MAGTEK
SECURITY FROM THE INSIDE

MagneFlex

DynaPro POS Application
User and Integration Manual

August 9, 2016

Manual Part Number:
D998200099-30

REGISTERED TO ISO 9001:2008

0. Table 0.1 - Revisions

Rev Number	Date	Name	Notes
10	April 15, 2016	Roger Applewhite	Initial Release
20	April 20, 2016	Phisa Kelleher	Updated Section 10
30	August 9, 2016	Phisa Kelleher	Added Section E - First Data EMV Receipt Requirements

CONFIDENTIAL

This document may not be reproduced or distributed. This document is for informational purposes only. Changes to this document may occur without notice.

Table of Contents

1 Introduction

MagneFlex for DynaPro is an application that, when used in conjunction with the MagTek family of DynaPro payment terminals and Magensa transaction processing services, provides a simplified and consistent interface for a POS Application to initiate and complete card-based payment transactions.

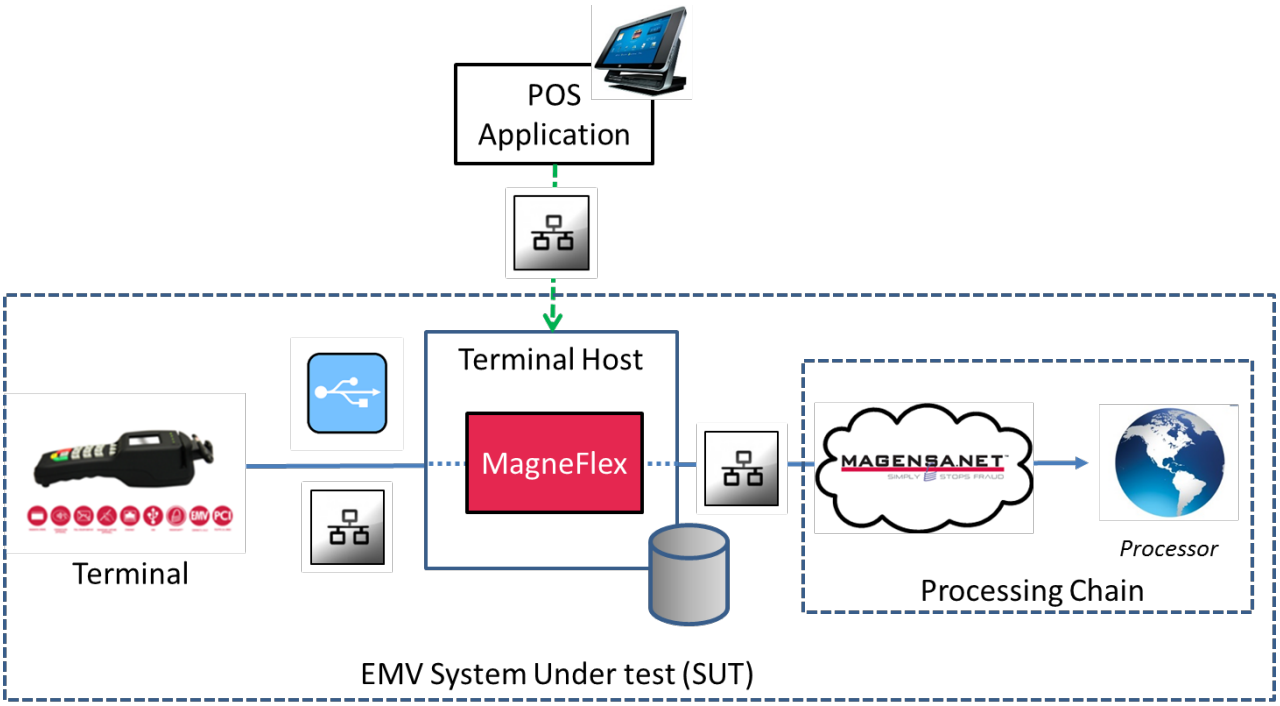
To maximize flexibility, **MagneFlex** is template-driven, using data translation techniques to seamlessly interoperate between the **MagneFlex** input API, the Terminal interface, payment card technologies (MSR, EMV, key-entered), and downstream Processor APIs.

This document is intended as both a guide to the **MagneFlex** architecture, a structured method for integrating a POS Application to MagTek and Magensa products, as well as a listing of available operations to effect Magnetic Stripe, EMV, and Reference transactions. It is therefore useful for payment system architects as well as software developers. In both cases, the organization integrating **MagneFlex** into a solution is called the “integrator”.

2 System Architecture

MagneFlex employs an architecture that contemplates the following elements of a payment system:

- **Terminal** – A MagTek DynaPro terminal. The Terminal includes magnetic stripe card reader, Integrated Circuit Card (ICC) reader, Near Field Communication (NFC) reader, PIN Entry Device (PED), EMV processing support, Secure Reading and Exchange of Data (SRED) support, and touchscreen user interface. Output data is encrypted under DUKPT/TDES.
- **Terminal Host** – a computer running Microsoft Windows 7 or later that sends commands to the Terminal via USB or a local IP network to drive it through operational steps required to effect a card read or other function. The Terminal Host runs the **MagneFlex** application as a local Windows service.
- **POS Application** – The application, provided by the integrator of **MagneFlex**, that will process a payment transaction through its steps from totaling items for purchase and calculating a payment transaction amount, to payment receipt generation and other post transaction tasks. The POS Application can be resident on the Terminal Host, or on a separate device that can access the Terminal Host via an IP network.
- **Processing Chain** – The downstream collection of decryption, gateway, merchant processing and card/payment authorization functions provided nominally by Magensa LLC and additional entities such as the merchant processor, card brand, card issuing bank, or even non-payment providers. For the purposes of this document, the endpoint called by Magensa is termed the 'Processor'. **MagneFlex** has two *operating modes* that utilize either the Magensa Payment Protection Gateway (MPPG) service, a traditional payments gateway, or Magensa's Decrypt and Forward service. Processors that are endpoints for **MagneFlex** must provide "host capture" services. Terminal capture is not supported.
- **MagneFlex** – An application package, operating as a local Windows web service on the Terminal Host that mediates transaction processing between the POS Application, Terminal, and the Processing Chain. Most task steps in the process are hidden or abstracted by **MagneFlex** so that the POS Application needs minimal awareness of both function and state during transaction processing. **MagneFlex** accepts JSON commands posted by the POS Application (REST), and operates as a multi-instance, multi-threaded service that can support transaction requests from multiple POS Applications simultaneously and drive multiple Terminals simultaneously. **MagneFlex** requires that the Terminal be configured for EMV online only. Please see appendix A.



This architecture not only supports modularity and flexibility for Payment system integrators, but also defines the elements that are generally certified by Processors as the System Under Test (SUT) for EMV certification. The architecture bears similarity to a canonical form often referred to in the payments industry as “software semi-integrated”. However, as there is no standardized definition of this form, MagTek makes no representations as to the fitness of **MagneFlex** in any organization’s definition of the term. However, **MagneFlex** is designed to be adaptable and extensible to a wide variety of payment system concepts and architectures, and can be used to solve a variety of architecture and integration issues.

3 Operating Modes

MagneFlex can function either as a front-end to Magensa's traditional payment gateway, the Magensa Payment Protection Gateway (MPPG) or it can pass transactions to Magensa's Decrypt and Forward system.

3.1 MPPG

In this mode, **MagneFlex** assumes the merchant has been boarded on MPPG by the integrator or their assignee, and will be using a Processor that was defined during that boarding process. MPPG maintains the interface to the Processor, as well as storing an account profile of the merchant, including all credentials and other information required by the Processor to accept a payment transaction request from the merchant. **MagneFlex**, in conjunction with MPPG, also maintains all required interface information required by the Processor in order to properly form a transaction request message per their specification. The POS Application *is not* required to be aware of any of this information or the interface to the Processor.

In addition, as the data from the Terminal is encrypted, MPPG performs a decryption function in Magensa's secure servers. This keeps any sensitive cardholder data produced by the Terminal encrypted until it arrives at Magensa. Therefore, no local system or network in front of Magensa, such as the merchant's systems or the Terminal Host, has access to unencrypted cardholder data. In addition, for certain Processors, Magensa will certify the Terminal, **MagneFlex** and the Processing Chain independent of the POS Application as the EMV SUT so that the integrator will be involved as little as possible in the certification process (or not at all). Magensa will provide a list of completed certifications from time-to-time.

To use this mode, integrators must be registered on MPPG as "resellers" and must board merchants onto MPPG. The details of obtaining a reseller account as well as details of the boarding operation are provided in other documents. Please contact Magensa for details.

3.2 Decrypt and Forward

In this mode, **MagneFlex** assumes that all data, including transaction formatting information (XML) and merchant credentials required by a downstream Processor or other entity, has been placed in **MagneFlex** XML Templates on the Terminal Host. **MagneFlex** uses this information to form a transaction message bound for the Processor. It sends the message to Magensa's Decrypt and Forward service, where cardholder data is decrypted and placed into the formed transaction message received from **MagneFlex**. It then calls the Processor using credentials provided by **MagneFlex**. The response is then returned to **MagneFlex** and then returned to the POS Application

Unlike MPPG, it is unnecessary for the integrator to board the merchant at Magensa. It simply stores the merchant credentials and other data in **MagneFlex**'s Configuration File. The integrator only requires an account on the Decrypt and Forward service to use **MagneFlex** with any merchant.

As with MPPG, no unencrypted cardholder data is found on any component of the system in front of Magensa.

With this model, it is necessary for the integrator to obtain the relationship with and manage the certification to the Processor. In addition, the integrator must create the **MagneFlex** XML Templates that contain the XML formatting of the message bound for the Processor. Therefore, the integrator is responsible for obtaining and utilizing the Processor's API. In essence, **MagneFlex** acts as a component of the merchant's or integrator's technology infrastructure that the integrator wishes to connect to one or more Processors.

This mode requires more effort and payment system knowledge on the part of the integrator, but can be adapted to a wide variety of Processors and processing functions, even ones that are not traditional Processors. This can provide a wider scope of functionality and endpoints than can be provided by MPPG. As MagneFlex and Magensa are treated as part of the integrator's infrastructure, the integrator controls the relationship with the Processor and thus manages functionality, certification schedules, and other operations directly with the Processor, not needing to rely on Magensa.

3.3 Demo

MagneFlex can simulate the Processing Chain locally so that integrators can learn the application without yet having obtained a relationship with a Processor. All transaction requests are returned as authorized or successfully completed, regardless of input.

4 Transaction Types

In the payment card industry, there are several recognized transaction types accepted by most Processors. **MagneFlex** and Magensa attempt to process one of these types through the processing chain when one of the operations **MagneFlex** provides is called by the POS Application. Generally speaking, **MagneFlex** assumes the Processor is operating in "Host Capture Mode", that is, settlement batches are collected, stored and processed by the Processor. **MagneFlex** and MagTek readers do not store settlement data for batch processing, what is sometimes referred to as "Terminal Capture Mode".

The following transaction types are available in **MagneFlex**:

- 1 – SALE: A request to a Processor to authorize and settle a purchase amount.
- 2 – AUTHORIZE: A request to a Processor to authorize funds only. To settle the funds later, a 3 – CAPTURE transaction must be processed.
- 3 – CAPTURE: Requests the Processor settle an open authorization for the amount provided.
- 4 – VOID: Requests the Processor cancel an outstanding authorization before it has been settled.
- 5 – REFUND: Requests the Processor refund a previously settled transaction.
- 6 – FORCE: Requests the Processor settle a transaction whose original authorization was not processed from a card. Normally for voice telephone authorized transactions.

Each transaction type above will have its own input data requirements. For MPPG, these requirements are controlled by the MPPG API. For Decrypt and Forward, they are controlled by the Processor, and must be reflected in the Decrypt and Forward templates discussed in a later section of this document.

MagneFlex processes a transaction when the POS Application initiates one of three available web service operations. Each operation gathers certain data from the Terminal and presented card and then launches the transaction type indicated. In general, any transaction type may be used with any **MagneFlex** operation. However, if the operation does not collect the required data for the transaction, the operation will fail. Below is a list of the operations and the data they generally collect:

- **ProcessCardSwipe** – Data from the magnetic stripe of a card.
- **ProcessEMVSRED** – Data from an EMV ICC. Due to the EMV process, this operation is constrained to "1 – SALE" and "2 – AUTHORIZATION" only.
- **ProcessReferenceID** – A reference ID presented to the operation by the POS Application. This operation is suited to transaction types that refer to earlier transactions. Therefore, "1 – SALE" and "2 – AUTHORIZATION", as they are initial transactions in the POS processing chain, are not appropriate for this operation.

5 Operations

MagneFlex operates as a JSON web service, using HTTP POST, listening to the Terminal Host's *localhost* address. The structure of the inputs and outputs are similar, so they will be listed once, with variations by operation noted in line.

5.1 Description

5.1.1 ProcessCardSwipe

Requests a magnetic stripe read (MSR) from the Terminal and forms a transaction message based on the Operating Mode configured and the transaction type requested. If the card swiped is an EMV ICC, it will still be processed as a magnetic stripe transaction. Therefore, if the system is certified and operating to accept EMV, this operation should not be used. ProcessEMVSRED should be used when both MSR and ICC are expected.
HTTP POST to localhost:9000/api/magneflexhost/ProcessCardSwipe

5.1.2 ProcessEMVSRED

Requests an MSR or ICC read, depending on what the cardholder presents. If the card is not EMV-capable, a traditional, non-EMV transaction will be processed. If the card is EMV-capable, an EMV transaction will be processed regardless of card read method (MSR, ICC, contactless).
HTTP POST to localhost:9000/api/magneflexhost/ProcessEMVSRED

5.1.3 ProcessReferenceID

Processes a transaction to the Processing Chain without data read from a card, but with a reference to an originating transaction. For example, this operation may be used to process a "3 – CAPTURE", a request to settle a previous authorization, by referring to the TransactionID of that authorization, not the card number used. The Terminal is not involved in this operation.
HTTP POST to localhost:9000/api/magneflexhost/processreferenceid

5.2 Input Fields

- AdditionalRequestData (O)[1] – A Key/Value pair structure for presenting additional data required by MagneFlex for future features[2].
- DeviceID(O)[*ProcessCardSwipe and ProcessEMVSRED only*] – The address the host uses to find the Terminal, prepended with the access method. In the case of USB, it is the Terminal Device ID. Example: USB://98D70CE31309160D. If Ethernet, the IP address of the Terminal. Example: IP://196.168.0.1. If this field is not used, the host will attempt to connect to the first Terminal it finds on USB. This field can also be passed to the Processor, in Decrypt and Forward mode, if desired.
- CardSwipeInput(O)[*ProcessCardSwipe only*]
    - CVV – Some Processors allow CVV2 to be submitted with an MSR transaction for extra security. Please contact your Processor.

[1] Optional fields, marked by '(O)', will be sent downstream by **MagneFlex** as presented. If the field is missing, its tag will not be sent. If it is included, the tag and value will be sent, regardless of the value, even if empty. For Decrypt and Forward, some Processors require the tag not be presented if its value is empty or null. Please refer to their requirements.
[2] This field may be used to override the Magensa login credentials stored in the configuration file. Please see "Credentials" in the Configuration File section.

    - ZIP - Some Processors allow cardholder zip code to be submitted with an MSR transaction for extra security. Please contact your Processor.
- Options(R)*[ProcessEMVSRED only]* – A transaction time configuration of the Terminal. See MagTek for further details. The nominal value is "0".
- TransactionInput
    - Amount(R) – Amount requested for the transaction. For Decrypt and Forward mode, this should correspond to the transaction Amount as defined by the Processor.
    - TransactionInputDetails(O) – A Key/Value pair structure for presenting additional data required by either MPPG or the XML Template for a particular Decrypt and Forward Processor. For MPPG, please see the MPPG API. For Decrypt and Forward, see the "Configuration File" and "XML Templates" sections for further details.
    - TransactionType(R) – The type of transaction, represented by number, to be sent to the Processor.
    - CashBack(R)[*ProcessEMVSRED only*] - If the POS Application needs to request a cashback amount, the value, <x.xx>, is placed here. If no cashback is needed, enter "0.00".
    - ReferenceAuthCode(O)[*ProcessReferenceID only*] – The Authorization Code from the original transaction, if your Processor requires it.
    - ReferenceTransactionID(R)[*ProcessReferenceID only*] – The TransactionID for the original transaction being referenced.

5.3 Output Fields

- CardSwipeOutput[*ProcessCardSwipe only*]
    - CardID – A tokenized representation of the swiped card. Can be used to identify the card in future transactions.
    - IsReplay – Indicates if the DUKPT key used to encrypt this transaction has already been presented to Magensa.
    - MagnePrintScore – The MagnePrint Score. MagnePrint is a unique magnetic stripe card authentication tool provided by MagTek and Magensa. Please refer to documentation available at www.magtek.com.
    - CRMToken – Reserved for future use.
    - AdditionalOutputData – Additional data that may be returned by the Processor in Decrypt and Forward mode.
- EMVSREDOutput[*ProcessEMVSRED only*]
    - CardID – A tokenized representation of the ICC card. Can be used to identify the card in future transactions.
    - IsReplay – Indicates if the DUKPT key used to create the SRED cryptogram for this transaction has already been presented to Magensa.
    - AdditionalOutputData – Additional data that may be returned by the Processor in Decrypt and Forward mode.
- CustomerTransactionID – A GUID identifying this **MagneFlex** transaction request. Created by **MagneFlex**.
- MagTranID – A GUID identifying this **MagneFlex** transaction request. Created by Magensa.
- TransactionOutput
    - CVVResult – The result of the CVV2 check if it was submitted in the input.

- AVSResult - The result of the zip code check if it was submitted in the input.
- AuthorizedAmount – The amount the card issuing institution authorized. In a partial authorization scenario, this may be less than the amount requested in the input.
- AuthCode – The authorization code for this transaction created by the card issuing bank.
- IsTransactionApproved – a boolean value indicating if the card issuing institution approved the transaction.
- IssuerAuthenticationData[*ProcessEMVSRED only*] – A string that contains the EMV AC2 response from the issuing institution. Provided for reference purposes.
- IssuerScriptTemplate1[*ProcessEMVSRED only*] – A string that contains the EMV issuer script 1 to be processed to the card. Provided for reference purposes.
- IssuerScriptTemplate2[*ProcessEMVSRED only*] – A string that contains the EMV issuer script 2 to be processed to the card. Provided for reference purposes.
- TransactionID – A string returned by the Processor identifying this transaction to them.
- TransactionMessage – A string returned by the Processor describing the state of this transaction at the Processor.
- TransactionOutputDetails – Additional output details in key/value format.
  - ProcessorResponse – The verbatim response from the Processor when using Decrypt and Forward mode.
  - BatchData[*ProcessEMVSRED only*]
- TransactionStatus – A string code indicating the status of the transaction. Returned by the Processor.

- TransactionUTCTimestamp
- AdditionalOuputData - Additional data that may be returned by the Processor in Decrypt and Forward mode.

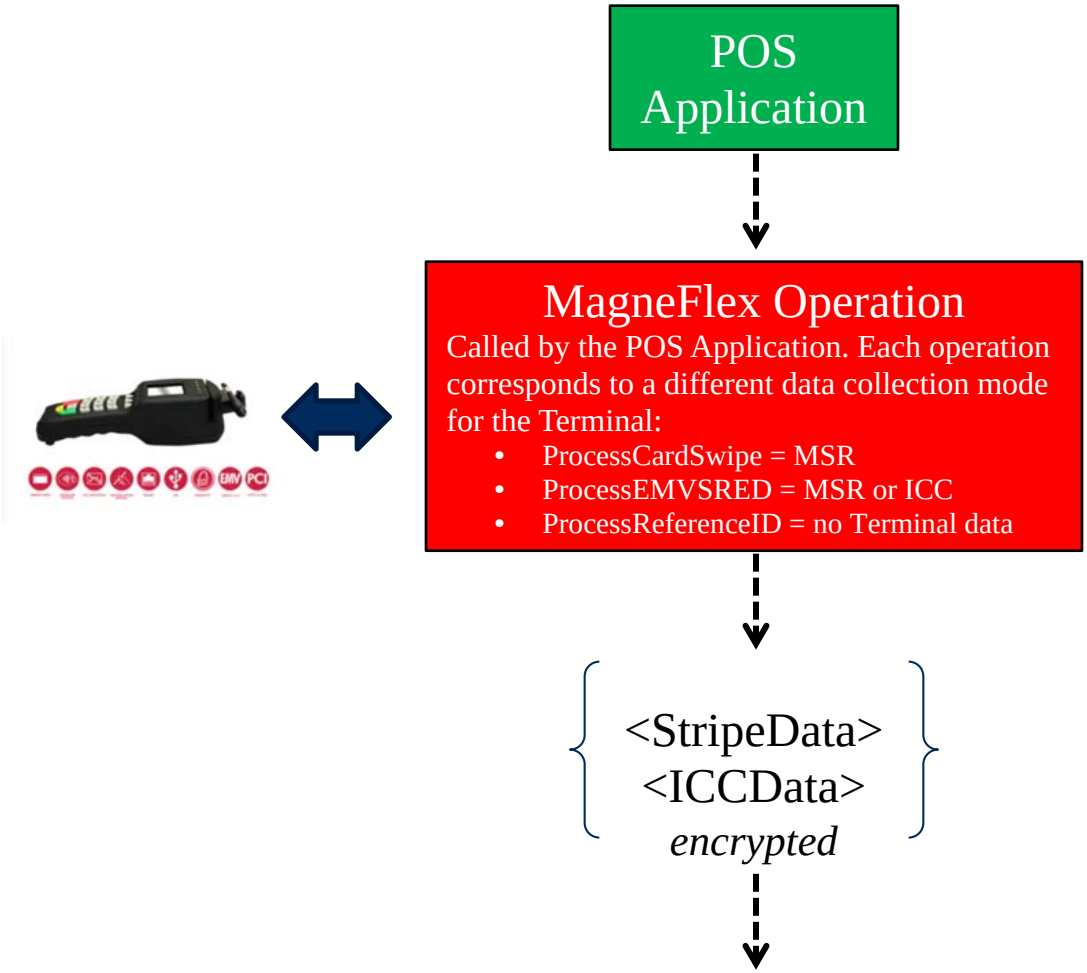
5.4 Exception

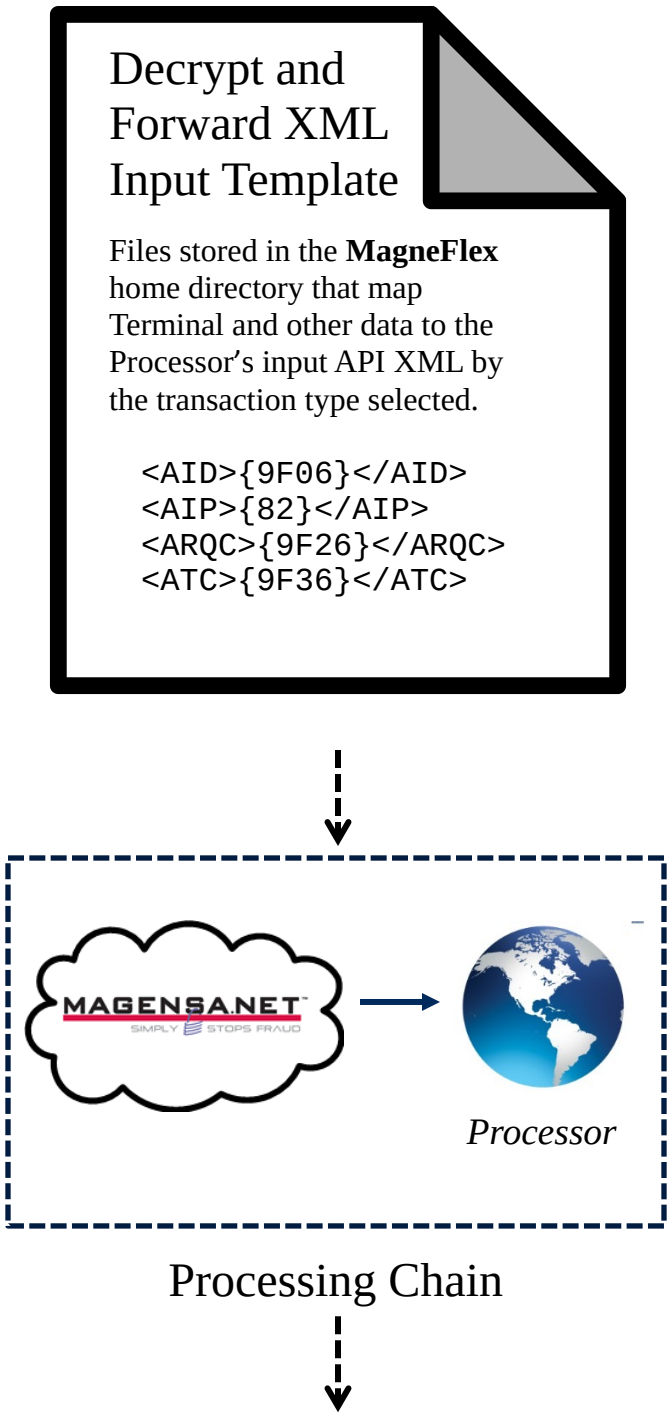
This is a generic exception class returned by **MagneFlex** if it encounters an internal error or an error is reported by the Terminal. See Appendix D for error messages.

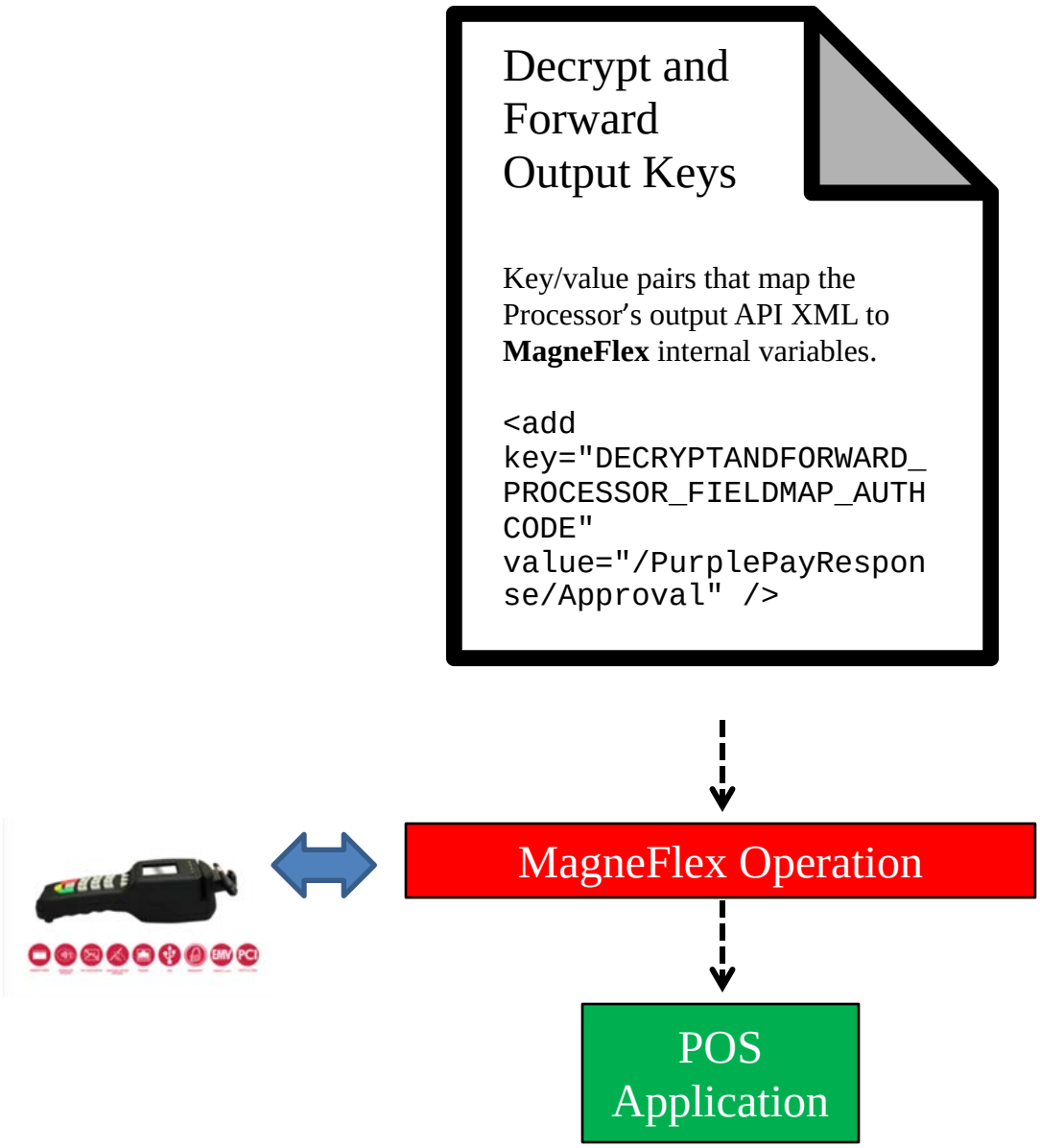
- faultcode(R): A numeric code indicating the fault type.
- faultstring(R): A message describing the fault.

6 Translation Scheme

The primary function of **MagneFlex** is to interoperate between the POS Application, the Terminal, the card technology presented (MSR, EMV, key-entered), and the Processor's API. To do this, **MagneFlex** *translates* between the POS Application, the Terminal, and the Processor employing keyed references and templates:




Decrypt and Forward XML Input Template
Files stored in the MagneFlex home directory that map Terminal and other data to the Processor's input API XML by the transaction type selected.
<AID>{9F06}</AID>
<AIP>{82}</AIP>
<ARQC>{9F26}</ARQC>
<ATC>{9F36}</ATC>
MAGENSA.NET
SIMPLY STOPS FRAUD
Processor
Processing Chain


Decrypt and Forward Output Keys
Key/value pairs that map the Processor's output API XML to **MagneFlex** internal variables.
<add key="DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_AUTHCODE" value="/PurplePayResponse/Approval" />
MagneFlex Operation
POS Application

6.1 Decrypt and Forward XML Input Templates

As shown above, Decrypt and Forward Mode requires maps from the Terminal's output (or the POS Application's input), for a certain transaction type, to the Processor's API XML structure. These maps are structured in **MagneFlex** as text files. Each of the defined combinations of operation and transaction type has its own template. **MagneFlex** finds these files on the Terminal Host's file system by examining keys in the Configuration File. The key name takes the structure "DECRYPTANDFORWARD_TEMPLATE_<operation>_<transactiontype>". The value associated with the key is the path (from the **MagneFlex** installation directory on the Terminal Host) and filename of the template.

Fallback for ProcessEMVSRED

As noted earlier in the document, ProcessEMVSRED may also be used with a non-ICC magnetic stripe card by swiping through the magnetic stripe reader, if the ProcessEMVSRED_CardType key in the configuration file is set to 3 or higher. When this is done, the reader detects the swiped card and processes in *fallback mode*. When in this mode, the reader accepts the swipe, but uses SRED[3] formatting to output the data. **MagneFlex** detects this and processes the transaction using the *fallback process*. The fallback process uses templates that are similar in construction to ProcessCardSwipe templates, but with additional information required to handle the SRED formatting. Please see Magensa support personnel for further information regarding the development of these templates.

The fallback process also handles the situation where a presented EMV card's chip fails. The reader, in this case, will then request the card be swiped. The resulting data will then be processed using fallback.

The templates available for **MagneFlex** include the following[4]:

- **ProcessCardSwipe**
  - Corresponding keys in the configuration file:
    - 1 – SALE: DECRYPTANDFORWARD_TEMPLATE_CARDSWIPE_SALE
    - 2 – AUTHORIZE: DECRYPTANDFORWARD_TEMPLATE_CARDSWIPE_AUTHORIZE
    - 3 – CAPTURE: DECRYPTANDFORWARD_TEMPLATE_CAPTURE
    - 4 – VOID: DECRYPTANDFORWARD_TEMPLATE_VOID
    - 5 – REFUND: DECRYPTANDFORWARD_TEMPLATE_REFUND
    - 6 – FORCE: DECRYPTANDFORWARD_TEMPLATE_FORCE
- **ProcessEMVSRED**
  - Corresponding keys in the configuration file:
    - 1 – SALE: DECRYPTANDFORWARD_TEMPLATE_EMV_SALE
    - 2 – AUTHORIZE: DECRYPTANDFORWARD_TEMPLATE_EMV_AUTHORIZE
    - DECRYPTANDFORWARD_TEMPLATE_EMV_REJECT[5]
    - DECRYPTANDFORWARD_TEMPLATE_EMV_FALLBACK_SALE
    - DECRYPTANDFORWARD_TEMPLATE_EMV_FALLBACK_AUTHORIZE

[3] Secure Reading and Exchange of Data.
[4] Not all transaction types can be used with all operations.
[5] This template may be needed by some Processors that require that **MagneFlex** automatically send an EMV reject transaction if the card or terminal declines any time after the First Card Action Analysis is completed.

        - DECRYPTANDFORWARD_TEMPLATE_EMV_FALLBACK_CAPTURE
        - DECRYPTANDFORWARD_TEMPLATE_EMV_FALLBACK_VOID
        - DECRYPTANDFORWARD_TEMPLATE_EMV_FALLBACK_REFUND
        - DECRYPTANDFORWARD_TEMPLATE_EMV_FALLBACK_FORCE
- ProcessReferenceID
    - Corresponding keys in the configuration file:
        - 3 – CAPTURE: DECRYPTANDFORWARD_TEMPLATE_CAPTURE
        - 4 – VOID: DECRYPTANDFORWARD_TEMPLATE_VOID
        - 5 – REFUND: DECRYPTANDFORWARD_TEMPLATE_REFUND
        - 6 – FORCE: DECRYPTANDFORWARD_TEMPLATE_FORCE

The data in a template can be static or dynamic. Static data is not intended to be changed across transactions. Decrypt and Forward will send it to the Processor verbatim. Dynamic data, however, is placed into the XML bound for the Processor at each transaction. Decrypt and Forward is directed to do this through "field replacement variables" in the template. These are formatted as {<variable name>}. Field replacement variables are labels for data available to Decrypt and Forward for insertion.

Overall, there are four field replacement variable data types that Decrypt and Forward can place into a Processor's XML that are sourced either from the Terminal's encrypted output, or from the POS Application's input directly:

- Required explicit data – These are the required input fields for an operation. For instance, ProcessEMVSRED requires "Amount" from the POS Application, which maps to the field replacement variable {Amount}. Names map directly from the input field to the field replacement variable.
- Optional explicit data – This is any data provided through "TransactionInputDetails" structure as shown in the input description for the operations. For example, the Processor may require the POS Application to submit a Transaction ID for an EMV SALE at transaction time. This could be done by the POS Application sending the following input to the ProcessEMVSRED operation:

  "TransactionInputDetails":[{"key": "{TransactionID}","value": "123456789"}]

  MagneFlex will then search the DECRYPTANDFORWARD_TEMPLATE_EMV_SALE template for the field replacement variable {TransactionID} and replace it with the value "123456789".
- Magnetic Stripe implicit data - Variables common to magnetic stripe card-based transaction processing that must be parsed from the Terminal's output after being decrypted. An example is the card number, whose field replacement variable is {CCNum}. See Appendix B for a list of variables available on the Decrypt and Forward API.
- EMV implicit data - EMV tags parsed from the Terminal's output after being decrypted. Values are mapped to field replacement variables with the same tag value. For instance, EMV tag 9F01 maps to {9F01}.

6.2 Decrypt and Forward Output Keys

XML data returned from the Processor through Decrypt and Forward is parsed by **MagneFlex** at nodes defined by key/value pairs in the Configuration File. The key names take on the following convention:

DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_<variable>_<operation>_<transactiontype>. The naming structure acts as a tree that **MagneFlex** traverses looking for a value to assign to the variable. The components "operation" and "transactiontype" can be omitted, which will cause the variable to be assigned the value regardless of operation and transactiontype used. The value itself is the node in the Processor's output XML where the needed data can be found. There are three types of variables:

- Defined output: These are the named variables in the operation output. An example is "AuthCode", which is represented by the key "DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_AUTHCODE" in the Configuration File. If the value of this key were "/PurplePayResponse/Approval", **MagneFlex** would find the AuthCode at that node in the Processor's (PurplePay, in this example) XML output. Some defined output is for POS Application use. Some output is required by **MagneFlex** to complete an EMV transaction with the Terminal.
- Additional variables: These are output variables presented through the "TransactionOutputDetails" structure. For instance, if the following were desired in the output,

  "TransactionOutputDetails":[{"key": "MyLuckyNumber","value": "987654321"}]
  a key of the value "DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_MYLUCKYNUMBER" would need to be in the Configuration File with a value that contained the node in the Processor's response where "MyLuckyNumber" could be found.
- **MagneFlex** Configuration – These are items, discussed further in the document, that are required for general operation of **MagneFlex**.

DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_PAYLOAD_<operation>_<transactiontype>: Some processors include nested Payloads in CDATA in their XML response. In order for MagneFlex to properly parse these nested Payloads, you must define the XML path to the nested Payload in this key. Sample value: "//Response//TransactionResponse//Payload"

7 Configuration File

MagneFlex requires a set of static configurations to be loaded into a file on the Terminal Host. The file contains key/value pairs used by **MagneFlex** for several purposes.

7.1 MagneFlex Configuration

These keys control the behavior of **MagneFlex**.

- PROTOCOL: The protocol used by the POS Application to call **MagneFlex**. Available values are "HTTP" and "HTTPS". See Appendix C for more information regarding using **MagneFlex** with HTTPS.
- ActivePaymentService: This key indicates the Operating Mode of **MagneFlex**: "DecryptAndForward" or "MPPG".
- BASEADDRESS: The host and port where **MagneFlex** listens for commands.
- DECRYPTANDFORWARD_PROCESSOR_DEMOMODE: If "true", places **MagneFlex** in demo mode. Demo fields and templates are included in the distribution.
- PERFORM_REJECT_TRANSACTION: If "true", automatically process a REJECT transaction if the Terminal or card declines an EMV transaction any time *after* the First Card Action Analysis is completed.

7.2 Terminal Management

These keys provide parameters **MagneFlex** sends to the Terminal to control its behavior. These keys are common to the DynaPro family of readers. For a detailed explanation of their use, please refer to the DynaPro SDK Reference.

- ProcessCardSwipe_DisplayMessage
- ProcessCardSwipe_Waitime
- ProcessCardSwipe_Tones
- ProcessEMVSRED_PINEntryWaitTime
- ProcessEMVSRED_ConfirmationWaitTime
- ProcessEMVSRED_Tones
- ProcessEMVSRED_CardType
- ProcessEMVSRED_ARPCTimeout
- ProcessTransaction_PINEntryWaitTime
- ProcessTransaction_ConfirmationWaitTime
- ProcessTransaction_Tones
- ProcessTransaction_CardType
- ProcessTransaction_ARPCTimeout

7.3 MPPG Mode Configuration

7.3.1 Credentials

These are the credentials of the merchant on MPPG:

- MPPG_UserName
- MPPG_Password
- MPPG_CustomerCode

7.4 Decrypt and Forward Mode Configuration

7.4.1 General Configuration Data

- DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_TRANSACTION_STATUS_SUCCESS: The explicit string value returned by the Processor if the transaction was successful.
- DECRYPTANDFORWARD_PROCESSOR_API: Processor web service message format type. Available value: "XML"
- DECRYPTANDFORWARD_URL_<operation>_<transactiontype>: The URL[6] to be called at the Processor for every combination of operation and transaction type[7]. Valid values of operation type are CARDSWIPE, EMV, EMV_FALLBACK, REFERENCEID. Valid transaction types include SALE, AUTHORIZE, CAPTURE, VOID, REFUND, FORCE. The <operation> and <transactiontype> can be omitted if the same URL is to be used by all.
- DECRYPTANDFORWARD_CUSTOM_HEADERS_<operation>_<transactiontype>: A text file(s) that contains any SOAP headers required by the Processors API for the operation and transaction type noted. If the headers are the same for all, <operation> and <transactiontype> can be omitted and a single file can be used. MagneFlex will treat it as the default. If the Processor accepts raw XML, this key can be ignored.
- EMV_TAG_ISSUER_AUTH_DATA: An optional element that specifies a specific EMV tag to send to the ICC.
- EMV_TAG_ISSUER_SCRIPT1_DATA: An optional element that specifies a specific EMV tag to send to the ICC.
- EMV_TAG_ISSUER_SCRIPT2_DATA: An optional element that specifies a specific EMV tag to send to the ICC.
- DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_FIXED_ISSUERAUTHENTICATIONDATA: An optional element to hard-code the ARPC to be sent back to the card.

[6] All URLs must be whitelisted on the Decrypt and Forward service at boarding. Please contact Magensa support for further details.
[7] Though, as noted earlier, not all combinations are valid.

7.4.2 Credentials

These are the credentials of the integrator on Decrypt and Forward.

- DECRYPTANDFORWARD_UserName
- DECRYPTANDFORWARD _Password
- DECRYPTANDFORWARD _CustomerCode

These fields can be overridden by using the AdditionalRequestData field as follows:

"AdditionalRequestData": [{
“key”: “Authentication”,
“value”:
"<Authentication><CustomerCode>CustomerCode</CustomerCode><UserName>Username</UserName><Password>Password</Password></Authentication>"
}]

7.4.3 Plug Data

This data is required for Decrypt and Forward when used with Operation ProcessReferenceID. Place it in the configuration file with values as shown.

- DECRYPTANDFORWARD_FIXED_TRACK1: value="65CCFABF7CA0BD0B9838068AC8D79CB79C056D20F5BCDFC13FFE07232ED1539E01BDFA299386090608ACB939231DACFA6E88D7AE77D9F4040DA0E360BA797834B5D80018E7ACC84D"
- DECRYPTANDFORWARD_FIXED_TRACK2: value="0B5F549A86201CD1CA33FF253C2F818A01CF54D1D2951170F55008C0F69156B2F8F33455D9287C29"
- DECRYPTANDFORWARD_FIXED_DEVICESERIAL: value="98EB0C8615101A0E"
- DECRYPTANDFORWARD_FIXED_KSN: value="950002000110AA20004D"
- DECRYPTANDFORWARD_FIXED_MAGNEPRINT: value="3818239FA4BFB7610B568826C44CFDBCFD40460B037AED3598FF3974AD9B6C7A46F4C059A17C3E9F2AD06D8B93C5A4DEA0D71C36A79F3E99"
- DECRYPTANDFORWARD_FIXED_MAGNEPRINTSTATUS: value="00000200"

8 Example Transactions

The following examples are provided to improve clarity into **MagneFlex** operation. The steps may not correspond to processing operations in the same order as presented here, though the effect is the same.

8.1 MPPG Mode

8.1.1 Step 1: POS Application calls MagneFlex

The POS Application requests an EMV Sale transaction be performed with the DynaPro. The operation to use for this transaction is ProcessEMVSRED. In this example, we assume the Processor requires a user-generated “TransactionID”. Therefore the TransactionInputDetails key/value array must be used. The key may be any string the integrator desires, as long as a corresponding field replacement variable is present in the XML Template for this operation for this Processor:

```
{
        "AdditionalRequestData": null,
        "DeviceID": null,
        "Options": 0,
        "TransactionInput":
        {
                "TransactionType": 1,
                "Amount": 1.00,
                "TransactionInputDetails":
                [
                        {
                                "key": "{TransactionID}",
                                "value": “123456789”
                        }
                ],
                "CashBack": 0.00
        }
}
```

8.1.2 Step 2: MagneFlex interacts with the Configuration File

MagneFlex first reads the following data from the Configuration File:

```
<add key="ActivePaymentService" value="MPPG" />
```

This indicates MPPG mode. Then **MagneFlex** reads the Terminal Management data corresponding to the EMV operation, ProcessEMVSRED:

```
<add key="ProcessEMVSRED_PINEntryWaitTime" value="20" />
<add key="ProcessEMVSRED_ConfirmationWaitTime" value="20" />
<add key="ProcessEMVSRED_Tones" value="1" />
<add key="ProcessEMVSRED_CardType" value="2" />
<add key="ProcessEMVSRED_ARPCTimeout" value="20" />
```

Finally, **MagneFlex** reads the MPPG credentials for this merchant:

```
<add key="MPPG_UserName" value="MPPGWSTestUser" />
<add key="MPPG_Password" value="password" />
```

```
<add key="MPPG_CustomerCode" value="9900000000000002" />
```

8.1.3 Step 3: MagneFlex interacts with DynaPro

MagneFlex calls the DynaPro and effects an EMV transaction. The DynaPro outputs encrypted and unencrypted data to **MagneFlex**

8.1.4 Step 4: MagneFlex calls Magensa's MPPG service

Magensa processes the transaction per the merchant's profile stored at Magensa. The response from the Processor is returned to **MagneFlex**

8.1.5 Step 5: MagneFlex completes the EMV transaction with the Terminal

As required by EMV specifications, certain data from the Processor must be sent to the Terminal for completion of EMV processing.

8.1.6 Step 6: MagneFlex returns response data to the POS Application

The output will be structured as follows:


```
{
	"MagTranID": "adbac28a-837c-4650-b3af-6e8a2b5f1fd4",
	"TransactionUTCTimestamp": "11/18/2015 1:14:49 AM",
	"CustomerTransactionID": "adbac28a-837c-4650-b3af-6e8a2b5f1fd4",
	"TransactionOutput":
	{
		"TransactionID": "000000000000000332",
		"TransactionStatus": "000",
		"TransactionMessage": "APPROVED",
		"AuthCode": "TEST57",
		"AVSResult": null,
		"IssuerAuthenticationData": null,
		"IssuerScriptTemplate1": null,
		"IssuerScriptTemplate2": null,
		"IsTransactionApproved": true,
		"CVVResult": null,
		"AuthorizedAmount": null,
		"TransactionOutputDetails": null
	}
	"EMVSREDOutput":
	{
		"AdditionalOutputData": null,
		"CardID": null,
		"IsReplay": false
	}
}
```

8.2 Decrypt and Forward Mode

8.2.1 Step 0: Configure Decrypt and Forward Templates

Before **MagneFlex** can be used in this mode with a particular Processor, templates must be created for each of the operation/transaction type combinations desired. The general steps are shown here. Contact Magensa support for further assistance.

1. Obtain the Processor's web service API.
2. Match the Processor's operations and transaction types to be used to corresponding operations and transaction types in **MagneFlex**.
3. Create the needed templates by replicating the desired Processor XML input for each operation/transaction type into the body of the template. Be sure to include any needed header information in the file denoted by DECRYPTANDFORWARD_CUSTOM_HEADERS.
4. Insert field replacement variables into the template for desired Terminal data (general and EMV) as well as POS Application data.
5. Enter key/value pairs into the Configuration File indicating the path to each template.
6. Examine the Processor's output schema to determine the location of needed return data. Enter key/value pairs into the Configuration File that indicate the name of the desired variable and the XML node in the Processor output where the variable may be found. The located data will be returned in the **MagneFlex** output. Three variables are available to be returned directly to the Terminal in EMV transactions:
    a. ISSUERAUTHENTICATIONDATA – Contains the ARPC or other data required by the card brand to be returned to the ICC
    b. ISSUERSCRIPTTEMPLATE1 – A string field to return an EMV issuer script to the ICC.
    c. ISSUER_SCRIPTTEMPLATE2 – A string field to return an EMV issuer script to the ICC.

8.2.2 Step 1: POS Application calls MagneFlex

The POS Application requests an EMV Sale transaction be performed with the DynaPro. The operation to use for this transaction is ProcessEMVSRED. In this instance, however, let us assume the Processor requires a cardholder address and a merchant password for every transaction[8]:

```
{
        "AdditionalRequestData": null,
        "DeviceID": null,
        "Options": 0,
        "TransactionInput":
        {
                "TransactionType": 1,
                "Amount": 1.00,
                "TransactionInputDetails":
                [
                        {
                                "key":"{MerchantPassword}",
```

[8] In step 0, it is assumed that the field replacement variables {MerchantPassword} and {Address} were placed in the Processor's XML in the template for EMV_SALE.

```
                        "value":"56^$3@3T"
                    },
                    {
                        "key":"{Address}",
                        "value":"123 4th Street"
                    }
                ],
                "CashBack": 0.00
            }
}
```


8.2.3 Step 2: MagneFlex interacts with the Configuration File

MagneFlex first reads the following data from the Configuration File:


```
<add key="ActivePaymentService" value="DecryptAndForward" />
```


This indicates Decrypt and Forward mode. Then **MagneFlex** reads the Terminal Management data corresponding to the EMV operation, ProcessEMVSRED:


```
<add key="ProcessEMVSRED_PINEntryWaitTime" value="20" />
<add key="ProcessEMVSRED_ConfirmationWaitTime" value="20" />
<add key="ProcessEMVSRED_Tones" value="1" />
<add key="ProcessEMVSRED_CardType" value="2" />
<add key="ProcessEMVSRED_ARPCTimeout" value="20" />
```


Next, **MagneFlex** reads additional configuration data for Decrypt and Forward:


```
<add key="DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_TRANSACTION_STATUS_SUCCESS"
value="000" />
<add key="DECRYPTANDFORWARD_PROCESSOR_API" value="XML" />
<add key="DECRYPTANDFORWARD_URL_EMV_SALE"
value="https://test1.purplepay.com/purplepay" />
<add key="DECRYPTANDFORWARD_CUSTOM_HEADERS" value="CustomHeaders.txt" />
```


Next, **MagneFlex** reads the credentials for this integrator of Decrypt and Forward at Magensa:


```
<add key="DECRYPTANDFORWARD_CustomerCode" value="9900000000000002" />
<add key="DECRYPTANDFORWARD_UserName" value="DAFWSTestUser" />
<add key="DECRYPTANDFORWARD_Password" value="Password" />
```


Next, **MagneFlex** reads the plug data:


```
<add key="DECRYPTANDFORWARD_FIXED_TRACK1"
value="65CCFABF7CA0BD0B9838068AC8D79CB79C056D20F5BCDFC13FFE07232ED1539E01BDFA29938
6090608ACB939231DACFA6E88D7AE77D9F4040DA0E360BA797834B5D80018E7ACC84D" />
<add key="DECRYPTANDFORWARD_FIXED_TRACK2"
value="0B5F549A86201CD1CA33FF253C2F818A01CF54D1D2951170F55008C0F69156B2F8F33455D92
87C29" />
<add key="DECRYPTANDFORWARD_FIXED_DEVICESERIAL" value="98EB0C8615101A0E" />
<add key="DECRYPTANDFORWARD_FIXED_KSN" value="950002000110AA20004D" />
```

<add key="DECRYPTANDFORWARD_FIXED_MAGNEPRINT"
value="3818239FA4BFB7610B568826C44CFDBCFD40460B037AED3598FF3974AD9B6C7A46F4C059A17C3E9F2AD06D8B93C5A4DEA0D71C36A79F3E99" />
<add key="DECRYPTANDFORWARD_FIXED_MAGNEPRINTSTATUS" value="00000200" />

Finally, **MagneFlex** reads the template configuration data for an EMV_SALE:

8.2.3.1 Input

<add key="DECRYPTANDFORWARD_TEMPLATE_EMV_SALE"
value="DecryptAndForwardTemplates\Purplepay-EMV-SALE.xml" />

8.2.3.2 Output

<add key="DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_TRANSACTIONID"
value="/PurplePayResponse/TransactionID" />
<add key="DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_TRANSACTIONMSG1"
value="/PurplePayResponse/ResponseText" />
<add key="DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_TRANSACTIONMSG2"
value="/PurplePayResponse/ErrMsg" />
<add key="DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_TRANSACTIONSTATUS"
value="/PurplePayResponse/ActionCode" />
<add key="DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_ISSUERAUTHENTICATIONDATA"
value="/PurplePayResponse/ICC/IssuerAuthData" />
<add key="DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_ISSUERSCRIPTTEMPLATE1"
value="/PurplePayResponse/ICC/IssuerScript1" />
<add key="DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_ISSUERSCRIPTTEMPLATE2"
value="/PurplePayResponse/ICC/IssuerScript2" />

8.2.4 Step 3: MagneFlex interacts with DynaPro

MagneFlex calls the DynaPro and effects an EMV transaction. The DynaPro outputs encrypted and unencrypted data to **MagneFlex**

8.2.5 Step 4: MagneFlex builds the XML payload for the Decrypt and Forward service

First, **MagneFlex** determines which XML Templates are needed for this operation. As the ProcessEMVSRED operation was selected by the POS Application, and the TransactionType “SALE” was submitted, **MagneFlex** opens the XML Template file indicated by the key “DECRYPTANDFORWARD_TEMPLATE_EMV_SALE” which is “DecryptAndForwardTemplates\Purplepay-EMV-SALE.xml”.
MagneFlex then opens the template file and interrogates it. The following is an example of such a file for a fictitious Processor called “Purplepay”. Please note the added comments shown in **green**:

<?xml version="1.0"?>**Tag definitions and structure are defined by the Processor, in this case the Processor “Purplepay”. These can generally be found in the API published by the entity. Required tags are placed in the XML Template verbatim.**
<Purplepay Version="2.0">
<Application Version="2.0.0">MAGNEFLEX</Application>**Many of the field values, such as “MAGNEFLEX” in this case, are defined or assigned by the Processor.**
<Gateway>Purplepay</Gateway>
<IndustryInfo Type="RETAIL">
<Password>{MerchantPassword}</Password> **This indicates to MagneFlex to search for this field name in the input XML provided by the POS Application. As this is optional explicit data, MagneFlex will find the field at the node “TransactionInputDetails”. When it has done**

so, it replaces the variable here with the literal value of the field it finds in the input, which is “56^$3@3T” in this case. If MagneFlex could not find this variable in the input, it would assume the variable was implicit and pass this line of XML as shown verbatim to the Decrypt and Forward service. If the variable is defined there, it would be replaced using data obtained from the decrypted Terminal payload. If not, this tag would be completely removed before the final XML message is sent to the Processor.

```
<Address>{Address}</Address>
<TotalAmount>{Amount}</TotalAmount>This field replacement variable must be present in the XML Template as it maps to a required MagneFlex input.
<TransactionID>{TransactionID}</TransactionID>
<TransactionType>SALE</TransactionType>
<ICC>These are example of field replacement variables that are related to EMV that will be replaced by Decrypt and Forward from the Terminal payload.
        <AID>{9F06}</AID>
        <ARQC>{9F26}</ARQC>
        <ATC>{9F36}</ATC>
        <AuthorizedAmount>{9F02}</AuthorizedAmount>
        <CVMResult>{9F34}</CVMResult>
        <CardSeqNum>{5F34}</CardSeqNum>
        <CryptInfoData>{9F27}</CryptInfoData>
        <CurrencyCode>{5F2A}</CurrencyCode>
        <TVR>{95}</TVR>
        <TermAppVer>{9F09}</TermAppVer>
        <TermCountryCode>{9F1A}</TermCountryCode>
        <TermType>{9F35}</TermType>
        <TransSeqNum>{9F41}</TransSeqNum>
        <TransType>{9C}</TransType>
        <UnpredictableNumber>{9F37}</UnpredictableNumber>
</ICC>
</Purplepay>
```

8.2.6 Step 5: MagneFlex sends the constructed XML message to the Decrypt and Forward service

Once the replacements are complete, **MagneFlex** will send the completed XML message to Decrypt and Forward along with the Terminal payload and some of the configuration data. Decrypt and Forward will perform further field replacement work, and then send the final message to the Processor at the URL specified in the Configuration File.

8.2.7 Step 6: MagneFlex receives the response from Decrypt and Forward and parses key data

The response from the Processor, which the Decrypt and Forward service returns verbatim to **MagneFlex**, must be parsed for data the Terminal requires to complete the transaction. The following is an example of the response to the call made above in step 4. See inline comments in Green:

```
<PurplepayResponse Version="2.0">
        <TransactionID>0000000000000000134</TransactionID>When parsing the return data, MagneFlex looks for tags defined in the configuration file that might match the output. In this case, we see that it matches “<add key="DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_TRANSACTIONID" value="/PurplepayResponse/TransactionID" />”. This tells MagneFlex that the tag “TransactionID” that it found in the “PurplepayResponse” maps to the MagneFlex output
```

field “TransactionID”. If the tag is defined in the configuration file, but not present in the output, MagneFlex will throw an error. However, some fields defined in the configuration file are used internally by MagneFlex and not returned in the output.

```
        <ActionCode>000</ActionCode>
        <Approval>TEST26</Approval>
        <ResponseText>APPROVED</ResponseText>
        <UniqueID>QkVhViRcQnPbQlQcRcSiPmUb</UniqueID>
        <RRN>526018426541</RRN>
        <RawResponseCode>00</RawResponseCode>
        <ICC>
                <IssuerAuthData>472AD94F9FECD47D3030</IssuerAuthData>
```

The location of this data was defined in the Configuration File as “ <add key="DECRYPTANDFORWARD_PROCESSOR_FIELDMAP_ISSUERAUTHENTICATIONDATA" value="/PurplepayResponse/ICC/IssuerAuthData" />”.

```
                <IssuerScript1></IssuerScript1>
                <IssuerScript2>9F180430303031860E04DA9F580903C0DC6EF04E9C8A09860E
                04DA9F590908460C835744CE4E5C</IssuerScript2>
        </ICC>
</PurplepayResponse>
```

8.2.8 Step7: MagneFlex completes the transaction with the terminal (EMV only) and responds back to the POS Application

Since the transaction is EMV, the data is sent back to the Terminal and it completes the transaction with the card. Then the Terminal responds with a final message to **MagneFlex**, and **MagneFlex** forms the final output for the POS Application and sends it:

```
{
        "MagTranID": "ad3ed21a-59d8-4359-83b5-c42ccc953b05",
        "TransactionUTCTimestamp": "11/18/2015 9:29:00 PM",
        "CustomerTransactionID": "ad3ed21a-59d8-4359-83b5-c42ccc953b05",
        "TransactionOutput":
        {
                "TransactionID": "000000000000000332",
                "TransactionStatus": "000",
                "TransactionMessage": "APPROVED",
                "AuthCode": "TEST57",
                "AVSResult": null,
                "IssuerAuthenticationData": null,
                "IssuerScriptTemplate1": null,
                "IssuerScriptTemplate2": null,
                "IsTransactionApproved": true,
                "CVVResult": null,
                "AuthorizedAmount": null,
                "TransactionOutputDetails":
                [
                        {
                                "Key": "ProcessorResponse",
                                "Value": "<PurplePayResponse
                                Version=\"2.0\"><TransactionID>000000000000000332<\/Transa
                                ctionID><ActionCode>000<\/ActionCode><Approval>TEST57<\/App
```

roval><ResponseText>APPROVED<\/ResponseText><UniqueID><\/
UniqueID><\/PurplePayResponse>"
}
]
},
"EMVSREDOutput":
{
"AdditionalOutputData": null,
"CardID": null,
"IsReplay": false
}
}

9 Appendix A – EMV Terminal Configuration

Terminals to be used with **MagneFlex** must be EMV configured and certified to the Processor in online-only mode. **MagneFlex** does not support offline EMV card processing. Please see the required EMV configuration published by the certifying Processor, however, for use by **MagneFlex**, at a minimum the Terminal must be configured as follows[9]:

Tag	Tag Type	Value	Description
9F1B	Application	00 00 00 00	Floor limit
DFDF71	Application	00 xx 00 00 00	Terminal Action Code - Denial
DFDF72	Application	bit 8, byte 4 must be '1'	Terminal Action Code - Online
DFDF02	Reader	Must at least contain '57'	Authorization Request Tags
DFDF17	Reader	Cannot contain '57' or '99'	Batch Data Tags

[9] DynaPros ordered from MagTek with default tags will meet these requirements.

10 Appendix B – Decrypt and Forward Magstripe Field Replacement Variables

Variables

1. {DecryptedData}
2. {CCName}
3. {CCNum}
4. {CCTrack1}
5. {CCTrack1Length}
6. {CCTrack2}
7. {CCTrack2Length}
8. {CCardType}
9. {KSN}
10. {MMYYCCExpdt}
11. {MM_YYCCExpdt}
12. {DD}
13. {MM}
14. {YY}
15. {YYYY}

Functions

1. $Add(Oprand1_Numeric,Operand2_Numeric,ToStringFormat_Optional)
2. $DateTimeNow(Optional_format,Optional_0_Local_Or_1_Universal)
3. $DecimalToString (Oprand1_Numeric,ToStringFormat_Optional)
4. $IndexOf(string,searchString,startPosition,numberOfCharacterPosition)
5. $IndexIgnoreCaseOf(string,searchString,startPosition,numberOfCharacterPosition)
6. $Length(string)
7. $Multiply(Oprand1_Numeric,Operand2_Numeric,ToStringFormat_Optional)
8. $Replace (string,oldValue,newValue)
9. $Substring(string,startindex,length)
10. $Substring(string,startindex,length)
11. $TLVLength(string)
12. $Trim(string)

11 Appendix C – Use of MagneFlex with Browsers and Mixed Content

MagneFlex will accept an HTTP call from any source including browsers. However, many browsers have restrictions on the use of mixed HTTP/HTTPS content. If the POS Application is served from an HTTPS source, the browser may not allow it to call **MagneFlex** with HTTP. There are two options for resolving this issue:

1. If available, configure the browser to accept mixed mode content, and then call **MagneFlex** with HTTP as usual. However, even when so configured, the browser may still warn regarding the presence of mixed mode content.
2. Install a valid TLS certificate on the Terminal Host[10]:
    a. Associate an Internet domain to the Terminal Host.
    b. Change the BASEADDRESS key in the Configuration File to a URL with this domain.
    c. Generate a TLS server certificate for the Terminal Host that contains the domain.
    d. Import this certificate into the Personal Store under Local Computer.
    e. Run the command below at the command console as Administrator:
        i. netsh http add sslcert ipport=0.0.0.0:9000 certhash=650db634b84af07bf5bfee7d247270b34d776383 appid={00112233-4455-6677-8899-AABBCCDDEEFF}[11]
    f. Change the PROTOCOL key in the Configuration File to “HTTPS”.

[10] These instructions are for reference only. You may need to make adjustments for your environment.
[11] The fields shown need to be adjusted for your certificate where Certhash is the thumbprint of certificate.

12 Appendix D – MagneFlex Error Messages

If MagneFlex detects an error, it will return two key value pairs: faultcode and faultstring. The fault string provides additional information regarding the error. The faultcode values are as follows:

- 100: SUCCESS (no error detected, successful operation[12])
- 101: GENERIC (error)
- 102: DEVICE_OPERATION_FAILED
- 103: DEVICE_OPERATION_TIMEOUT
- 104: DEVICE_OPEN_FAILED
- 105: INVALID_AMOUNT
- 106: INVALID_CASHBACK
- 107: PROCESSOR_CALL_FAILED (Processor unreachable or returns unprocessable data)

[12] MagneFlex only detects processing level errors in itself and Magensa. A transaction fault at the Processor may still be deemed by MagneFlex to be a “SUCCESS”, if the Processor was able to return data to Magensa.

13 Appendix E – First Data EMV Receipt Requirements

- Transaction Type (Purchase)
- Application Preferred Name (Tag 9F12) is present on card in a supported characters set so it must be printed (VISA CREDIT or DEBIT)
- Card Entry Method indicates that card information was obtained from a contactless tap (Contactless)
- Currency indicator USD$ must be printed on the total line
- Transaction Amount (Tag 9F02) must be printed (75.01)
- Application PAN (Tag 5A) must be truncated and masked (**********8106)
- Authorization Mode identifies that the transaction was sent online to the issuer (Issuer)
- EMV AID must be printed on receipt and identified by the tag name:
- AID (A0000000031010)
- EMV Information should be printed in the order shown and identified by the tag names:
- TVR (0000008000)
- IAD (06010A03A40002)
- TSI (E800)
- ARC (00)
- Cardholder Verification Method was “Signature” so a Signature Line must be printed