

Merchant Warrior Fuse

Hosted Sales Page Integration Guide

Last Updated - 06/2011



Table of Contents

PREAMBLE	3
RELATED DOCUMENTS	3
WHO SHOULD READ THIS DOCUMENT?	3
CONTACT	3
GENERAL INFORMATION	4
INTRODUCTION	4
AVAILABLE METHODS	4
REQUEST FORMAT	4
RESPONSE FORMAT	4
PERFORMING A TRANSACTION	5
<i>Request Parameter Overview</i>	5
<i>Request Parameter Table</i>	6
<i>Request Parameter Table (cont)</i>	<i>Error! Bookmark not defined.</i>
<i>Response</i>	8
APPENDIX A – GENERATING THE VERIFICATION HASH	10
OVERVIEW.....	10
TRANSACTION TYPE HASH	10
URL HASH	10
302 REDIRECT & POST NOTIFICATION VERIFICATION HASH	10
CUSTOM FIELDS VERIFICATION HASH	10

Preamble

Merchant Warrior (MW) is an Australian based payment provider that offers a range of online payment solutions to Merchants worldwide. MW enables Merchants to connect to the major banks throughout Australia, and via our international partners, banks throughout Europe, American and Asia.

MW is a PCI DSS Certified Tier 1 gateway. This qualification was obtained after an independent audit carried out by Securus Global, a certified QSA and QPSAC.

Related Documents

Please also see the document entitled “Warrior Express – Integration Guide”, which should be used as an accompaniment. It contains information on transaction verification, as well as information relating to the asynchronous POST notification, appendixes, etc. As the MWF platform leverages the existing API, all of the information in that document relating to transaction processing is pertinent, and applies.

Who should read this document?

This document has been written to assist the programmers responsible for integrating MWF into a third party codebase. The assumption has been made that the reader will be familiar with basic industry standard terms such as HTTP, SSL, POST, XML, etc, and that they will have a basic understanding of the way that a hosted sales page generally works.

Contact

If anything in this document is unclear, incorrect or requires clarification, please don't hesitate to contact our support department, either by phone on **+617 3166 5489** or via email at infoservices@merchantwarrior.com. Our support department team is available 24/7.

General Information

The following sub-sections of this document will outline the various API methods present in the **Merchant Warrior Fuse (MWF)** solution.

Introduction

MWF, using the existing API provided by Merchant Warrior, allows a Merchant to create and maintain a single integration point, regardless of the banking partner (provider) they choose to use.

Before you are able to connect to the MWF platform, you must first obtain a Merchant ID, API Key and API Passphrase. These will be issued to you once you create a MWF account.

Available Methods

The MWF platform consists of a hosted page on Merchant Warrior's servers that can be utilized as a frontend to the MWE API, as well as a "Transparent Redirect" option that allows the Merchant to host the payment form, but not capture the payment details. With this in mind, there aren't any methods per-se, as the only function available via MWF is a regular purchase transaction.

This document outlines the Hosted Page method of connectivity.

Request Format

All requests are sent to the MWF solution using the HTTP POST method, and must be performed over the HTTPS protocol. A payment form is hosted on Merchant Warrior's servers, which the Merchant POSTS' a limited subset of data to, which initiates the transaction process.

For the Hosted Sales Page version of MWF, the requests must be sent to either <https://secure.merchantwarrior.com/> (live) or <https://securetest.merchantwarrior.com/> (testing).

Response Format

All MWF responses, regardless of the status, are returned via asynchronous POST as text/xml and contain, **at minimum**, the following XML elements:

```
<mwResponse>  
  <responseCode></responseCode>  
  <responseMessage></responseMessage>  
</mwResponse>
```

The element "mwResponse" will contain (at least) two child elements – responseCode and responseMessage. responseCode will be an signed integer (see Appendix D – Response Code Mapping) which allows the Merchant to quickly identify the status of the response. responseMessage will contain a textual string which offers more information on the specific response received.

Along with the POST notification, a 302 Redirect is returned to the Merchant via the client's browser once a transaction is completed. This request contains a query string, which contains the Transaction ID for querying, as well as summary data (transaction state, etc).

Performing a Transaction

In order to perform a transaction, a form must be posted to one of the request endpoints listed above, with a number of required (and some optional) fields. Assuming the request validates correctly, a form will be displayed to the customer.

Request Parameter Overview

The following parameters are accepted. Parameters that are **bold** are required.

```
method
merchantUUID
apiKey
transactionAmount
transactionCurrency
transactionProduct
customerName
customerCountry
customerState
customerCity
customerAddress
customerPostCode
customerPhone
customerEmail
hash
returnURL
notifyURL
urlHash
hashSalt
logoURL
custom1
custom2
custom3
```

For a more details about each field, please refer to the table on the next page.

Request Parameter Table

Required	Parameter
API Method	
Y	method
	<p>Example: processCard</p> <p>Notes: This is case sensitive. Currently, the only valid value for this parameter is 'processCard'</p>
Authentication Parameters	
Y	merchantUUID
	<p>Example: 123456789abcd</p> <p>Notes: The value of this parameter is assigned to you by Merchant Warrior.</p>
Y	apiKey
	<p>Example: 1a3b5c</p> <p>Notes: The value of this parameter is assigned to you by Merchant Warrior.</p>
General Transaction Parameters	
Y	transactionAmount
	<p>Example: 10.00</p> <p>Notes: The amount must be formatted to have two decimal places. Any amounts without two decimal places or amounts less than one cent will be rejected.</p>
Y	transactionCurrency
	<p>Example: AUD</p> <p>Notes: One of the following: AUD, CAD, EUR, GBP, JPY, NZD, SGD, USD. This is provider dependant. Please check with MW before attempting to process transactions in any currency other than AUD. This field is case insensitive.</p>
Y	transactionProduct
	<p>Example: Test Product</p> <p>Notes: A product (or sale) description. This field's primary purpose is to help the transaction be identifiable for reporting and accounting purposes.</p> <p>Valid Length: Up to 255 characters. Some banks limit this field to 40 characters.</p>
Customer Parameters	
N	customerName
	<p>Example: Mr. Example Person</p> <p>Notes: This field can only contain alphanumeric characters, as well as the full stop and hyphen character. This field MUST be the full name of the customer and MUST contain at least one space character.</p> <p>Valid Length: Between 3 and 255 characters.</p>
N	customerCountry
	<p>Example: AU</p> <p>Notes: Two letter ISO 3166-1 alpha-2 country code.</p> <p>Valid Length: Two characters.</p>
N	customerState
	<p>Example: Queensland</p> <p>Notes: Freeform field, keep consistent for your records and reporting.</p> <p>Valid Length: Up to 75 characters.</p>
N	customerCity
	<p>Example: Brisbane</p> <p>Notes: Freeform field, keep consistent for your records and reporting.</p> <p>Valid Length: Up to 75 characters.</p>

Request Parameter Table (cont)

Required	Parameter
N	customerAddress Example: 123 Test Street Notes: Freeform field. Valid Length: Up to 255 characters.
N	customerPostCode Example: 4000 Notes: This can also accommodate ZIP/Post codes for international transactions. Valid Length: Between 4 and 10 characters.
N	customerPhone Example: 0401234567 or 61731234567 Notes: Anything other than +, -, space and 0-9 will be stripped. Valid Length: Up to 25 characters.
N	customerEmail Example: person@example.com Notes: Must be valid if present. Sending this optional parameter is highly recommended. Valid Length: Up to 255 characters.
Redirect & Notification Parameters	
Y	returnURL Example: https://www.example.com/return.php Notes: The customer will be redirected to this URL upon completion of the transaction
Y	notifyURL Example: https://www.example.com/notify.php Notes: The asynchronous POST notifications will be sent to this URL.
Y	urlHash Example: Queensland Notes: The urlHash field is a combination of your API Passphrase, and specific parameters sent in the transaction. See Appendix A – Generating the Verification Hash (Type B – URL) for information on how to construct the hash correctly. Valid Length: 32 characters. Valid Length: Up to 75 characters.
Y	hashSalt Example: 3x4mpl3s4lt! Notes: Used to salt the return hash used in the 302 Redirect to redirectURL upon the completion of a transaction.
Y	logoURL Example: https://www.example.com/logo.png Notes: The URL to an image that will appear in the header of each page (max 70px high)
Custom Fields	
N	custom1 Example: Custom Field 1 Notes: Freeform field. Returned as <custom1> in the XML response. Valid Length: Up to 500 characters.
N	custom2 Example: Custom Field 2 Notes: Freeform field. Returned as <custom2> in the XML response. Valid Length: Up to 500 characters.
N	custom3 Example: Custom Field 3 Notes: Freeform field. Returned as <custom3> in the XML response. Valid Length: Up to 500 characters.

Response

Asynchronous POST

As stated in the “Response Format” section, all responses for the methods outlined in this document follow the same format, and are sent to the notifyURL via asynchronous HTTP POST. Take the following for example:

Successful Transaction:

```
<mwResponse>
  <responseCode>0</responseCode>
  <responseMessage>Transaction approved</responseMessage>
  <transactionID>1-918490ae-9a1c-11de-8649-000c29753ad4</transactionID>
  <authCode>1</authCode>
  <authMessage>Approved</authMessage>
  <authResponseCode>0</authResponseCode>
  <custom1>Custom Field 1</custom1>
  <custom2>Custom Field 2</custom2>
  <custom3>Custom Field 3</custom3>
  <customHash>e1af7c6a17758ef2907a50c96fa56d28</customHash>
  <hash>082949a8dfaccda185a135db425377b</hash>
</mwResponse>
```

If, however, the transaction did not pass MWF’s validation (due to an incorrect parameter, etc), a response similar to the following will be returned:

Failed Response:

```
<mwResponse>
  <responseCode>-1</responseCode>
  <responseMessage>MW - 008:Invalid merchantUUID / apiKey
  combination</responseMessage>
</mwResponse>
```

The <responseCode> and <responseMessage> fields will **always** be present in the transaction response. There are three possible types of <responseCode>’s that can be returned:

1. <responseCode> < 0. MWF validation error.
2. <responseCode> = 0. Transaction was successful
3. <responseCode> > 0. Transaction was declined by the provider.

If the transaction passed MWF’s validation, then generally the <auth*> fields will be present in the response. The <auth*> fields contain the upstream provider response data.

Please refer to “Warrior Express – Integration Guide” for more information and sample code.

302 Redirect

As an asynchronous notification is not always reliable, an immediate response is also sent in the form of a 302 redirect to the returnURL. For security reasons, not all of the transactional data is present in the result's Query string. The following table lists the possible return fields.

Field	Description
status	A textual representation of the transaction result.
reference	If status is 'error', this will not be present. Otherwise, this contains a transactionID that can be used to query the transaction via an API call with the queryCard method.
hash	A hash used to verify the status & transactionID, using hashSalt. Refer to Appendix A.
Code	If status is 'error', this will contain an MWE error code.
message	If status is 'error', this will contain a textual representation of the error code.
custom1	The value of the <custom1> parameter, URL Encoded. Will be blank if not provided.
custom2	The value of the <custom2> parameter, URL Encoded. Will be blank if not provided..
custom3	The value of the <custom3> parameter, URL Encoded. Will be blank if not provided..
customHash	A hash used to verify the custom* parameters. Refer to Appendix A.

Appendix A – Generating the Verification Hash

Overview

The verification hash is used to prove to MWF that the transaction request being sent has been generated by the Merchant, and not a malicious third party who has discovered the Merchant's merchantUUID and apiKey. Even if somebody captures the POST data that gets sent to MWF, they will not be able to create new (or refund old) transactions without knowing the API Passphrase – which you are able to change whenever you want via the Merchant Warrior Barracks.

To generate a verification hash, you need to concatenate specific fields that you're sending MWF in a specific order, convert them to lower case, and then MD5 them.

Transaction Type Hash

To generate a transaction type hash, concatenate the following fields:

```
| apiPassphrase + merchantUUID + transactionAmount + transactionCurrency
```

Once concatenated, convert everything to lowercase, and then md5 the string:

```
Step 1 (concatenate):  
passPhrase123456789abcd10.00AUD  
  
Step 2 (convert to lower):  
passphrase123456789abcd10.00aud  
  
Step 3 (md5):  
d941117d8774b12e218650542af6af56
```

URL Hash

To generate a URL hash, concatenate the following fields:

```
| apiPassphrase + merchantUUID + returnUrl + notifyURL
```

Once concatenated, convert everything to lowercase, and then md5 the string, as above.

302 Redirect & POST Notification Verification hash

To generate a verification hash used for the 302 Redirect and POST notification, concatenate the following fields:

```
| apiPassphrase + hashSalt + merchantUUID + status + transactionID
```

Once concatenated, convert everything to lowercase, and then md5 the string, as above.

Custom Fields Verification hash

To generate the custom fields hash, concatenate, convert to lowercase & md5 the following fields:

```
| apiPassphrase + custom1 + custom2 + custom3
```

Be sure to decode the custom* fields first – e.g. "Custom+Field+1" becomes "Custom Field 1".