Partial_Auth_Indicator>, value=“0” for off (default) or “1” for on
 - First Data: key=<PartAuthrznApprvlCapabl>, value=“0” for off (default) or “1” for on
 - WorldPay: key=<PartialApprovedFlag>, value=“0” for off or “1” for on (default)
- **ProcessorResponse** – *First Data only, CAPTURE/VOID only (required)*. The complete response from the processor for the referenced transaction.
- **ResponseCode** – *Elavon only, VOID only*. The response code is returned in the processor response for a SALE transaction, and must be sent for VOID transactions.
- **ReversalType** – *WorldPay only, VOID only*. If you want to set the new authorized amount for a previous AUTHORIZE transaction by specifying the new amount (instead of specifying the amount to subtract), include the key <ReversalType> and the value 2 in the <TransactionInputDetails> element. For example, if the original authorized amount was \$10.00, and you want to set the new authorized amount to \$8.00, then include the <ReversalType> key/value pair and set the amount in your VOID request to \$8.00.

- **Tip** – *TSYS only, CAPTURE only*. For CAPTURE transactions, the <Amount> element is not included at all in the request; instead, the <tip> element is included and the amount of the tip only (not including the base amount) is included.
- **TipAdjust** – *Heartland only, CAPTURE only*. For CAPTURE transactions, the <Amount> element holds the base plus tip amount, and the <TipAdjust> element is included and contains the string “CreditTxnEdit” to indicate that this transaction includes a tip adjustment.

2.2.4 Method Outputs

MPPG provides output data from three sources: the card-issuing financial institution, the merchant processor, and the MPPG service itself. Generally, all output data follows a similar format, though some data may not be present in all calls and not all data may be of interest to the user. The listing below includes data most commonly used:

2.2.4.1 Data from the Card-Issuing Financial Institution

- **AuthCode** – Authorization Code. If the transaction is approved by the financial institution, this numeric field is returned and is unique to this transaction.
- **AuthorizedAmount** – The amount of funds authorized by the financial institution. The amount may be less than that requested. If so, the merchant must ask the cardholder for the difference in funds via a different payment type (such as cash) or a different card. If the cardholder does not wish to provide the additional funds, the transaction may be deleted using VOID.

2.2.4.2 Data from the Merchant Processor

- **IsTransactionApproved** – A boolean representing the approval status of the transaction.
- **TransactionID** – A String to be used, as applicable, with transaction types CAPTURE, VOID, or REFUND.
- **TransactionMessage** – A message (String) from the merchant processor concerning the status of this transaction.
- **TransactionStatus** – A code (String) from the Merchant Processor concerning the status of this transaction.
- **ProcessorResponse** – A String from the merchant processor that contains additional data concerning the transaction. This data is required by First Data for CAPTURE, VOID, and REFUND. However, its contents may also be of general interest to the user.

2.2.4.3 Data from the MPPG Service

- **CustomerTransactionID** – Echoed from the input.
- **MagTranID** – A unique, internal value assigned by MPPG to identify this interaction.
- **EMVSREDDDataMasked** – a TLV-encoded container of EMV response data that is used to create a customer receipt.
- **Token** – A String returned by **ProcessCardSwipe** and **ProcessEMVSRED** for use with **ProcessToken**. It is a secured representation of the card data presented to the reader in this transaction.

- **IsReplay** – a Boolean indicating if the KSN for the reader cryptogram has been processed before. A value of 'true' may indicate an attempt to reprocess, in this transaction, identical reader data already processed to MPPG in the past.
- **CardID** – A String uniquely identifying the card presented for this transaction. From this, the user can determine if they have accepted this card either in the past or through a different channel. This value cannot be used to process a transaction and cannot be transmuted into the original card data.

2.3 Receipts

EMV regulations require merchants provide to cardholders the option of receiving a paper or electronic receipt for an approved **SALE**, **AUTHORIZE** or **REFUND** transaction, formatted with specific data. The following lists items required in various transaction type scenarios, however not all items are required for each, as noted. EMV tag data can generally be found in **EMVSREDDataMasked**.

SALE

- 1: Merchant Name
 - 2: 123 Anywhere Street
 - 3: Merchantville, CA, 12345
 - 4: 123-456-7890
 - 5: SALE
 - 6: Term: 001
 - 7: Ref Num: 1228171757
 - 8: 12/29/17 2:14PM
 - 9: Auth Code: 400101
 - 10: AMEX
 - 11: XXXXXXXXXXXXX1003
 - 12: CHIP
 - 13: AMOUNT: \$39.50
 - 14: X_____
 - 15: AID: A000000025010801
 - 16: AMERICAN EXPRESS
 - 17: AMERICAN EXPRESS
 - 18: TVR: 8040008000
 - 19: TSI: 6800
- 1: The merchant's name
 - 2-3: The merchant's address
 - 4: The merchant's telephone number
 - 5: SALE

- 6: The terminal ID, as provided by the Merchant Processor and used when boarding this merchant on MPPG.
- 7: **CustomerTransactionID** generated by the user or the **TransactionID** returned by MPPG.
- 8: The local date and time of the transaction.
- 9: **AuthCode**.
- 10: The brand of the card presented by the cardholder. This can be determined by analyzing EMV tag 9F06 and matching its contents using a regular expression as follows (for use with Java, other platforms may require minor modifications to syntax):
 - VISA: “^.*((31010)|(32010)|(33010)).*\$”
 - MASTERCARD: “^.*((41010)|(43060)).*\$”
 - DISCOVER: “^.*1523010.*\$”
 - AMEX: “^.*2501.*\$”
- 11: The masked card number. This can be found in the output data of the reader (see the reader’s Programmer’s Reference Manual), or for some merchant processors, in **ProcessorResponse**.
- 12: For all readers other than eDynamo, Inspect EMV tag DFDF52 in the initial reader output for the card type. If the card type is ‘0x01’, print MAGSTRIPE, otherwise, print ‘CHIP’. For eDynamo, if a magstripe was processed, print MAGSTRIPE, otherwise print ‘CHIP’.
- 13: **AuthorizedAmount**.
- 14: The cardholder verification method: Signature (sample signature line shown; required for **SWIPE**, optional for **INSERT** and **TAP**), or “PIN VERIFIED” (if EMV tag 99 exists for a DynaPro family device, see Table B: Input Parameters for PIN support at various processors).
- 15: The EMV Application Identifier (“AID”), found in EMV tag 9F06.
- 16: The EMV Application Preferred Name, found in EMV tag 9F12. The tag contains the byte encoded application name, per ISO/IEC 8859-1 (ASCII equivalent). Decode for display on the receipt. If the tag does not exist, print nothing.
- 17: The EMV Application Label, found in EMV tag 50. The tag contains the byte encoded application label, per ISO/IEC 8859-1 (ASCII equivalent). Decode for display on the receipt. If the tag does not exist, print nothing.
- 18: EMV tag 95 (Terminal Verification Results or “TVR”).
- 19: EMV tag 9B (Transaction Status Information or “TSI”).

AUTHORIZE to process a restaurant tip

Same as **SALE**, with the addition of:

TIP: _____
Total: _____

It is recommended a signature line always be presented for this transaction.

REFUND

1: Merchant Name
2: 123 Anywhere Street
3: Merchantville, CA, 12345
4: 123-456-7890

5: REFUND

6: Term: 001

2 - Operational Overview

7: Ref Num: 1228171757

8: 12/29/17 2:14PM

9: Auth Code: 400101

10: AMOUNT: \$39.50

- 1: The merchant's name
- 2-3: The merchant's address
- 4: The merchant's telephone number
- 5: REFUND
- 6: The terminal ID, as provided by the Merchant Processor and used when boarding this merchant on MPPG.
- 7: **CustomerTransactionID** generated by the merchant or the **TransactionID** returned by MPPG.
- 8: The local date and time of the transaction.
- 9: **AuthCode**.
- 10: The amount refunded.

3 Integration and Testing Preparation

3.1 Integration Preparation

New POS integrator customers should visit www.magtek.com to arrange contact with a MagTek sales representative. They will assist with the following steps to begin integrating a POS application with a MagTek EMV reader and MPPG.

- Choose the best reader model for the application
- Determine the reader interface method best suited to the application
 - Access MagTek SDKs: driver libraries and software or MagneFlex
- Obtain readers for integration and testing
- Access documentation and samples
 - Programmer's Reference Manual for the selected reader(s).
 - Interface documentation and software (SDK or MagneFlex)
 - MPPG Programmer's Reference Manual
 - MPPG Input Samples
- Execute contracts with Magensa to begin the MPPG customer onboarding process, receive additional integration information, and access integration support through integration@magensa.net.

3.2 Testing Preparation

The following is required to test a reader/POS application/MPPG integration:

- Test EMV cards – links to purchase test cards can be found in the onboarding documentation.
- PILOT account credentials – allows access to the PILOT MPPG system to test transactions.
- Application configuration:
 - Verify that your application has access to the proper values for CustomerCode, Username, and Password.
 - Verify that the ProcessorName has been set correctly.
 - Verify that your MagTek reader is connected to the system running your application and ready for operation.
 - Verify that you have your EMV test cards at hand.
 - Your application must retain the following information in order to be able to perform a CAPTURE, VOID, or REFUND on a previous transaction (see the CAPTURE column of Table C: Input Methods for reference):
 - Transaction amount
 - TransactionID (*Chase/Heartland/TSYS/WorldPay*)
 - Transaction token (*Chase/Elavon/First Data*)
 - AuthCode (*Chase only*)
 - The full XML response from the processor that was returned in the key/value pair <ProcessorResponse> (*First Data only*)
 - Merchant Category Code sent to the processor (*First Data only*)
- Application logs – Part of the certification process is to submit the XML/SOAP requests and responses that go through your application as well as the ARQC data that was read from the Magensa reader, so please be sure to include support in your application for capturing this data.

4 Certification

The certification process is intended to assist the POS integrator and Magensa integration support in determining that the integration project has been successful. It is not intended as a Quality Assurance process or a guarantee of error-free processing if successfully completed, but rather as a baseline measure of the integrator's ability to successfully communicate with the MPPG service. The certification scripts are in the following sections.

Once certification testing has been completed, the customer may request production credentials as well as documentation concerning merchant onboarding.

4.1 Retail Certification

Your application must be able to successfully process a SALE transaction and then a REFUND transaction against that sale in order to be certified as having met the baseline requirements to process transactions for retail installations.

4.1.1 SALE (Chase/Elavon/First Data/Heartland/TSYS/WorldPay)

For this procedure, the base transaction amount will be \$12.34.

- 1) Verify that your application populates the ProcessEMVSRED XML/SOAP message with the following values:
 - a) EncryptionType = (EMV tag DFDF57)
 - b) EMVSREDDData = (ARQC data⁷ or EMV tag DFDF59)
 - c) KSN = (EMV tag DFDF56)
 - d) NumberOfPaddedBytes = (EMV tag DFDF58)
 - e) PaymentMode = EMV
 - f) TransactionType = SALE
 - g) Additional processor-specific elements that are appropriate for the specified processor, as described in section 2.2.3.3 Processor-Dependent Input Data.
- 2) Select SALE operation from your application.
- 3) Enter \$12.34 as the amount for this transaction.
- 4) When prompted to present your test EMV card, insert the card into the reader.
- 5) When your application receives the EMV card data, it should build the XML/SOAP request message and send the request to the MPPG service; the MPPG service should then send an XML/SOAP response message.
- 6) Save your log of the XML/SOAP request and response so that you can submit them for certification.

⁷ See footnote 5 on page 14.

4.1.2 REFUND

Please select one of the REFUND process(es) in this section that is appropriate for the processor(s) that are supported by your application.

NOTICE

Chase retail transactions can be refunded by reference ID or by token, whereas Chase restaurant transactions can only be refunded by token.

4.1.2.1 REFUND By Reference ID (Chase/Heartland/TSYS/WorldPay)

REFUND transactions that use reference IDs must use the ProcessReferenceID method to send the transaction request to the MPPG service.

- 1) Verify that your application populates the ProcessReferenceID XML/SOAP message with the following values:
 - a) Reference ID from SALE response
 - b) Authorized amount from SALE response, or sale amount from SALE request if the response's authorized amount element is empty
 - c) Additional processor-specific elements that are appropriate for the specified processor, as described in section 2.2.3.3 Processor-Dependent Input Data.
- 2) Select REFUND operation from your application.
- 3) Select the previous transaction that will be used to request a REFUND transaction.
- 4) Confirm the refund amount for this transaction.
- 5) Start the REFUND operation. The application should build the XML/SOAP request message and send the request to the MPPG service; the MPPG service should then send an XML/SOAP response message.
- 6) Save your log of the XML/SOAP request and response so that you can submit them for certification.

4.1.2.2 REFUND By Token (Chase/Elavon/First Data)

REFUND transactions that use tokens must use the ProcessToken method to send the transaction request to the MPPG service.

- 1) Verify that your application populates the ProcessToken XML/SOAP message with the following values:
 - a) Token from SALE response
 - b) Authorized amount from SALE response, or sale amount from SALE request if the response's authorized amount element is empty
 - c) Additional processor-specific elements that are appropriate for the specified processor, as described in section 2.2.3.3 Processor-Dependent Input Data.
- 2) Select REFUND operation from your application.
- 3) Select the previous transaction that will be used to request a REFUND transaction.

- 4) Confirm the refund amount for this transaction.
- 5) Start the REFUND operation. The application should build the XML/SOAP request message and send the request to the MPPG service; the MPPG service should then send an XML/SOAP response message.
- 6) Save your log of the XML/SOAP request and response so that you can submit them for certification.

4.2 Restaurant Certification

The following processors can be used for the restaurant industry:

- Chase
- First Data
- Heartland
- TSYS
- WorldPay

Elavon does not support AUTHORIZE and CAPTURE, so this processor cannot be used for the restaurant industry.

Your application must be able to successfully process an AUTHORIZE transaction, a CAPTURE transaction against the authorization, and then a REFUND transaction against the CAPTURE transaction in order to be certified as having met the baseline requirements to process transactions for restaurant installations.

4.2.1 AUTHORIZE (Chase/First Data/Heartland/TSYS/WorldPay)

For this procedure, the base transaction amount will be \$56.78.

- 1) Verify that your application populates the ProcessEMVSRED XML/SOAP message with the following values:
 - a) EncryptionType = (EMV tag DFDF57)
 - b) EMVSREDDData: ARQC data⁸ or EMV tag DFDF59
 - c) KSN: EMV tag DFDF56
 - d) NumberOfPaddedBytes: EMV tag DFDF58
 - e) PaymentMode = EMV
 - f) TransactionType = AUTHORIZE
 - g) (*First Data only*) Merchant Category Code = (your MCC)
- 2) Select AUTHORIZE operation from your application.
- 3) Enter \$56.78 as the amount for this transaction.
- 4) When prompted to present your test EMV card, insert the card into the reader.

⁸ See footnote 5 on page 14.

- 5) When your application receives the EMV card data, it should build the XML/SOAP request message and send the request to the MPPG service; the MPPG service should then send an XML/SOAP response message.
- 6) Save your log of the XML/SOAP request and response so that you can submit them for certification.

4.2.2 CAPTURE

For this procedure, the tip amount to capture will be \$13.22 that will be added to the base amount of \$56.78, for a total of \$70.00.

4.2.2.1 CAPTURE By Reference ID (Heartland/TSYS/WorldPay)

- 1) Verify that your application populates the ProcessReferenceID XML/SOAP message with the following values:
 - a) Reference ID from AUTHORIZE response
 - b) TransactionType = CAPTURE
 - c) Additional processor-specific elements that are appropriate for the specified processor, as described in section 2.2.3.3 Processor-Dependent Input Data.
- 2) Select CAPTURE operation from your application.
- 3) Select the previous transaction that will be used to request a CAPTURE transaction.
- 4) Enter the amount for this transaction (for more information, please see Table B: Input Parameters):
 - a) *(TSYS only)* Enter \$13.22 as the tip amount; this amount will be added to the base amount of \$56.78 that was collected during the AUTHORIZE transaction.
 - b) *(WorldPay only)* Enter \$70.00 as the total amount (base + tip) for the AUTHORIZE and CAPTURE transactions.
- 5) Start the CAPTURE operation. The application should build the XML/SOAP request message and send the request to the MPPG service; the MPPG service should then send an XML/SOAP response message.
- 6) Save your log of the XML/SOAP request and response so that you can submit them for certification.

4.2.2.2 CAPTURE By Token (Chase/First Data)

- 1) Verify that your application populates the ProcessToken XML/SOAP message with the following values:
 - a) Token from AUTHORIZE response
 - b) TransactionType = CAPTURE
 - c) Additional processor-specific elements that are appropriate for the specified processor as described in section 2.2.3.3 Processor-Dependent Input Data:
 - i) *(Chase only)* Authorization code from AUTHORIZE response
 - ii) *(First Data only)* Merchant Category Code (MCC) used in AUTHORIZE request, and entire processor payload returned in AUTHORIZE response
- 2) Select CAPTURE operation from your application.
- 3) Select the previous transaction that will be used to request a CAPTURE transaction.

- 4) Enter \$70.00 as the total amount (base + tip) for the AUTHORIZE and CAPTURE transactions (for more information, please see Table B: Input Parameters).
- 5) Start the CAPTURE operation. The application should build the XML/SOAP request message and send the request to the MPPG service; the MPPG service should then send an XML/SOAP response message.
- 6) Save your log of the XML/SOAP request and response so that you can submit them for certification.

4.2.3 REFUND

Please select one of the REFUND process(es) in this section that is appropriate for the processor(s) that are supported by your application.

4.2.3.1 REFUND By Reference ID (Heartland/TSYS/WorldPay)

Please see section 4.1.2.1 REFUND By Reference ID (Chase/Heartland/TSYS/WorldPay) for the procedure to follow. Note that Chase refunds for restaurant must be done by token; please see the next section for Chase refunds.

4.2.3.2 REFUND By Token (Chase/First Data)

Please see section 4.1.2.2 REFUND By Token (Chase/Elavon/First Data) for the procedure to follow.

5 Reporting Test Results

After you have completed your tests for the environment in which your solution will operate, please send the following information for review and baseline certification for operation with the MPPG service:

- Your username
- Your customer code
- Retail:
 - Text file containing XML/SOAP request and response messages for SALE
 - Text file containing XML/SOAP request and response messages for REFUND
- Restaurant:
 - Text file containing XML/SOAP request and response messages for AUTHORIZE
 - Text file containing XML/SOAP request and response messages for CAPTURE
 - Text file containing XML/SOAP request and response messages for REFUND

Appendix A

If you receive FaultCode and/or FaultReason elements, then your request message was received but not accepted by the MPPG service. Please review the following sections for the causes of some of the more typical error responses; for a list of error responses that are not included here, please see section 2 of the MPPG programmer's reference.

NOTICE

This is not a complete and exhaustive list of every possible cause behind error codes. This list's purpose is only to provide guidance in identifying the most likely contributing factors. If you don't find a resolution to your error after reviewing this list and implementing the recommended remediation(s), please carefully check your XML/SOAP request against the section of the MPPG programmer's reference that describes the parameters for the method you are using.

A.1 Fault Error Codes

A.1.1 FaultCode: 602 (KSN is required)

The KSN element is empty. Please enter the KSN from the data that was read from your MagTek reader.

A.1.2 FaultCode: 603 (CustomerCode is required)

The CustomerCode element is empty. Please enter the CustomerCode from your onboarding documentation.

A.1.3 FaultCode: 604 (Username is required)

The Username element is empty. Please enter the Username from your onboarding documentation.

A.1.4 FaultCode: 605 (Password is required)

The Password element is empty. Please enter the Password from your onboarding documentation.

A.1.5 FaultCode: 619 (Amount is not valid)

An invalid amount (for example, zero) was entered for the transaction. Please verify that the amount you entered was correct, especially making sure that there are no decimal points in amounts for Chase, Elavon, or First Data (and decimal point must be included for WorldPay).

A.1.6 FaultCode: 701 (Access Denied)

The most likely reason to receive this response is that at least one of the Username, Password, or CustomerCode values are missing or incorrect. Please check these values against your MPPG onboarding documentation to verify that your credentials are correct.

Another possible cause is that the MPPG account is not yet active. If you have verified that your Username, Password, and CustomerCode all match the values found on your onboarding documentation, please contact Magensa support to confirm whether your account has been activated.

A.1.7 FaultCode: 708 (EncryptionType is required)

There is no value associated with the EncryptionType element. Please be sure to enter the value from EMV tag DFDF57 for this element.

A.1.8 FaultCode: 712 (Unknown Error)

One possible cause for this error is that either the EMV data or the KSN was not entered correctly. The most likely reason is that part of the hexadecimal string was truncated from the string; please verify that the EMV and KSN data returned from the reader match exactly with the data being sent as part of the XML request.

Another possible cause for this error is that the reader that provided the KSN was not injected with one of the standard MagTek production keys, and the Magensa Hardware Security Module (HSM) does not have a copy of the reader's key to use for decrypting. The most likely reasons for this are (a) the reader in question was injected with a test key instead of a production key; or (b) the reader was injected with the customer's own key, but that key was not provided to MagTek for adding to the HSM.

A.1.9 FaultCode: 721 (Payload post failed)

The most likely reason for this error is that the Magensa services behind the MPPG service could not reach the destination processor. The processor may have either been experiencing either unexpectedly heavy traffic or an intermittent internet outage during the time that this error message was received from the MPPG service. However, reception of this message is verification that communication between your application and the MPPG service was completed successfully.

A.1.10 FaultCode: 738 (Invalid EMVSRED Data)

Either there is an error in the EMV data, or the value for the NumberOfPaddedBytes element is incorrect. Please check both values carefully against the data that was received from the MagTek reader.

A.1.11 FaultCode: 749 (Error in <token method>)

Decryption of the token data failed.

Error in TokensCallRedeemTokenWebService An unexpected error occurred.

The typical reason for this error is that the token value used in the XML/SOAP request message was truncated or otherwise incomplete. Please verify that the entire token is being passed in the token parameter.

Error in TokensCallRedeemTokenWebService Token verification failed

Please see the information in section 00.0.0.

A.1.12 FaultCode: 5000 (Unknown Error)

A typical reason for this error is that the processor name in the ProcessorName element has not been spelled correctly. The processor name's spelling must be exact, including use of matching uppercase and lowercase letters. Please check the spelling for your processor name against the name that was included in your onboarding documentation.

Another possible reason for this error is that non-hexadecimal characters were part of the EMVSREDData element. Please check that no characters outside of the range 0-9 and A-F are included in the string in the EMVSREDData element.

A.2 Other Errors

A.2.1 FaultCode: DeserializationFailed

The most likely reason for receiving this response is that at least one of the required values for the request is missing. Either the entire element and its associated value are missing completely, or the value is missing from the element. Please verify that you are including all of the required elements in your MPPG request message. For more information, please check the MPPG programmer's reference.

NOTICE

For MSR card swipes, you must include all three encrypted track elements, even if you have no track 1 or track 3 data. If you only have track 2 data, please include empty elements for track 1 and track 3.

If one of the XML tags is missing, incomplete, or out of sequence, then this error will be triggered. You can use an XML validation tool to check for these types of errors.

A misspelled value can also cause this error. For example, entering the string "SALES" instead of "SALE" in the <TransactionType> element will cause this error.

One other possible reason for receiving this response is if the data in the request message was not placed in the expected order. Please verify that the order of your XML tags in your request matches up with the order of XML tags in the example data.

For any deserialization error that occurs, please read the complete text in the fault code element as it may give you a hint regarding which element or value caused the issue.