

# ANZ EGATE™ VIRTUAL PAYMENT

---

CLIENT INTEGRATION GUIDE  
09.2017



**Prepared By:** ANZ

**Department:** Principal Consultant, eBusiness Solutions  
Date

**Written:** 1 November 2016

**Software Versions:** MIGS Payment Server 2.4, Virtual  
Payment Client 1.0

**Document Revision #:** 2.0

Before reading or using this Manual, please read our disclaimer. By accepting or reading this Manual you agree to be bound by the terms of the disclaimer.

#### **Disclaimer**

We may make improvements and/or changes to the products and services described in this Manual at any time.

To the fullest extent permitted by any applicable law:

- We give no warranties of any kind whatsoever in relation to this Manual including without limitation in respect of quality, correctness, reliability, currency, accuracy or freedom from error of this Manual or the products it describes. All terms, conditions, warranties, undertakings, inducements or representations whether expressed, implied, statutory or otherwise relating in any way to this Manual are expressly excluded.
- Without limiting the generality of the previous sentence neither us nor our affiliates, employees, directors, officers or third party agents will be liable to you for any direct or indirect loss or damage (including without limitation consequential punitive or special loss or damage) however arising in respect of this Manual or any failure or omission by us, even if we are advised of the likelihood of such damages occurring.
- If we have to accept any liability our total aggregate liability to you, including any liability of our affiliates, employees, directors, officers and third party agents

collectively, and regardless of whether such liability is based on breach of contract, tort, strict liability, breach of warranties, failure of essential purpose or otherwise, is limited to US\$500.00.

- While we have no reason to believe that the information contained in this Manual is inaccurate, we accept no responsibility for the accuracy, currency or completeness of the information in this Manual.
- We do not warrant or represent that we have checked any part of this Manual that is a copy of information we have received from a third party. We are merely passing that information on to you.

#### **License Agreement**

The software described in this Manual is supplied under a license agreement and may only be used in accordance with the terms of that agreement.

#### **Copyright**

MasterCard owns the intellectual property in this Manual exclusively. You acknowledge that you must not perform any act which infringes the copyright or any other intellectual property rights of MasterCard and cannot make any copies of this Manual unless in accordance with these terms and conditions.

Without our express written consent you must not:

- distribute any information contained in this Manual to the public media or quote or use such information in the public media; or
- allow access to the information in this Manual to any company, firm, partnership, association, individual, group of individuals or other legal entity other than your officers, directors and employees who require the information for purposes directly related to your business.

# Contents

<b>Preface</b>	<b>6</b>
What Is the Purpose Of This Guide	6
Who Should Read This Guide	6
Related Documents	6
<b>Terminology Used in this Guide</b>	<b>7</b>
Where to Get Help	7
<b>The Merchant Setup Process</b>	<b>8</b>
<b>Introduction to VPC</b>	<b>9</b>
What is MIGS?	9
What is the Virtual Payment Client?	9
<b>Steps to Integrating the VPC</b>	<b>10</b>
<b>MIGS Test Host Simulator</b>	<b>11</b>
<b>e-Payments</b>	<b>12</b>
Working with e-Payments	12
Introduction to Processing Payments	12
e-Commerce Transaction Modes	13
Payment Processing Modes	14
Purchase and Auth/Capture merchants	14
Subsequent or Financial Transactions	14
Integration Options	14
Server-Hosted transactions	15
Integrating Server-Hosted Payments	16
MIGS Processing of Server-Hosted Requests	16
Transaction response	16
What the Cardholder Sees	16
Merchant-Hosted Transactions	21
MOTO	21
Merchant Hosted Web Payment Pages	21
What the Cardholder Sees	21
<b>Best Practices</b>	<b>22</b>
Best Practices for Securing the Data	22
Use a unique Merchant Transaction Reference ID for each transaction	22
Check that the field values in the transaction response match those in the transaction request	22
Check for a replay of a transaction	22
Check the integrity of a transaction using Secure Hash	22
Other Best Practices	22
MerchTxnRef	22
What is Merchant Administration?	24
Cannot utilise the Advanced Merchant Administration (AMA) functionality?	24
Receipt number (RRN), MerchTxnRef, AuthorizeId and TransactionId	24
<b>Cardholder Authentication</b>	<b>25</b>
MasterCard SecureCode and Verified by Visa	25
Introduction	25
Authentication Process Flow	26
Server-Hosted Payment and Authentication Process Flow	26
<b>Payment Transactions for Server-Hosted Payments</b>	<b>27</b>
Transaction Request Fields	27
Required Transaction Request fields for a Server-Hosted Payment Request	27
Optional Transaction Request fields for a Server-Hosted Payment Request	28

Optional Merchant Defined Fields	28
Sending a Transaction Request for Server-Hosted Payments	28
Transaction Response Fields	29
Required Transaction Response fields for Server-Hosted Payment Response	29
Optional Transaction Response fields for Server-Hosted Payment Response	31
<b>Payment Transactions for Merchant-Hosted Payment</b>	<b>35</b>
Transaction Request Fields	35
Required Transaction Request fields for Merchant-Hosted Payment Request	35
Optional Transaction Request fields for Merchant-Hosted Payment Request	36
Sending a Transaction Request for Merchant-Hosted Payments	37
Transaction Response Fields	37
Required Transaction Response fields for Merchant- Hosted Payment Response	37
Optional Transaction Response fields for Merchant-Hosted Payment Response	39
<b>Advanced Functionality Fields</b>	<b>40</b>
Capture	40
Transaction Request Fields - Capture	40
Transaction Response Fields - Capture	41
Refund	43
Transaction Request Fields - Refund	43
Transaction Response Fields - Refund	44
QueryDR Transaction	46
Transaction Request Fields – Query DR	46
Transaction Response Details – Query DR	46
Bypass Card Selection Page on the Payment Server	47
Transaction Request Fields - Bypass Card Selection Page	47
Transaction Response Fields - Bypass Card Selection Page	47
<b>Troubleshooting and FAQs</b>	<b>48</b>
Troubleshooting	48
What happens if a Transaction Response fails to come back?	48
What to do if a Session Timeout occurs?	49
Does the Cardholders Internet browser need to support cookies?	49
How do I know if a transaction has been approved?	49
Frequently Asked Questions	49
Can the Payment Servers payment pages be modified for a Merchant?	49
Is a Shopping Cart required?	49
Does the Payment Server handle large peaks in transaction volumes?	49
How long will an authorisation be valid on a cardholder account?	49
What is the RRN and how do I use it?	49
RRN, MerchTxnRef, OrderInfo, AuthorizeId and TransactionId	49
<b>Appendix 3 – Test Environment</b>	<b>50</b>
Test Cards	50
Response Codes	50
Issuer Response Code Mapping	51

## List of Tables

Required Transaction Request fields for a Server - Hosted Payment Request	27
Optional Transaction Request fields for a Server - Hosted Payment Request	28
Required Transaction Response fields for Server - Hosted Payment Response	29
Optional Transaction Response fields for Server - Hosted Payment Response	31
Adding Secure Hash to a Transaction Request Example	33
Adding Secure Hash to a Transaction Response Example	34
Required Transaction Request fields for Merchant - Hosted Payment Request	35
Optional Transaction Request fields for Merchant - Hosted Payment Request	36
Sending a Transaction Request using the Post Method Example	37
Required Transaction Response fields for Merchant - Hosted Payment Response	37
Optional Transaction Response fields for Merchant - Hosted Payment Response	39
Transaction Request Fields - Capture	40
Transaction Response Fields - Capture	41
Transaction Request Fields - Refund	43
Transaction Response Fields - Refund	44
Transaction Request Fields - Query DR	46
Transaction Response Details - Query DR	46
Transaction Request Fields - Bypass Card Selection Page	47

## List of Figures

Merchant Simulator Infrastructure	11
How a transaction is processed	13
Information for Server - Hosted pages	15
What the Customer sees in a Server-Hosted transaction	17
The Shop & Buy Checkout Page	17
MIGS Payment Server's Payment Options Page	18
The MIGS Payment Server's Payment Details Page	18
The MIGS Payment Servers Payment Pending Page	19
The MIGS Payment Servers Redirection Page	19
The merchants Shop & Buy's Receipt Page	20
Information Flow in Merchant-Hosted transaction	21
Diagram Showing Information Flow	23
Server-Hosted Payment and Authentication Process Flow	26
What happens if a transaction response fails to come back	47

# Preface

## What Is the Purpose Of This Guide

This Merchant Developers Guide describes the Virtual Payment Client (VPC) API (Application Programming Interface) which allows you to payment enable your e-commerce application or on-line store. It seeks to guide you on how to use the functionality of the Virtual Payment Client API. The document describes Version 1.0 of the Virtual Payment Client API.

In addition, the guide outlines the business logic around payment processing on the MIGS Payment Server and how to use the VPC to perform payment processing and, if required, integrated administration functions.

## Who Should Read This Guide

The MIGS Virtual Payment Client API provides an easy to use, low integration effort solution for payment enabling web-sites, e-commerce applications and online stores. The solution uses standard web technology allowing merchants to integrate payment capabilities into their online store without installing or configuring any payments software making it suitable for most website hosting environments.

This guide is specifically aimed at business analysts and integrators who want to effectively integrate the VPC into merchant applications, and merchant bank personnel who will be involved with the support of the process.

## Related Documents

To complete the merchant offering, transactions processed through MIGS via the Virtual Payment Client Guide can be administered via the MIGS Merchant Administration portal. To understand what this portal offers, readers should reference the MIGS Merchant Administration Guide.

## Terminology Used in this Guide

<b>Access Code</b>	The access code is an identifier that is used to authenticate you as the merchant while you are using the Virtual Payment Client. The access code is generated and allocated to you by MIGS when you are established as a merchant on the server.
<b>Acquiring Bank</b>	The bank with which you have a merchant facility that allows you to accept online credit card payments.
<b>Merchant Administration</b>	Merchant Administration allows you to monitor and manage your electronic transactions through a series of easy to use, secure web pages.
<b>Payment Server</b>	The Payment Server is the MasterCard Internet Gateway Service (MIGS). MIGS facilitates the processing of secure payments in real-time over the Internet between your online store/ website and your bank. All communications between the cardholder, your online store, the Payment Server and the bank is encrypted, making the whole procedure not only simple and quick, but also secure.
<b>Purchase</b>	Purchase is a single transaction that debits the funds from a cardholder's credit card account and credits these funds to the merchants account. The transfer of funds occurs after end-of-day settlement between the card-issuing bank and the merchant's bank.
<b>Secure Hash Secret</b>	Secure Hash Secret plays a role in security as it is used to detect whether the transaction request and response has been tampered with. The Secure Hash Secret is generated automatically and assigned to you by MIGS when you are established as a merchant on the server. It is a unique value for each merchant and made up of alphanumeric characters. Only your MIGS know what the secure hash secret value is. Your secure hash secret is added to the transaction request details before an SHA256 algorithm is applied to generate a secure hash. The secure hash is then sent to MIGS with the transaction request details. Because MIGS is the only other entity apart from you that knows your secure hash secret, it recreates the same secure hash and matches it with the one that you sent. If they match MIGS continues processing the transaction. If it does not match it assumes that the transaction request has been tampered with, and will reject the request.  The Secure Hash Secret can be accessed using Merchant Administration. Please see the Merchant Administration Guide for more information.
<b>Virtual Payment Client (VPC)</b>	The Virtual Payment Client is the interface that provides a secure method of communication between your online store and the Payment Server, which facilitates the processing of payments.

## Where to Get Help

Should you require assistance with Virtual Payment Client Integration, please contact ANZ using the below details:

Phone: 1800 039 025 - 24/7

Email: ANZCommerceSupport@anz.com - response time is 1 business day.

# The Merchant Setup Process

The following table guides you through the basic steps to payment enable an online store, assuming that ANZ Merchant Services has approved your ANZ eGate Merchant Facility.

Table 1 What the merchant needs to do to process payments from an online store

Step 1
<b>Receive the Integration support material and documentation from your bank</b> The Virtual Payment Client integration support material issued to assist you during your integration and set up phase includes the following: a) Virtual Payment Integration Guide b) Example code
Step 2
<b>Obtain your Access Code and Secure Hash Secret from Merchant Administration</b> a) <b>Access Code</b> The access code uniquely authenticates a merchant and their Merchant ID on the Payment Server. b) <b>Secure Hash Secret</b> If you are using Server-Hosted Payments, the Secure Hash Secret is a key used as the initial piece of encryption data to create an SHA256 Secure Hash to ensure transaction data is not tampered with while in transit to the Payment Server. Your access code and secure hash secret can be found in Merchant Administration in the Setup menu option on the Configuration Details page. Please refer to your Merchant Administration User Guide for details on how to locate your Access Code and Secure Hash Secret.
Step 3
<b>Perform a basic payment using the supplied example code</b> You can perform a basic test payment using the example code provided. Successful completion of a payment using the example code validates that your system is set up correctly, and ensures basic functionality is available before implementing the integration with your online store. The example code covers common web server scripting languages. You will need to select the appropriate example for your specific web environment.
Step 4
<b>Design and implement the integration</b> You are now ready to payment enable your online store. This step requires a web developer familiar with both your online store and the web programming language used in integrating the Virtual Payment Client. This guide provides the reference information and best practise guidelines to assist you with this task. You may also refer to the example code for further assistance.
Step 5
<b>Test your integration</b> You need to test your integration by performing test payments. MIGS has a test bank facility to test all the different response codes that you are likely to encounter in a live environment. Performing test transactions validates that you have correctly integrated the Virtual Payment Client with your online store and that your application handles common response codes and error conditions. For more information, please refer to the Test Card information supplied at the end of this document.
Step 6
<b>Go Live</b> Once you are satisfied that your integration works correctly, change the configuration of your website from test mode to live production mode. This includes changing the ANZ eGate Merchant ID, Access Code & Secure Hash Secret in your integration. Refer to the your Merchant Administration User Guide for details on how to locate your Access Code and Secure Hash Secret. The production profile allows you to process live transactions with ANZ.
Step 7
<b>Conduct final Pre-Production testing.</b> It is recommended that you follow standard IT practices and complete final pre-production testing to validate that end-to-end functionality works correctly, including successful settlement of funds from ANZ.
Step 8
<b>Commence live processing of online payments.</b> You should now be ready to launch your payment enabled online store and start processing online payments from your customers.



## Introduction to VPC

### What is MIGS?

MIGS is the MasterCard Internet Gateway Service. MIGS is provided to banks to facilitate ePayments and general payments where the card is not presented to the merchant (called Card-Not-Present, or CNP).

MIGS processes all card types, not just MasterCard.

You have been provided this document because ANZ has implemented MIGS as their processing server for CNP transactions.

### What is the Virtual Payment Client?

The MIGS Virtual Payment Client is a mechanism for merchants to connect to the MIGS. It is termed 'Virtual' to contrast it to the Payment Client itself, but the VPC is a connection mechanism only – there is no supplied client software.

This manual outlines instructions for connecting your application to the MIGS Payment Server via this VPC connection mechanism.

# Steps to Integrating the VPC

Before you start integrating, you will need to determine if your online store supports the functions that you require. Your online store will determine the transaction types you can or cannot integrate.

## Step 1

**You need the following support material and information:**

- a) This Guide
- b) Example Code for your site (written in ASP, JSP, PHP and Perl)
- c) Access Code
- d) Secure Hash Secret (only used for Server-Hosted Payments).

## Step 2

**Determine which integration model you will be using?**

You need to know whether you are using:

- Server-Hosted Payments Integration Model, or
- Merchant-Hosted Payments Integration Model.

## Step 3

**Determine which Payment Model you will be using, Purchase or Authorisation/Capture**

- **Purchase** – requires a single transaction to transfer funds from the cardholder's account to your account.
- **Authorisation/Capture** – requires two transactions, the Authorisation, followed separately by a Capture.

## Step 4

**Determine if you will be using any Advanced functionality?**

The available advanced functionality includes:

- Verified-by-Visa and MasterCard SecureCode
- Capture
- Refund
- QueryDR

## Step 5

**Perform a basic transaction using the supplied example code**

You can perform a basic test payment transaction using the example code provided. Successful completion of a transaction using the example code validates that your system is set-up correctly and ensures basic functionality is available before you implement the integration with your online store. The standard example code covers common web server scripting languages. You need to select the appropriate example for your specific web environment.

## Step 6

**Determine how you are going to get the transaction request input fields and where to store the transaction response output fields in your online store.**

You need to consider:

- **Session Variables** - some online stores may require session variables to be collected and sent to the Virtual Payment Client in the transaction request. The session variables are returned in the transaction response allowing your online store to continue with the order process
- **Merchant Transaction Reference (vpc\_MerchTxnRef)** - You need to determine how you are going to produce a unique value for a transaction using the vpc\_MerchTxnRef field.

## Step 7

**Design and implement the integration**

You are now ready to payment enable your online store. This step requires a web developer familiar with both your online store and the web programming language used in your web environment.

This guide provides the reference information and best practice guidelines to assist you with this task. You may also refer to the example code for further assistance.

## Step 8

**Test your integration**

You need to test your integration by performing test transactions. MIGS The Payment Server has a test acquirer facility to test all the different response codes that you are likely to encounter in a live environment.

Performing test transactions allows you to test your integration, so that you won't encounter problems when processing real transactions. For more information, please refer to the Test Environment Section of this document.

## MIGS Test Host Simulator

The Host Simulator module provides a comprehensive transaction testing facility for the all VPC supported functions.

The merchant simply prefix's their merchant ID (which will be supplied to you by ANZ) with the word 'TEST' to initiate the routing of all transactions received from their application to the Test Host Simulator module. Full sets of test logs are written to the Merchant Administration Portal so that the Merchant can view and validate test results.

Exception handling is tested by simulating different responses from the MIGS Payment Server. Responses can be varied by using different values in the payment. For example, \$10.00 will return an 'Approved' by the MIGS Server, while \$10.51 will return 'Insufficient Funds'. Please refer to Appendix 1 for a full list of codes.

Once the transaction testing cycle has been completed successfully the code can be copied to the merchant's production system application. Once installed in the production system and all necessary ANZ processes have been completed, ANZ will set-up the merchant's Live Link. The merchant can access this link by simply removing the word 'TEST' from their merchant ID. The merchant's test profile will remain and can be accessed for further testing.

It is important to recognise that the merchant cannot perform live transactions unless specifically configured by ANZ to do so.

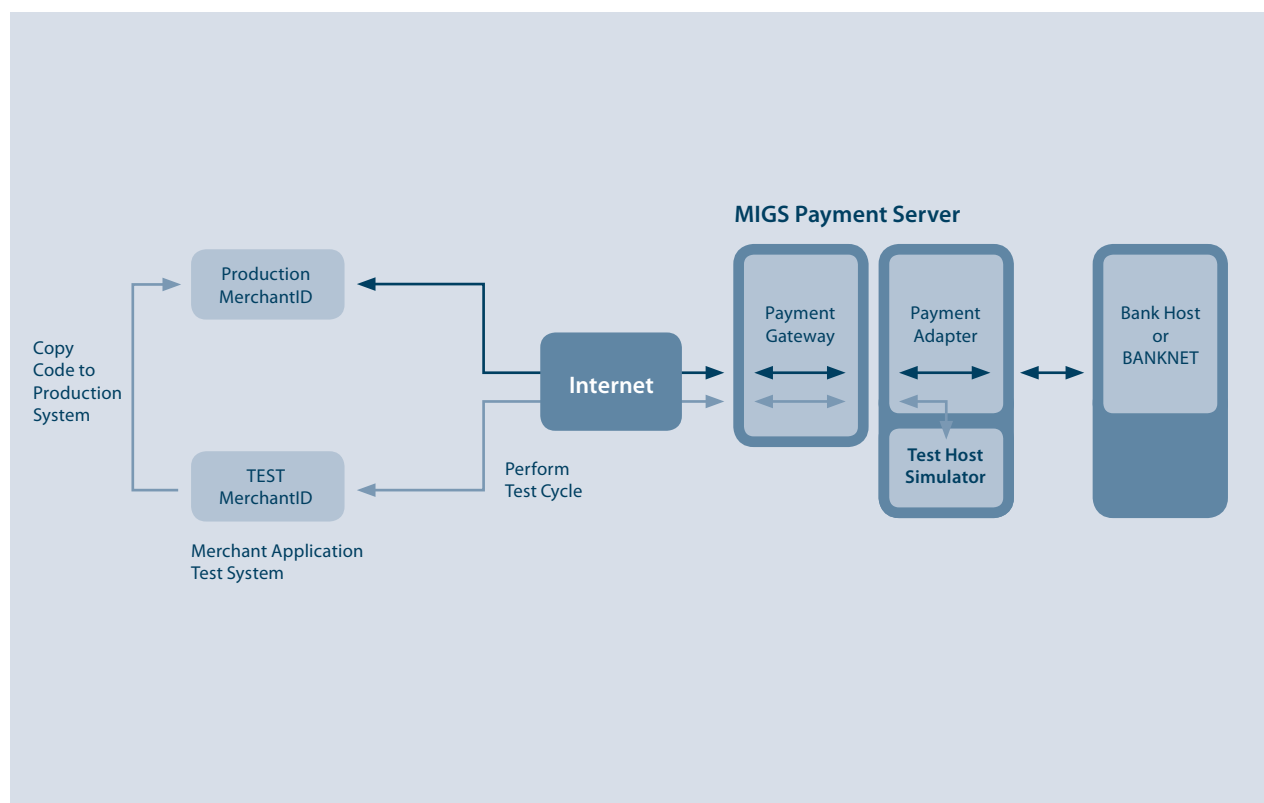


Figure 1: Merchant Simulator Infrastructure

# e-Payments

## Working with e-Payments

MIGS enables a merchant to perform secure transactions over the Internet. To do this they need to integrate the Host Application (Shop and Buy application) with the MIGS Payment Client, or the MIGS Virtual Payment Client. This guide deals only with the Virtual Payment Client. For information on the MIGS Payment Client please refer to the MIGS Payment Client integration Guide.

The Virtual Payment Client a series of commands available to the Host Application through an API (Application Program Interface). It interacts with the MIGS Payment Server, which processes secure transactions in real-time over the Internet.

The Virtual Payment Client is a remote interface to the MIGS Payment Server, which processes the secure transactions sent by the VPC.

## Introduction to Processing Payments

The typical payment process for Internet purchases is:

1. The cardholder purchases goods or services from a merchant using the Internet.
2. The merchant's online store sends a Virtual Payment Client transaction request via the MIGS Payment Server for authorization and processing.
3. MIGS directs the transaction to the cardholder's financial institution (issuing bank) for authorisation of the payment. The cardholder's account is debited and the funds are transferred to the merchant's account

As the Virtual Payment Client is designed primarily for processing payments from Internet sites, this guide presents information for enabling an online store. But it is also possible to payment enable other channels such as call centres or IVR systems using advanced functions of the Virtual Payment Client.

## e-Commerce Transaction Modes

### Processing an e-Commerce Transaction

During a transaction, the funds are transferred from the cardholders account to the merchants account in the following steps:

1. The cardholder purchases goods or services from a merchant via the internet, over the phone, etc.
2. The merchant sends the request for payment to MIGS, which processes the transaction on behalf of the merchant, by switching the transaction authorisation request to the card issuer (the customers bank) (3).
3. The card issuing institution adjusts the cardholder's credit limit for the funds and returns the result to MIGS. MIGS passes the result of the transaction on to the merchant.
4. Periodically (normally once a day), these records are transferred by MIGS to ANZ.
5. The merchant's bank settles the transaction with the issuing (Cardholder's) bank as part of normal credit card processing.
6. The issuing bank adds an entry to the cardholder's statement for subsequent payment by the cardholder.
7. The acquiring bank deposits the funds into the merchant's bank account.

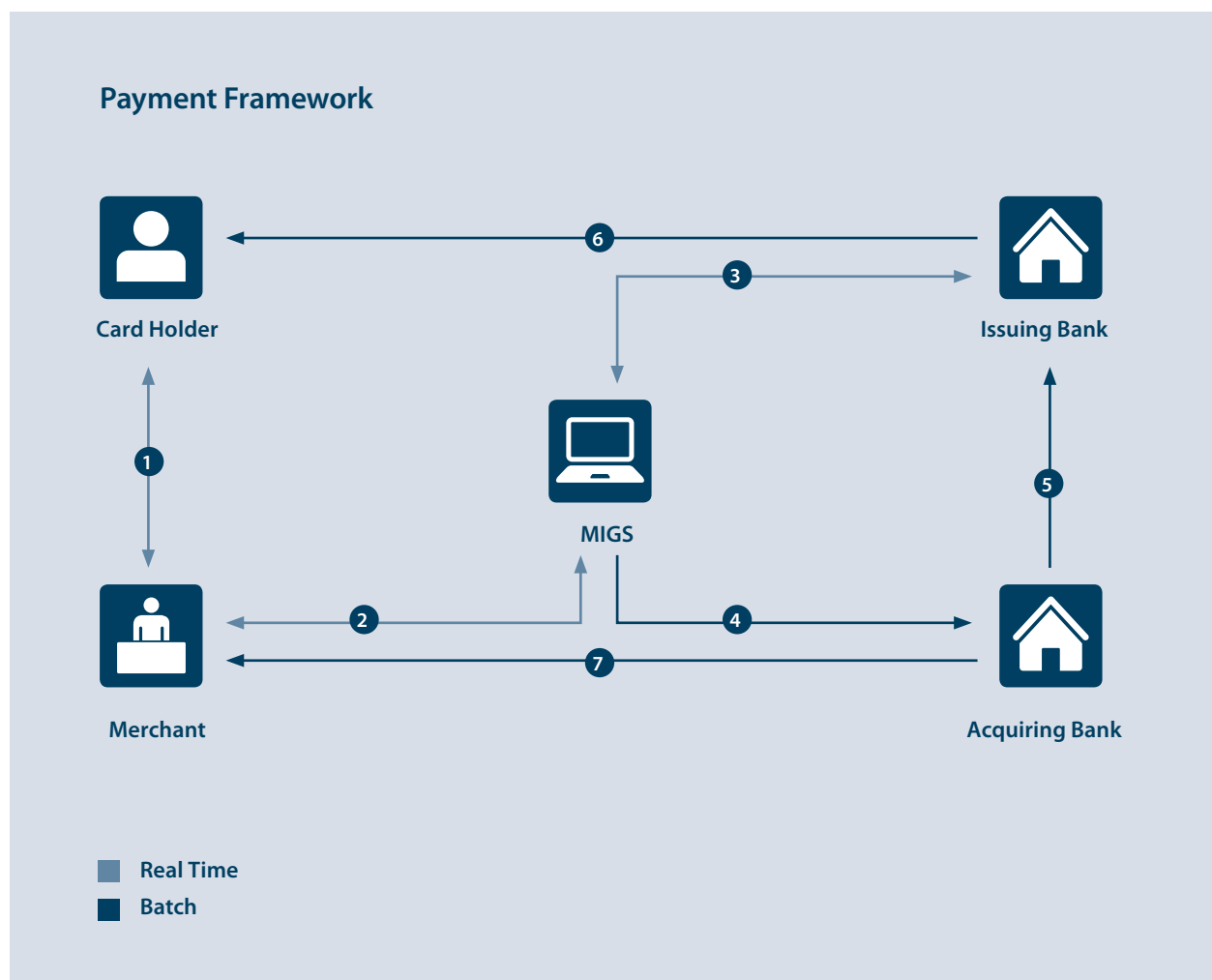


Figure 2: How a transaction is processed

# Payment Processing Modes

## Purchase and Auth/Capture merchants

### Purchase Merchant

Purchase mode transactions capture funds in a single transaction and the funds are immediately transferred into the merchant's account when the merchant's bank settles the transaction. Each instance of a purchase transaction will show up on the customer's card statement.

### Auth/Capture Merchant

Auth/Capture merchants perform at least two transactions to capture the funds from the customer's card and deposit them in the merchant's account.

- The authorisation (Auth) transaction verifies that the card details are correct and will also reserve the funds for that merchant.
- The capture transaction refers back to the initial authorisation transaction, and instructs the transfer of the funds from a customer's card into the merchant's account.

The merchant can perform more than one capture transaction, for example the merchant may not have the full ordered amount of goods in stock but ships what they do have. Later, when they ship the remaining goods the merchant can perform another capture transaction that refers back to the initial authorisation transaction which transfers the remaining funds to the merchant's account.

This auth reservation of funds will reserve the funds for a predetermined period of time, (such as 5 days), as determined by the issuing bank.

**Please check with your bank if you are unsure in which mode you are operating.**

## Subsequent or Financial Transactions

For every order there is normally one shopping transaction. Each shopping transaction may require a number of associated financial transactions, for example, **captures** and **refunds**. Financial transactions represent the flow of information between the customer, the merchant and the bank when purchasing goods and services.

Subsequent transactions can capture or refund transactions.

Refunds are where the merchant re-credits funds back to a customer's card. In this case both transactions are listed on the cardholder's statement.

The merchant can perform as many capture and refund transactions as they want providing the total amount captured does not exceed the original auth transaction. You must have captured the funds before you can perform a refund (i.e. the original transaction was a purchase or an authorised transaction that has been captured), otherwise an error will occur.

## Integration Options

There are two integration options for collecting credit card details on the MIGS Payment Server. However, your bank may not support both options. Please check with ANZ on which option(s) you are allowed to use.

The 2 options are:

1. **Merchant-Hosted** – (Internet or Mail Order/Telephone Order (MOTO)) Used for any merchant application, such as a merchant web shop & buy application or a call centre operation, where the merchant collects the card details.
2. **Server-Hosted** – (Internet only with possible authentication) Only possible from a web application, such as a merchant shop & buy application or e-mail, as the customer can only input their credit details direct to MIGS via a web page that is displayed from the MIGS Payment Server.

## Server-Hosted transactions

Server-Hosted transactions use the Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocol to provide secure transmission of sensitive data between a customer's web browser and the MIGS Payment Server.

1. A customer **1** and **6** decides to purchase goods and enter details into the merchants shop and buy application software at the checkout page.
2. The customer pays for the goods and the merchant software sends a Virtual Payment Client transaction request to the MIGS Payment Server, using an Access Code which is unique to each merchant. In addition, data integrity is protected by use of a Secure Hash Secret **2**.
3. The MIGS Payment Server receives the customer's card details **3** and displays a series of screens. The first screen displays the cards supported by the processor supports, for example MasterCard, Visa, and American Express. The customer chooses the card type they want to use for the transaction. The second screen accepts the details for the chosen card such as card number, card expiry, a card security number if required.

4. The MIGS Payment Server passes the details **4** to the card issuing institution. When the payment has been processed, the MIGS Payment Server temporarily displays the result of the transaction before displaying the final screen, which asks the customer to please wait while they are redirected back to the merchant's site (see page 22) and the MIGS Payment Server passes the result back to the merchant's site detailing the result of the transaction **5**. This information is then passed back to the user for their records **6**.

The MIGS Virtual Payment Client is a set of commands which all the sending and receiving of this data from the MIGS Payment Server via browser redirects.

In a Server-Hosted transaction, the customer's browser connection is completely severed from the merchant application, so any session variables that are required to identify the current session must be collected and sent to the MIGS Payment Server, where they are returned appended to the result message.

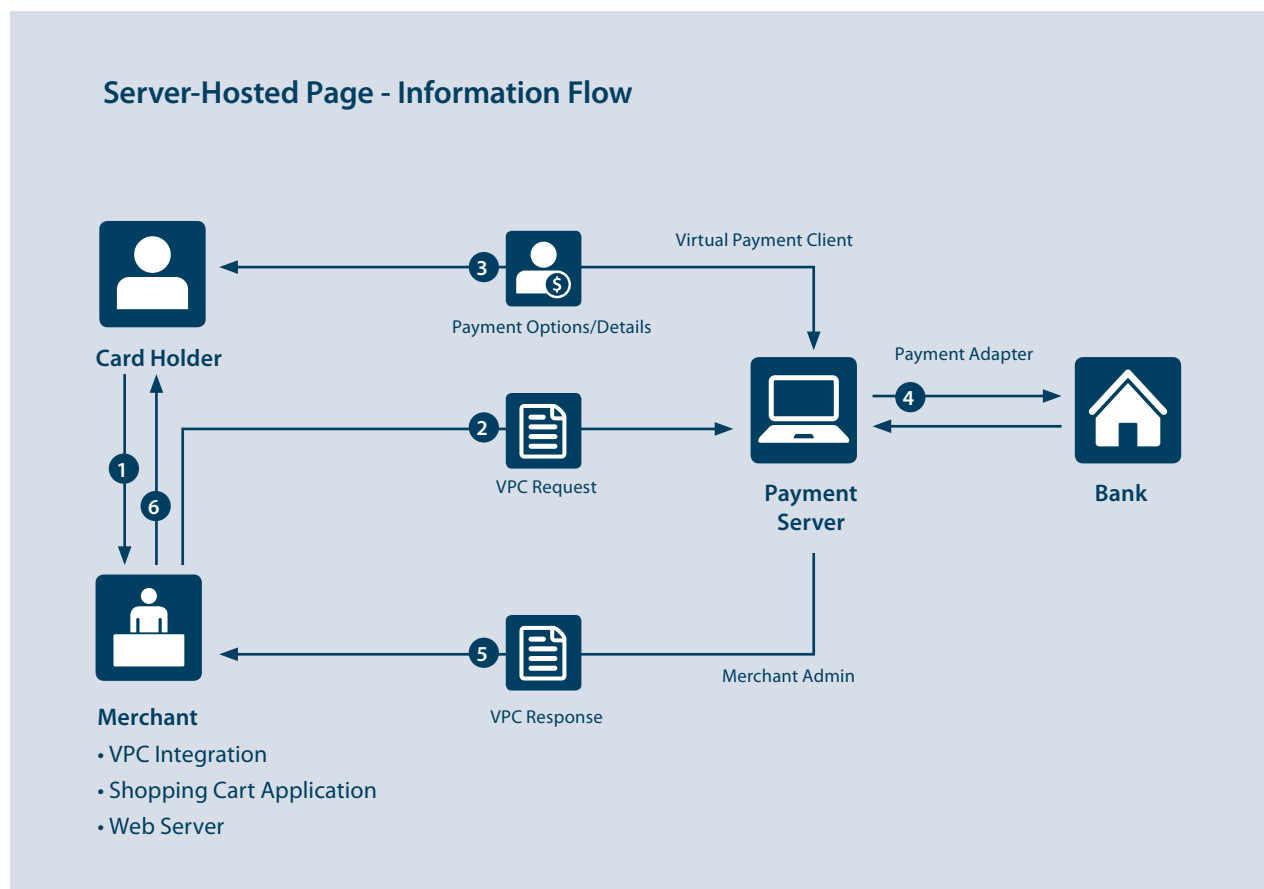


Figure 3: Information for Server-Hosted pages

## Integrating Server-Hosted Payments

To process a payment using Server-Hosted Payments your online store needs to be integrated with the Virtual Payment Client in order to send the transaction request and handle the transaction response.

To do this you need to do the following:

1. Collect the minimum required information for a transaction request. This will include your merchant ID, your access code, the order amount, a transaction reference number and an optional order information field. You may require additional information when using optional features.
2. Formulate a transaction request.
3. Add any session variables required by the online store to resume the order process with the cardholder after the transaction request has been processed.
4. Calculate the secure hash and append it along with the Secure Hash type to the transaction request.
5. Redirect the cardholders Internet browser using the transaction request you just created.

At this point the cardholder session with your online store is interrupted while the cardholder submits their card details directly to the MIGS Payment Server.

### MIGS Processing of Server-Hosted Requests

When a transaction request arrives at the MIGS Payment Server by redirecting the cardholder's Internet browser, it checks to make sure the Merchant Access Code is correct for the merchant ID and also checks if the secure hash is present. If both are correct, the Payment Server:

- Displays the card selection page (normally branded by ANZ) for the cardholder to choose their card type.
- Accepts the cardholders card details for the selected card type.
- Processes the transaction request and notifies the merchant's bank of the status of the transaction so the funds can be settled into the merchant's account.
- Sends back a transaction response to your website page nominated in the transaction request indicating whether the transaction was successful or declined.
- The MIGS Payment Server can also return error messages back, if for example there is a communication error in the banking network and the transaction cannot proceed.
- If either the Merchant Access Code, Secure Hash Type or the Secure Hash are incorrect, the MIGS Payment Server returns a transaction response back to your website page nominated in the transaction request with a response code and a message detailing the error.

### Transaction response

The transaction response is returned to your website using an Internet browser redirect as specified in the `vpc_ReturnURL` field. The transaction response will always have a secure hash for the online store to check data integrity.

The online store needs to process the transaction response by checking the secure hash received from the Payment Server is correct:

- Check the value of the `vpc_TxnResponseCode`.
- If this is equal to '7' then the MIGS Payment Server has detected an error related to the message and you need to handle this condition.
- If this is equal to '0' then the transaction was completed successfully and you can display a receipt to the cardholder.
- If it is equal to any other value the transaction has been declined and this must be declared to the cardholder. The action taken in this event (suggesting a retry with another card, suggesting another payment method etc.) can be determined by the merchant.

If the value of the secure hash and Secure Hash Type received is not equal to the hash value calculated from the data, the data may have been tampered with in the redirection process and you should check the transaction response data against the original transaction request. For example, you should check that the transaction amount is the same as what you sent originally.

The online store response page needs to be able to handle:

- Successful transactions
- Declined transactions
- Error Conditions – if `vpc_TxnResponseCode` equals '7' an error has occurred.

All three of these conditions are valid responses that occur back from the MIGS Payment Server. The next section provides you with an overview of securing your payments.

### What the Cardholder Sees

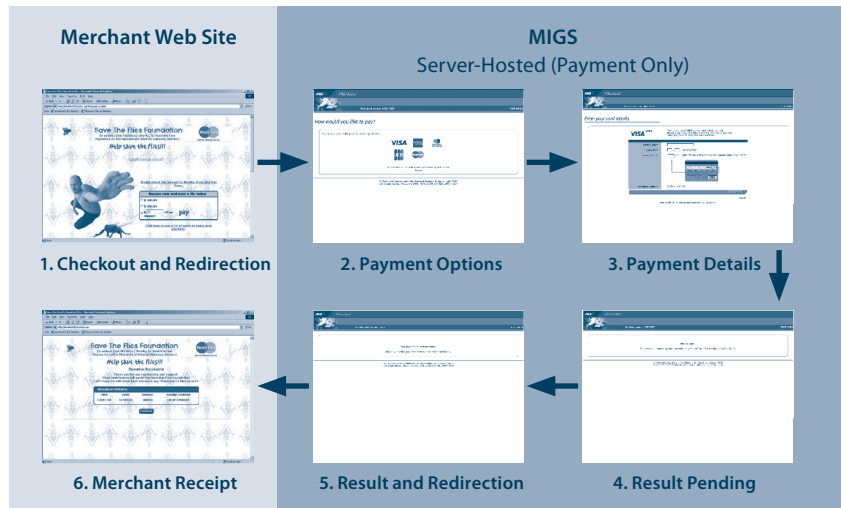
In a Server-Hosted transaction (without authentication, covered later in this guide) the cardholder is presented with six pages:

1. The Merchant's web-site checkout page
2. The MIGS Payment Server's Payment Options page
3. The MIGS Payment Server's Payment Details page
4. The MIGS Payment Server's Payment Pending page
5. The MIGS Payment Server's Redirection page
6. The Merchant's web-site receipt page.



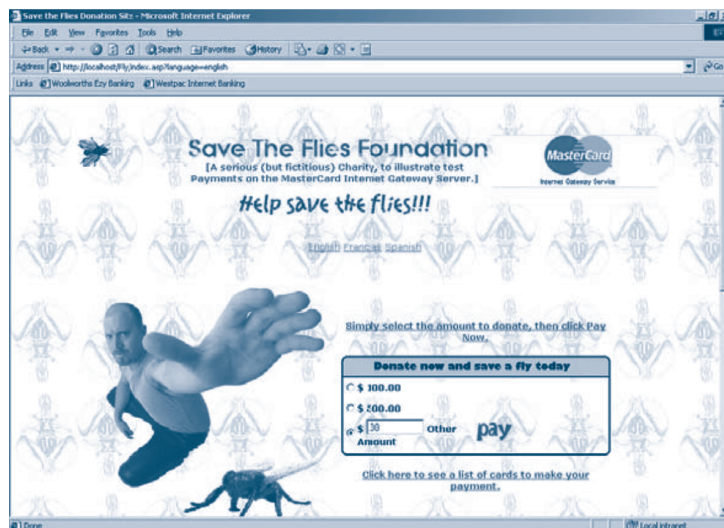
Examples of these pages are shown below

### What the Customer sees in a Server-Hosted transaction



### The Shop & Buy Checkout Page

The checkout page displays the line items that the customer wants to purchase and the total amount to pay, including any delivery charges and taxes. The customer accepts the amount and proceeds to the MIGS payment pages to enter their card details.



## MIGS Payment Server's Payment Options Page

The payment options page presents the customer with the card types the merchant accepts. The customer clicks a card type and proceeds to the Payment Details page.

The screenshot shows the ANZ eGate interface. At the top, there's a header with the ANZ logo and "ANZ eGate™". Below the header, a dark blue bar contains "Merchant name: ANZ TEST" on the left and "TEST MODE" on the right. The main content area has the heading "How would you like to pay?". Below this, a box says "Pay securely by clicking on the card logo below:". Inside this box are logos for VISA, American Express, Diners Club International, JCB, and MasterCard. Below the logos, it says "Your details will be sent to and processed by ANZ eGate." with a "Cancel" link. At the bottom of the page, there is a copyright notice: "© Copyright Australia and New Zealand Banking Group Limited (ANZ) 100 Queen Street, Melbourne 3000, ABN 11 005 557 522, 1996 - 2004".

## The MIGS Payment Server's Payment Details Page

On the Payment Details page, the customer enters their card details, including the card number and expiry date, and clicks the pay button. MIGS then processes the payment.

The screenshot shows the ANZ eGate interface for the "Enter your card details" page. The header is the same as the previous page. The main heading is "Enter your card details". Below this, a box contains the VISA logo and the text: "You have chosen VISA as your method of payment. Please enter your card details into the form below and click 'enter payment' to complete your purchase." The form has the following fields: "Card Number:" with a text input; "Expiry Date:" with two dropdown menus for month and year; "Security Code:" with a text input and a note "The 3 digits after the card number on the signature panel of your card." pointing to a VISA card image. Below the form, it says "Purchase Amount: AUD \$100.00". At the bottom of the form area, there is an "enter payment" button and a "Cancel" link. Below the form area, it says "Your details will be sent to and processed by ANZ eGate."

## The MIGS Payment Servers Payment Pending Page

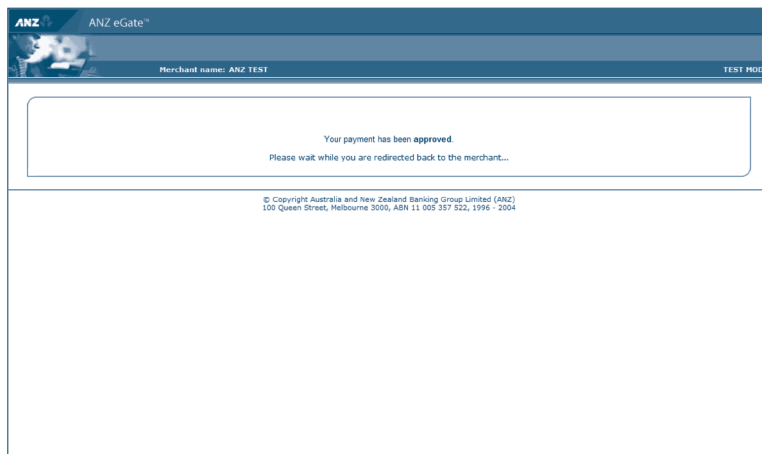
As the payment processor is processing the payment, a payment pending page can be displayed to the customer.



The screenshot shows the ANZ eGate interface. At the top, there is a header with the ANZ logo and 'ANZ eGate™'. Below the header, a dark blue bar contains 'Merchant name: ANZ TEST' on the left and 'TEST MODE' on the right. The main content area is a light blue box with a rounded border. Inside this box, the text reads: 'Please wait...' followed by 'The server is processing your payment using MasterCard for the value of AUD \$100.00.' Below this box, there is a small copyright notice: '© Copyright Australia and New Zealand Banking Group Limited (ANZ) 100 Queen Street, Melbourne 3000, ABN 11 005 357 522, 1996 - 2004'.

## The MIGS Payment Servers Redirection Page

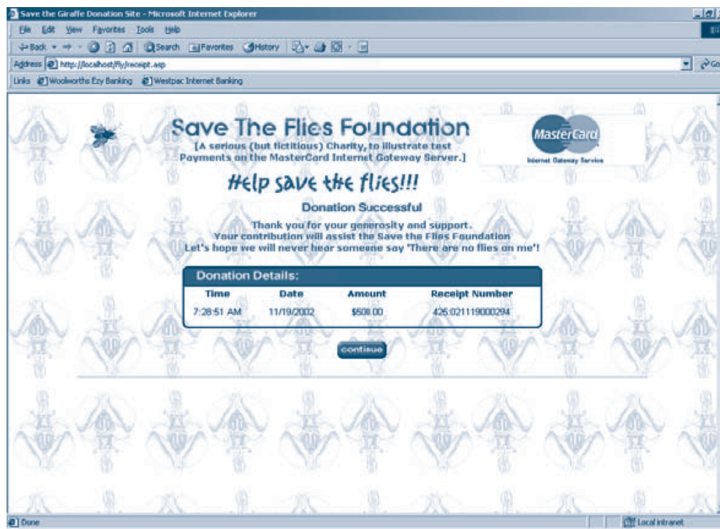
The redirection page is displayed in the customer's browser and the Digital Receipt is passed to the merchant's shop and buy application.



The screenshot shows the ANZ eGate interface. At the top, there is a header with the ANZ logo and 'ANZ eGate™'. Below the header, a dark blue bar contains 'Merchant name: ANZ TEST' on the left and 'TEST MODE' on the right. The main content area is a light blue box with a rounded border. Inside this box, the text reads: 'Your payment has been approved' followed by 'Please wait while you are redirected back to the merchant...'. Below this box, there is a small copyright notice: '© Copyright Australia and New Zealand Banking Group Limited (ANZ) 100 Queen Street, Melbourne 3000, ABN 11 005 357 522, 1996 - 2004'.

## The merchants Shop & Buy's Receipt Page

The shop and buy receives the Digital Order and creates a Digital Receipt as a receipt page is displayed to the customer.



## Merchant-Hosted Transactions

There are 2 types of merchant hosted transactions.

### MOTO

MOTO (Mail Order/Telephone Order) transactions are purchase/auth transactions orders where the customer provides their card details to a merchant, via mail order or by telephone (including Interactive Voice Response (IVR) systems).

### Merchant Hosted Web Payment Pages

The merchant has the option of providing their web-site own payment pages for collecting the card details. The customer provides their payment details (card type, card number and expiry date) directly to the merchant.

Merchant-Hosted transactions carry a higher risk than Server-Hosted transactions, as the customer's card details are captured and stored by the merchant.

1. A customer **1**, **5** purchases goods or services.
2. The merchant collects the card details using the Internet, IVR, mail order or telephone order and submits the details to be processed via the Virtual Payment Client **2**.
3. The message is sent over the Internet to the MIGS Payment Server **3**.

The message includes the purchase amount, card details (submitted to the merchant), and a merchant-specified transaction reference.

4. The issuing bank processes the information and passes the result back to the MIGS Payment Server. This result, which includes the transaction results and payment reference details, is sent from the MIGS Payment

Server back to the merchant's site page specified in the outgoing request **4** where it is processed. A receipt is also passed back to the customer for their records **5**.

The basic inputs used for a Merchant-Hosted transaction are:

- **CardNumber** – The card number or the customer.
- **CardExpiry** – The expiry date of the card.
- **MerchantId** – The merchant identifier allocated by their bank.
- **MerchTxnRef** – Identifies this particular transaction on the MIGS Payment Server. This should be a unique value for each transaction attempt, which makes it easy for the merchant to track transactions.
- **Amount** – Contains the value of this transaction. It is an integer that expresses the value in the lowest currency denomination, for example, cents, pence and yen.

In a Merchant-Hosted transaction, session variables are not sent to the MIGS Payment Server because the merchant's session is always maintained.

### What the Cardholder Sees

In a Merchant-Hosted transaction the cardholder is presented with two pages:

1. The merchants shop and buy checkout page.
2. The merchants shop and buy receipt page.

The MIGS Payment Server does not display any pages in a Merchant-Hosted style transaction, as all pages are displayed by the merchant's application.

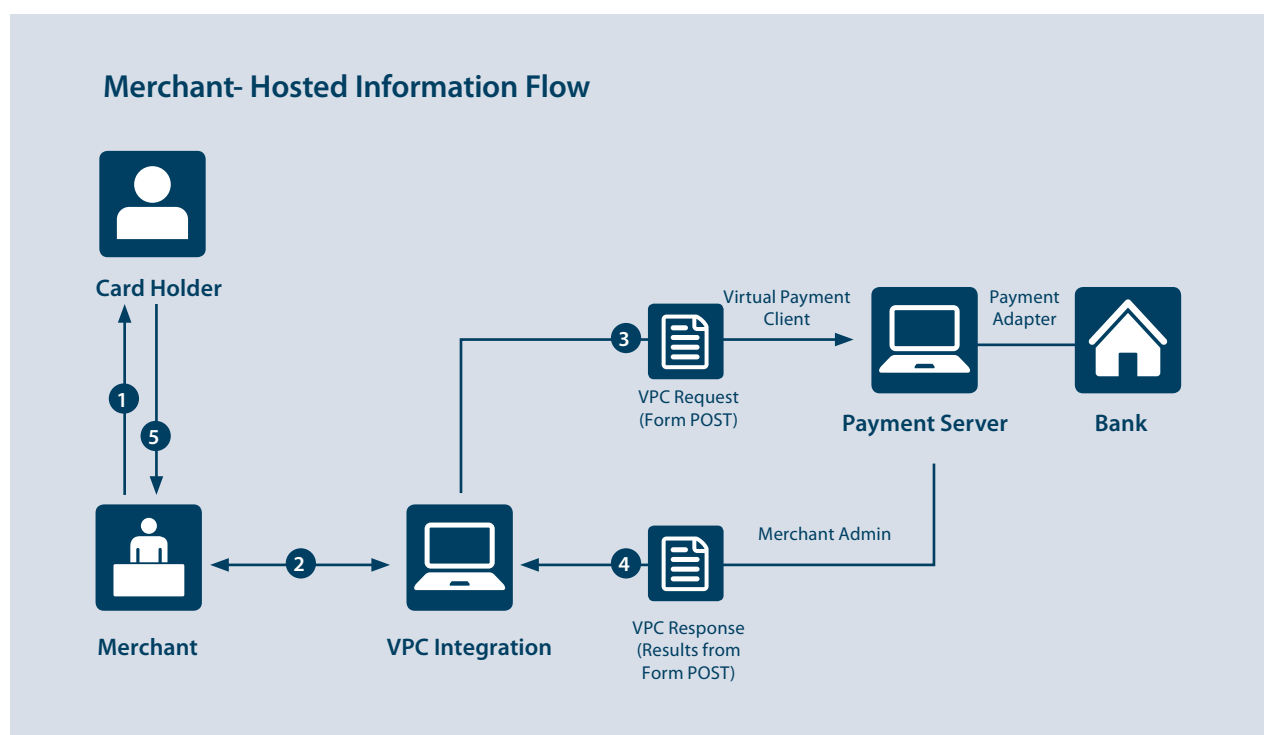


Figure 4: Information Flow in Merchant-Hosted transaction

# Best Practices

## Best Practices for Securing the Data

This section describes the security features available for the Virtual Payment Client. It is recommended that you understand this section before you start integrating

All websites collecting sensitive or confidential information need to protect the data passed between the browser and the MIGS Payment Server. The Payment Server is responsible for securing the cardholder details when you implement the Server-Hosted Payments Integration Model. It uses Secure Sockets Layer (SSL) or Transport Layer Security (TLS), which encrypts sensitive financial data between a cardholder and the Payment Server.

SSL (Secure Sockets Layer) and Transport Security Layer (TSL) is a security technology that is used to secure server to Internet browser transactions. This includes the securing of any information passed by an Internet browser (such as a cardholder's credit card number) to a web server (such as your online store or MIGS). SSL protects data submitted over the Internet from being intercepted and viewed by unintended recipients.

There are three security issues that apply to a payment:

- **Confidentiality** – To protect important information such as credit card numbers.
- **Identification/Authentication** – To ensure that transaction requests are going to the MIGS Payment Server and that the transaction response came from the MIGS Payment Server.
- **Integrity** – When you send and receive messages, you need to be sure that they are not being altered. In Server-Hosted Payments, the transaction request and response is sent by redirecting the cardholder's Internet browser providing an opportunity for the cardholder to modify details if best practice suggestions in this section are not followed.

### Common Best Practices to ensure Transaction Integrity

The following Best Practices are guidelines only. It is recommended that you consult with security experts with experience in your web environment to ensure that your security is suitable for your needs.

#### Use a unique Merchant Transaction Reference ID for each transaction

Each transaction should be assigned a unique transaction reference ID. Most online stores and web programming environments will generate a unique session ID for each cardholder, which can be used as the unique transaction reference ID. You can alternatively create a unique reference ID by combining a unique order number with a payments attempt counter. You may also consider appending a timestamp to the transaction reference ID to help ensure that each one is unique. Before sending a cardholder to the Payment Server, you should store this unique transaction reference ID with the order details in your online store database. The merchant transaction reference ID is returned in the transaction response and allows you to match the response against the order. The unique transaction reference ID is required to provide transactional integrity, protect against replay attacks and aid in reconciliation.

#### Check that the field values in the transaction response match those in the transaction request

You should ensure that important fields such as the amount in the transaction response and the merchant transaction reference ID match up with the values in the original order.

#### Check for a replay of a transaction

You should check each transaction response to ensure that your unique Merchant Transaction Reference ID matches the order, and that it does not correspond with any previous order that has already been processed.

#### Check the integrity of a transaction using Secure Hash

The Secure Hash is used to prevent the cardholder from modifying a transaction request and response when passing it through the cardholder's browser. Using the Secure Hash ensures a high level of trust in the transaction result. The benefit of using Secure Hash is that the integrity of each response can be checked without having to create a new SSL or TSL connection to the MIGS Payment Server for each transaction. If you have network restrictions that do not allow an outbound socket from your site, then you should use this process. The Secure Hash Secret must be kept secret to provide security and should be changed periodically for this method to be effective. The Secure Hash method is only applicable when using the Server-Hosted Payments integration model.

## Other Best Practices

### MerchTxnRef

#### Unique value

MerchTxnRef is normally used for querying an exact transaction on the MIGS Payment Server. In a case where the merchant requires to know the specific result of a transaction, for example, when a Digital Receipt is not received by the merchant, then the MerchTxnRef is used to locate the details.

Although MIGS allows any reference to be entered with a shopping transaction, it is advised that some unique identifier is used by the merchant to allow an easy cross-reference with the merchant's host system. A example of a MerchTxnRef would be the merchant's unique order number assigned to each sale. This allows the merchant to look up the transaction on MIGS with the same reference used to lookup the transaction on their own host system. To guarantee uniqueness, different payment operations on the same sale need also to be identified, as stated below.

#### Identifying Payment Attempts

If a transaction for a sale is declined, and a subsequent attempt is made to process a payment for this sale, the merchant should modify the MerchTxnRef for each subsequent attempt, by appending extra characters for each attempt. For example **MerchTxnRef** = '1234/1' on first attempt, '1234/2' on second attempt, and '1234/3' on third attempt, etc. This is the preferred way of implementing a unique **MerchTxnRef**. Because under a fault condition, such as if the Digital Receipt does not arrive back at the merchant's site, you may need to check if the transaction was carried out successfully.

Automated lookups and financial operations cannot be performed if the merchant has not given each transaction attempt a unique **MerchTxnRef** number, as there will be multiple results showing the same **MerchTxnRef**.

#### Sending Session Variables to the MIGS Payment Server

In a Server-Hosted transaction, the customer browser's connection is completely severed from the merchant application. Some merchant applications use session variables to keep track of where the shop and buy application is up to and to prevent unauthorised entry without the customer signing in. This stops hackers from spoofing transactions.

Session variables that are required to identify the current session so the MIGS Payment Server can return to the merchant's program from where it left must be collected and sent to the MIGS Payment Server. The session variables are not used by the MIGS Payment Server, but are returned appended to the response. There can be as many session variables as required using any name the merchant shop and buy application needs, providing they legally conform to HTTP/HTTPS protocols. To make them conform to the standard, URL, you need to encode all session variables before sending them.

To send them to the MIGS Payment Server the merchant must append them to the merchant ReturnURL.

At the MIGS Payment Server, the merchant session variables are recovered and temporarily stored in the MIGS Payment Server with the other transaction variables. They are sent back to the merchant appended to the response at the completion of the transaction.

The merchant shop and buy application recovers the session variables from the response, and uses them to restore the merchant session. The session continues as though it had never been broken.

#### Receipt Failure

The two ways of dealing with a digital receipt that fails to come back are:

- Flag the transaction as having an error that the merchant needs to manually check using Merchant Administration on the Payment Server.
- Utilise Advanced Merchant Administration (AMA) commands to search the MIGS Payment Server database for the transaction by using the **QueryDR** command if **MerchTxnRef** is unknown. The **MerchTxnRef** is used as the transaction identifier when searching using **QueryDR**.

Because the Digital receipt has failed to come back, there is no transaction number available from the Payment Server to identify the transaction in question, and this is why you use the **MerchTxnRef**. It is important to have a unique **MerchTxnRef** for every transaction otherwise the query could return multiple results. Only the most recent transaction is returned in the **QueryDR** command if there are multiple results, but this may not be the transaction you are looking for.

When you find the required **MerchTxnRef** in the **QueryDR**, check if it is successful by the **QSIResponseCode** field (equal to '0'). If the **QSIResponseCode** is zero, then the transaction is successful and you just need to extract the relevant data details from the **QueryDR** results for your records. If the **QSIResponseCode** is not 0, you need to determine the next course of action based on what you would do if the **QSIResponseCode** were not 0 in a normal digital receipt coming back from the Payment Server.

If you query the Payment Server for the **MerchTxnRef** using the **QueryDR** call and you do not receive any results, then it is safe to repeat the transaction. It is safe to use the same **MerchTxnRef**, as the existing one does not show up in the Payment Server's database.

If the **QueryDR** is flagged as having multiple results (returns 'Y' in the **MultipleResults** field), the **MerchTxnRef** is not unique.

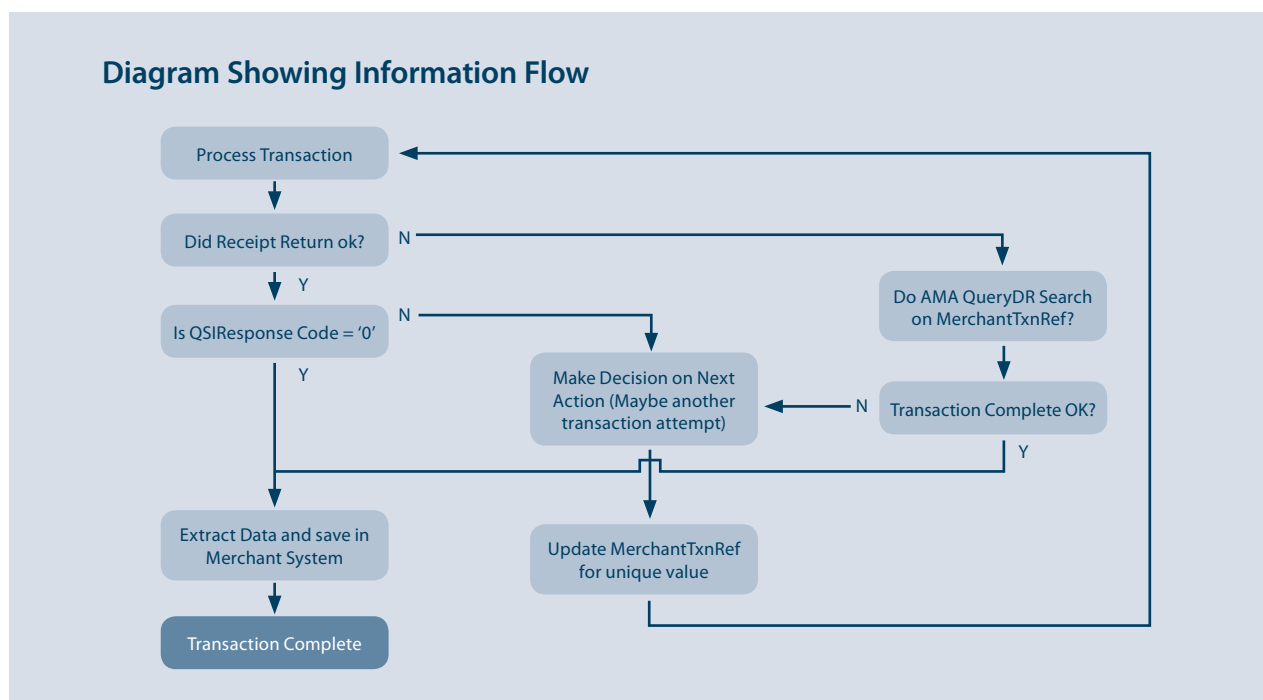


Figure 5: Diagram Showing Information Flow

### What is Merchant Administration?

Merchant Administration is the Internet based portal, which allows merchants to monitor and manage their on-line processing and administration of payments through a series of easy-to-use pages.

To use Merchant Administration, you need to have access to the Internet through a browser (such as Internet Explorer). You also need the MIGS URL (or web site address).

The merchant can use one of two methods to manage their transactions:

- **Merchant Administration** - using a browser interface to interactively perform historical searches, captures, refunds and to perform setup activities. For more details, please refer to the Merchant Administration User Guide.
- **Advanced Merchant Administration** - using the Payment Client to directly access the MIGS Payment Gateway to perform all transaction-related actions (for example, captures, refunds and voids) integrated with merchants' software interfaces.

### Cannot utilise the Advanced Merchant Administration (AMA) functionality?

The following reasons may cause AMA functionality to not work:

- A separate operator needs to be created for AMA API calls
- Your merchant account is required to have the privileges to execute AMA functions, please check with the helpdesk that this is enabled for your merchant account.
- Advanced admin methods privilege has to be enabled in operator set-up in Merchant Administration. An AMA operator cannot connect to Merchant Administration unless the AMA privilege is removed.
- Check that you are not using incorrect merchant ID, operator ID or password details.

### Receipt number (RRN), MerchTxnRef, AuthorizeId and TransactionId

- The **Receipt Number (RRN)** is normally a unique number for a particular Merchant ID generated by the bank. This is the value that is passed back to the customer for their records. You cannot search for this field in Merchant Administration or using AMA, but it is displayed in Merchant Administration on the transaction details pages as the Reference Retrieval Number (RRN).
- **MerchTxnRef** is generated by the merchant shop and buy application. Ideally it should be a unique value for each transaction, and the merchant should retain this number so that transaction can be traced within the merchant's application and the MIGS Payment Server.
- **TransactionID** is a unique number generated by the MIGS Payment Server that matches the shopping transaction number. The shopping transaction number is only relevant to a shopping transaction, for example Auth transactions. It is the key reference value for transactions when using AMA transactional functions like captures and refunds.
- The **AuthorizeID** field in the Digital Receipt is another identifier that is passed in the Digital Receipt and sent by the issuing bank for the authorisation. This field cannot be searched for in Merchant Administration or AMA but it is displayed in Merchant Administration as the "Authorisation Code". It is one of the fields returned in an AMA query and the AMA transaction result (captures, refunds).



## Cardholder Authentication

### MasterCard SecureCode and Verified by Visa

#### Introduction

MasterCard and Visa have introduced new authentication methods for internet payments, which involves validating the presence of the cardholder in a traditional internet card-not-present environment.

Essentially, this involves the cardholder entering a password known only to themselves and their issuing bank during the course of the payment, similar to the use of a PIN during an ATM transaction. This reduces the likelihood of fraud and customer chargeback.

MIGS supports the MasterCard initiative, SecureCode® and the Visa initiative, Verified by Visa®, which authenticates the cardholder by redirecting their internet browser to their card issuers authentication server, which is accessible via the internet. Both SecureCode and Verified by Visa implement an authentication process called Authentication, and MIGS supports the consolidation of both card scheme implementations of Authentication. This consolidation makes the whole authentication process

easier for the merchant. MIGS allows authentication to be performed without any changes to the merchant 'request for payment' process.

Authentication is performed as an integral part of an Authorization or Purchase transaction.

From a cardholders experience, there is simply a new step added, where the cardholder is redirected to the issuers 3DSecure Access Control Server (ACS) and a password is inputted by the cardholder. If the password matches the password selected by the cardholder at the point of enrolment in their Issuers authentication program, then the transaction is considered authenticated and payment can proceed.

If the cardholder does not enter the correct password, and therefore cannot authenticate themselves, MIGS will not proceed with the payment.

Other failures (for example, communication errors) may result in the authentication attempt failing, but the payment going ahead. The general rule is that if authentication is possible it will be performed, but if it is performed it must succeed otherwise the payment will not be processed.

### Authentication Process Flow

The MIGS interaction with any other entity apart from the cardholder or merchant is described for information only, as the merchant will only witness passing of control to MIGS and the return of authentication data if authentication was attempted.

It is important to note that the merchant has no control over an authentication attempt if he is configured for Authentication.

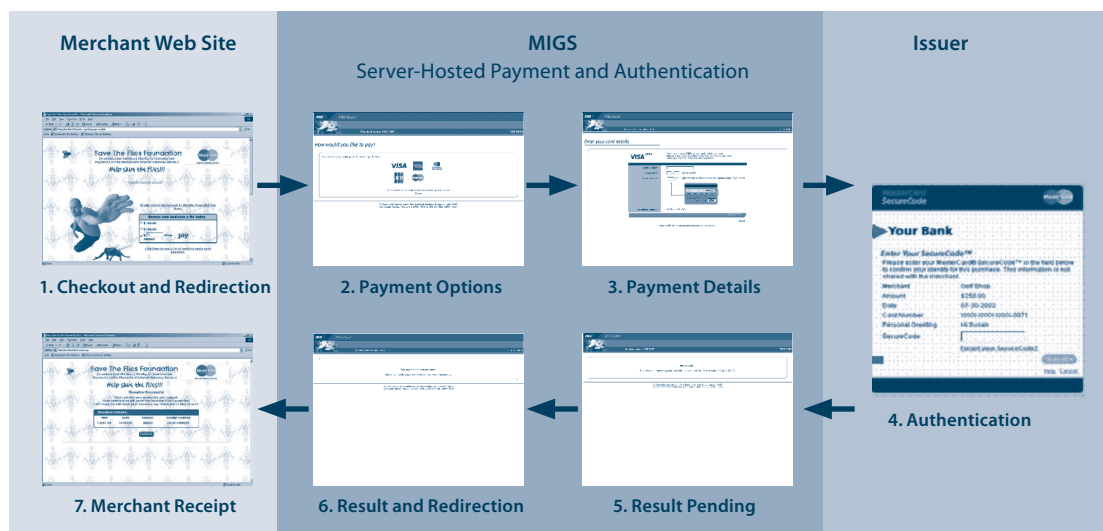
MIGS will detect the submission of a MasterCard or Visa Card by the cardholder and, if the merchant has been enabled for Authentication, MIGS will interrogate the MasterCard or Visa Directory Service to check if the cardholder is enrolled in his issuer Authentication program.

If the cardholder is not enrolled, or the issuer does not support Authentication, authorisation is performed as normal.

If the cardholder is enrolled, the MasterCard or Visa directory service will return the URL of the issuer's ACS, and MIGS will redirect the cardholder's browser to this ACS to allow the issuer to authenticate. The ACS then returns the cardholder's browser to MIGS, along with the result of the authentication attempt.

MIGS will continue with the authorization of the transaction if the authentication was successful.

### Server-Hosted Payment and Authentication Process Flow



For a Server-Hosted transaction the merchant has no extra considerations on the submission of the payment request, but may elect to accept and record the new Authentication result fields, described later in this section.

## Payment Transactions for Server-Hosted Payments

Server-Hosted Payments requires you to use <https://migs.mastercard.com.au/vpcpay> URL for the Virtual Payment Client. You must use HTTPS protocol or the VPC will reject the transaction request.

### Transaction Request Fields

Transaction requests contain the information collected for a cardholder's order that is used for processing by the MIGS Payment Server. The transaction request must include all

the required fields for Server-Hosted Payments. You can also include optional fields such as Verified-by-Visa and MasterCard Secure Code.

### Required Transaction Request fields for a Server-Hosted Payment Request

The required fields that must be included in a transaction request when using Server-Hosted Payments are:

Field Name	Required/Optional	Field Type	Length	Example Value
vpc_Version	The version of the Virtual Payment Client API being used. The current version is 1.			
	Required	Numeric	1,8	1
vpc_Command	Indicates the transaction type. This must be equal to <b>pay</b> .			
	Required	Alphanumeric	1,16	pay
vpc_MerchTxnRef	A unique value created by the merchant to identify the transaction request. It is used to track the progress of a transaction and allows it to be identified on the Payment Server should a communications failure occur and the transaction response is not received. It can contain similar information to the vpc_OrderInfo field, but it must be unique. It may be in part an order number or invoice number, but it should also reflect the transaction attempt. For example, if a cardholder has insufficient funds on their card and you allow them to repeat the transaction with another credit card. The value may be test1234/1 on the first attempt, test1234/2 on the second attempt and test1234/3 on the third attempt. It can use text made up of any of the base US ASCII characters in the range, hexadecimal 20 to 126.			
	Required	Alphanumeric - Special characters	1,40	test1234/1
vpc_AccessCode	The access code authenticates you on the Payment Server so that a merchant cannot access another merchant's Merchant ID. The access code is provided to you when you registered your merchant profile with your Payment Provider.			
	Required	Alphanumeric	8	6ab89f3
vpc_Merchant	The unique merchant ID assigned to you by your Payment Provider.			
	Required	Alphanumeric	1,16	TESTMERCHANT01
vpc_OrderInfo	Your own identifier used to identify the transaction with the cardholder. For example, a shopping cart number, an order number, or an invoice number.			
	Required	Alphanumeric - Special characters	1,34	test1234
vpc_Amount	The amount of the transaction in the smallest currency unit expressed as an integer. For example, if the transaction amount is \$49.95 then the amount in cents is 4995.			
	Required	Numeric	1,10	4995
vpc_Locale	Used in SSL type transactions for specifying the language that is used on the Payment Server pages that are displayed to the cardholder. If the locale is not supplied the Payment Server defined default of 'en' is used.			
	Required	Alphanumeric	2,5	en
vpc_ReturnURL	The URL that is displayed to the cardholder's browser when the Payment Server sends the transaction response. It must be a complete URL. The Return URL must start with either http:// or https:// and may be up to 255 characters. If the return URL is not supplied, your default vpc_ReturnURL that you nominated when you registered your merchant profile with your Payment Provider is used.			
	Required	Alphanumeric -Special characters	1,255	http://returnurl/Receipt.asp

Required Transaction Response Fields for Server Managed Payments

## Transaction Request fields for a Server-Hosted Payment Request

The required and optional fields that can be included in a transaction request when using Server-Hosted Payments are:

Field Name	Required/Optional	Field Type	Length	Example Value
Secure Hash – Optional Transaction Request Fields				
vpc_SecureHash	Used to allow the Virtual Payment Client to check the integrity of the transaction request.			
	Required	Alphanumeric	64	9FF46885DCA8563ACFC620 58E0FC447BD2C033D505BD 8202F681DCAD7CED4DD2
Secure Hash Type – Optional Transaction Request Fields				
vpc_SecureHashType	Used to allow the Virtual Payment Client to check the integrity of the transaction request			
	Required	Alphanumeric	6	SHA256
Verified-by-Visa and MasterCard SecureCode - Optional Transaction Request Fields				
Verified-by-Visa and MasterCard SecureCode are payment authentications designed to prevent credit card fraud by redirecting cardholders to their card issuer where they enter a password that they had previously registered with their card issuer.				
No additional input fields are required for a Verified-by-Visa or MasterCard Secure Code payment authentication, but you do need ANZ to enable you to use Verified-by-Visa or MasterCard SecureCode.				
Card Security Code (CSC) - Optional Transaction Request Fields				
The Card Security Code (CSC) is a security feature used for card not present transactions that compares the Card Security Code on the card with the records held in the card issuer's database.				
If the Payment Provider allows you to set your own CSC level, you may override the Payment Server default level on a per transaction basis. You can then specify the minimum level that you wish to accept for the current transaction using the vpc_CSCLevel field.				
vpc_CSCLevel	You may set this value to the minimum CSC level that they are willing to accept for this transaction. If no value is provided, your default value set in the Payment Server will be used. If this value is present then vpc_CardNum and vpc_CardExp must also be present as well for the transaction to complete.			
	Optional	Alpha	1	M
	Optional	Alphanumeric - Special characters	1,5	Qld
	Optional	Numeric	1	M
Ticket Number - Optional Transaction Request Fields				
vpc_TicketNo	Allows you to include a ticket number, such as an airline ticket number in the transaction request. The ticket number is stored on the Payment Server database for the transaction. The ticket number is not returned in the transaction response.			
	Optional	Alphanumeric - Special characters	1,16	AB1234

### Optional Transaction Request fields for Server-Hosted Payments

#### Optional Merchant Defined Fields

Server-Hosted Payments also supports up to 5 merchant defined fields that will be returned to you in the transaction response. These fields must be less than 255 bytes and must not start with vpc\_. These fields are not stored in the Payment Server.

#### Get Method

The HTTP Redirect allows the merchant to redirect their URL to the MIGS Payment Server.

#### Sending a Transaction Request for Server-Hosted Payments

The GET method with a Query String containing the transaction request fields, and a HTTPS Redirect, is used to send the transaction request via the Virtual Payment Client to the Payment Server, when using Server-Hosted Payments.

## Transaction Response Fields

### Required Transaction Response fields for Server-Hosted Payment Response

The transaction response contains the results of the transaction request fields that were processed by the MIGS Payment Server. It indicates whether the payment was successful or not. The required fields that are included in the transaction response fields for Server-Hosted Payments are:

Field Name	Required Optional Input	Field Type	Length	Example Value
Server-Hosted Payments – Transaction Response fields				
vpc_Version	The value of the <b>vpc_Version</b> transaction request input field that is returned in the transaction response.			
	Input	Numeric	1,2	1
vpc_Command	The value of the <b>vpc_Command</b> transaction request input field that is returned in the transaction response.			
	Input	Alpha	3	pay
vpc_MerchTxnRef	The value of the <b>vpc_MerchTxnRef</b> transaction request input field that is returned in the transaction response.			
	Input	Alphanumeric – Special characters	1,40	test1234/1
vpc_Merchant	The value of the <b>vpc_Merchant</b> transaction request input field that is returned in the transaction response.			
	Input	Alphanumeric – Special characters	1,16	TESTMERCHANT01
vpc_OrderInfo	The value of the <b>vpc_OrderInfo</b> transaction request input field that is returned in the transaction response.			
	Input	Alphanumeric – Special characters	1,34	test1234
vpc_Amount	The value of the <b>vpc_Amount</b> transaction request input field that is returned in the transaction response.			
	Input	Numeric	1,10	4995
vpc_Locale	The value of the <b>vpc_Locale</b> transaction request input field that is returned in the transaction response. It specifies the language that is used on the Payment Server pages that are displayed to the cardholder. If the Locale is not supplied in the transaction request, the default value of 'en' (English) used in the Payment Server.			
	Input	Alphanumeric	2,5	en
vpc_TxnResponseCode	A response code that is generated by the Payment Server to indicate the status of the transaction. A <b>vpc_TxnResponseCode</b> of "0" (zero) indicates that the transaction was processed successfully and approved by the Acquiring Bank. Any other value indicates the transaction was declined.			
	Required	Alphanumeric	1	0
vpc_TransactionNo	A unique number generated by the Payment Server for the transaction. It is stored in the Payment Server as a reference and used to perform actions such as a refund or capture.			
	Required	Numeric	1,12	3465
vpc_Message	Indicates any errors the transaction may have encountered.			
	Optional	Alphanumeric	10,200	Merchant [TESTCORE23] does not exist

Field Name	Required Optional Input	Field Type	Length	Example Value
Server-Hosted Payments – Transaction Response fields				
vpc_AcqResponseCode	Acquirer's Response Code is generated by the financial institution to indicate the status of the transaction. The results can vary between institutions so it is advisable to use the <b>vpc_TxnResponseCode</b> as it is consistent across all acquirers. It is only included for fault finding purposes.			
	Optional	Alphanumeric	2,3	00
vpc_ReceiptNo	This is also known as the Reference Retrieval Number (RRN), which is a unique identifier. This value is passed back to the cardholder for their records if the merchant application does not generate its own receipt number.			
	Optional	Alphanumeric	1,12	RP12345
vpc_BatchNo	A date supplied by the acquirer to indicate when this transaction will be settled. If the batch has today's date then it will be settled the next day. When the acquirer closes the batch at the end of the day, the date will roll over to the next processing day's date.			
	Optional	Alphanumeric	1,8	20021021
vpc_AuthorizId	An identifying code issued by the bank to approve or deny the transaction. This is an optional field and may not be supplied by all acquirers.			
	Optional	Alphanumeric	1,12	ABC12345
vpc_Card	A code issued by the Payment Server for the card type used by the cardholder in the transaction.			
	Optional	Alphanumeric	0,2	MC

Required Transaction Response Fields for Server-Hosted Payments

### Optional Transaction Response fields for Server-Hosted Payment Response

If you integrate advanced functionality when using Server-Hosted Payments, then optional fields that can be included in a transaction response from the Payment Server are:

Field Name	Required Optional Input	Field Type	Length	Example Value
Secure Hash – Transaction Response field				
vpc_SecureHash	This field is only returned for a Server-Hosted Payment as the response is returned via the cardholder's browser as a QueryString, which is visible to the cardholder. It allows you to check message integrity to ensure the response values have not been tampered with.			
	Optional	Alphanumeric	64	9FF46885DCA8563ACFC62058E0FC447BD2C033D505BD8202F681DCAD7CED4DD2
Secure Hash Type – Optional Transaction Request Fields				
vpc_SecureHashType	This field is only returned for a Server-Hosted Payment as the response is returned via the cardholder's browser as a QueryString, which is visible to the cardholder. It allows you to check message integrity to ensure the response values have not been tampered with.			
	Optional	Alphanumeric	6	SHA256
Verified-by-Visa and MasterCard SecureCode – Transaction Response fields				
<p>These fields are only returned in the transaction response if the transaction is a Verified-by-Visa and MasterCard SecureCode payment authentication. You must be enabled on the Payment Server by your bank to perform Verified-by-Visa and MasterCard SecureCode payment authentications.</p> <p>The <b>vpc_TxnResponseCode</b> is used to determine if the authentication passed or a failed.</p> <p>If the <b>vpc_TxnResponseCode</b> is not equal to 'F', the payment authentication passed OK and the Authentication process has completed satisfactorily.</p> <p>If the <b>vpc_TxnResponseCode</b> is equal to 'F', the Authentication process failed and no payment took place.</p> <p>If a payment authentication has been successful, extra fields are returned in the transaction response for a Verified-by-Visa and MasterCard Secure Code payment authentication. The fields are not used by you but are returned to allow you to store them as a record of authentication for the transaction, which can be used to resolve disputes. They cannot be used again for any future transactions.</p> <p>All payment authentication transactions use a <b>vpc_VerStatus</b> response code value to show whether the card authentication was successful or not.</p>				
vpc_VerType	Either '3DS' or 'SPA'.			
	Optional	Alphanumeric	3,20	3DS
vpc_VerStatus	The status codes used by the Payment Server.			
	Optional	Alphanumeric	1	N
vpc_VerSecurityLevel	<p>The Verification Security Level is generated at the card issuer as a token to prove that the cardholder was enrolled and authenticated OK. It is shown for all transactions except those with authentication status "Failure". This field contains the security level to be used in the AUTH message.</p> <p>MasterCard '0' -Merchant not participating (a merchant will not see this if they are configured for MasterCard SecureCode).</p> <p>MasterCard '1'-Cardholder not participating.</p> <p>MasterCard '2'-Cardholder authenticated.</p> <p>Visa '05' -Fully Authenticated.</p> <p>Visa '06' -Not authenticated, (cardholder not participating), liability shift.</p> <p>Visa '07' - Not authenticated. Usually due to a system problem, for example the merchant password is invalid.</p>			
	Optional	Numeric	1,2	06
vpc_VerToken	This value is generated by the card issuer as a token to prove that the cardholder authenticated OK. This is a base64 encoded value.			
	Optional	Alphanumeric	28	gIGCg4SFhoeliYqLj2Oj5CR kpM=

Field Name	Required Optional Input	Field Type	Length	Example Value
<b>Verified-by-Visa and MasterCard SecureCode – Transaction Response fields</b>				
<b>vpc_3DSXID</b>	It is a unique transaction identifier that is generated by the merchant to identify the 3DS transaction. It is a 20-byte field that is Base64 encoded to produce a 28-character value.			
	Optional	Alphanumeric	28	uyPfGlgsoFQhklkIsto+IFWs9 2s=
<b>vpc_3DSECI</b>	The 3-D Secure Electronic Commerce Indicator, which is set to '05' when the cardholder authenticates OK, and '08' when the cardholder is not enrolled. (These values may change depending on the locale or issuer).			
	Optional	Numeric	2	08
<b>vpc_3DSenrolled</b>	This field is only included if the card is within an enrolled range. This is the value of the VERes. enrolled field. It will take values (Y - Yes, N - No, U – Unavailable for Checking).			
	Optional	Alpha	1	N
<b>vpc_3DSstatus</b>	This field is only included if payment authentication was used and a PAREs was received by the MPI. It will take values (Y – Yes, N – No, A – Attempted Authentication, U – Unavailable for Checking).			
	Optional	Alpha	1	N
<b>Card Security Code (CSC) - Transaction Response fields</b>				
<b>vpc_CSCResultCode</b>	The Card Security Code result code indicates the CSC level used to match the data held by the cardholder Issuing Bank.			
	Optional	Alphanumeric	1	M
<b>vpc_CSCRequestCode</b>	The CSC level that was requested in the Payment Server for the transaction. If the CSC Level value was not sent, then this will be your default CSC level set in the Payment Server.			
	Optional	Alphanumeric	1	S
	Optional	Alpha	1	G

#### Optional Transaction Response Fields for Server-Hosted Payments

##### Receiving the Transaction Response

To receive the transaction response, you must specify a return Internet address (return /URL). This address is also where the cardholder is returned to when they have completed the purchase. The vpc\_ReturnURL field must contain a valid URL (starting with "http://" or "https://") for every transaction request. If the ReturnURL value does not form a valid URL, an error is generated in the Payment Server which will stop the transaction.

##### Calculating and Validating the Secure Hash Secret

Secure Hash Secret is used to detect whether the transaction request and response has been tampered with. It is added to the transaction request details before an SHA256 algorithm is applied to generate a secure hash. The secure hash is then sent to the Payment Server with the transaction request details. Because the Payment Server is the only other entity apart from you that knows your secure hash secret it recreates the same secure hash and matches it with the one that you sent. If they match the Payment server continues processing the transaction. If it doesn't match, it assumes that the transaction request has been tampered with and will stop processing the transaction and send back an error message.

##### How the Secure Hash is Created and Verified

The vpc\_SecureHash field is used for the SHA256 HMAC (FIPS 180-2) secure hash of your secure hash secret and the transaction request. The secure hash value is the Hex encoded SHA256 HMAC output of the transaction request or response fields with the Secure Hash Secret used as a key. The order that the fields are hashed in are:

- All transaction request fields, except the Secure Hash Type are concatenated to the Secure Hash Secret in alphabetical order of the field name. The sort should be in ascending order of the ASCII value of each field string. If one string is an exact substring of another, the smaller string should be before the longer string. For example, Card should come before CardNum.
- Fields must not have any separators between them and must not include any null terminating characters.



For example, if the Secure Hash Secret is **DA7A193F76C2FCB8187300D790C7F23**, and the transaction request includes the following fields:

Field Name	Example Value
vpc_Version	1
vpc_Commnd	pay
vpc_MerchTxnRef	tx1
vpc_Merchant	TESTANZ
vpc_AccessCode	D16B1C2C
vpc_Amount	1000

#### Example of a Secure Hash Calculation

In ascending alphabetical order the transaction request fields inputted to the SHA256 hash would be:

The concatenated value is as follows:

**vpc\_AccessCode=D16B1C2C&vpc\_Amount=1000&vpc\_Command=pay&vpc\_MerchTxnRef=tx1&vpc\_Merchant=TESTANZ&vpc\_OrderInfo=order1&vpc\_Version=1**

**Note:** The last character of each field value (other than the last) is followed directly by "&". The concatenated value must be represented in the UTF-8 character encoding format.

**Note:** The values in all name value pairs should not be URL encoded for the purpose of hashing.

The Secure Hash value is:

**753A21929C9C53D6777C00DC55A5F5A9D1105A4D309E5C5AEEF51FA6DEE4F0CC**

and the resultant Request is (note the Secure Hash and Secure Hash Type fields):

**vpc\_AccessCode=D16B1C2C&vpc\_Amount=1000&vpc\_Command=pay&vpc\_MerchTxnRef=tx1&vpc\_Merchant=TESTANZ&vpc\_OrderInfo=order1&vpc\_Version=1&vpc\_SecureHash=753A21929C9C53D6777C00DC55A5F5A9D1105A4D309E5C5AEEF51FA6DEE4F0CC&vpc\_SecureHashType=SHA256**

**Note:** Non-VPC fields (fields that do not begin with "vpc\_") are returned ONLY for 3-Party integrations. In the Transaction Response, - the values for these fields cannot exceed 255 characters - the maximum number of fields returned is 5 - the maximum length of the response string in the URL is 2048 characters.

#### Adding Secure Hash to a Transaction Request

Although the risk of a cardholder tampering with the transaction request is minimal, it is recommended that you include a Secure Hash in your transaction request. If a cardholder changes a transaction request, it will be detected because if the Secure Hash generated by the Payment Server does not match the one generated by you, the payment is rejected.

If the secure hash does not match, the Virtual Payment Client will immediately return the cardholder to the

merchant's site with an error, by setting the **vpc\_TxnResponseCode** field to 7 to indicate that the secure hash is incorrect.

- During integration, this may mean that you have not calculated your hash properly.
- During production, this would usually mean that a cardholder is attempting to commit fraud.

To create a Secure Hash, the following fields are required for a transaction request using Server-Hosted Payments.

```
<input type="hidden" name="vpc_Version" value="1">
<input type="hidden" name="vpc_AccessCode" value="6ab89f3">
<input type="hidden" name="vpc_MerchantId" value="TESTWEBANZ01">
<input type="hidden" name="vpc_OrderInfo" value="test1234">
<input type="hidden" name="vpc_Amount" value="4995">
<input type="hidden" name="vpc_Locale" value="en">
<input type="hidden" name="vpc_ReturnURL" value="http://192.168.21.205/Receipt.asp">
<input type="hidden" name="vpc_SecureHashType" value="SHA256">
<input type="hidden" name="Secure_Secret" value="ebe65403de22d35c7685cb8403315c00">
```

The fields in the transaction request, except the Secure Hash Type must be concatenated in ascending alphabetical order with the Secure Hash Secret used as the key:

**sha256\_input = vpc\_AccessCode + vpc\_Amount + vpc\_Locale + vpc\_MerchantId + vpc\_OrderInfo + vpc\_ReturnURL + vpc\_Transaction + vpc\_Version**

The order used is the Virtual Payment Client field names,

```
<input type="hidden" name="vpc_TicketNo" value="ABC123">
```

And the sha256\_input would then become:

**sha256\_input = vpc\_AccessCode + vpc\_Amount + vpc\_Locale + vpc\_MerchantId + vpc\_OrderInfo + vpc\_ReturnURL + vpc\_TicketNo + vpc\_Transaction + vpc\_Version**

not the alphabetical order of the names you may use in your online store.

Any extra functionality fields must be also concatenated to the sha256\_input in ascending alphabetical order as shown above, for example, if Ticket Number functionality is added, then the extra field to be added is:

You should also ensure that:

- UTF-8 encoding should be used to convert the input from a printable string to a byte array. Note that 7-bit ASCII encoding is unchanged for UTF-8.
- The hash output must be hex-encoded.

### Adding Secure Hash to a Transaction Response

When you receive the transaction response from the Payment Server, you should calculate the Secure Hash and compare it to the Secure Hash from the Payment Server to ensure that the data has not been tampered with in the transaction response. If you do not check the Secure Hash,

the transaction response can be retrieved securely from the Payment Server using QueryDR. To create a Secure Hash, the following fields are required for a transaction response using Server-Hosted Payments.

```
String version = req.getParameter("vpc_Version ");
String merchantId = req.getParameter("vpc_MerchantId");
String orderInfo = req.getParameter("vpc_OrderInfo");
String amount = req.getParameter("vpc_Amount");
String locale = req.getParameter("vpc_Locale");
String txnResponseCode = req.getParameter("vpc_TxnResponseCode");
String acqResponseCode = req.getParameter("vpc_AcqResponseCode");
String receiptNo = req.getParameter("vpc_ReceiptNo");
String xtnNo = req.getParameter("vpc_TransactionNo");
String batchNo = req.getParameter("vpc_BatchNo");
String authorizeID = req.getParameter("vpc_AuthorizeID");
String secureHashType = req.getParameter("vpc_SecureHashType");
String resp_Secure_Hash = req.getParameter("vpc_SecureHash");
```

The fields in the transaction response must be concatenated in ascending alphabetical order with the Secure Secret used as the key:

**sha256\_input = amount + authorizeID + batchNo + locale + merchantId + orderInfo + qsiResponseCode + receiptNo + transactionNo + version**

The order used is the Virtual Payment Client field names, not the alphabetical order of the names you may use in your online store.

You should also ensure that:

- UTF-8 encoding should be used to convert the input from a printable string to a byte array. Note that 7-bit ASCII encoding is unchanged for UTF-8.
- The hash output must be hex-encoded.

## Payment Transactions for Merchant-Hosted Payment

Merchant-Hosted Payments requires you to use <https://migs.mastercard.com.au/vpcdps> URL for the Virtual Payment Client. You must use HTTPS protocol or the Virtual Payment Client will reject the Transaction Request. In Merchant-Hosted Payments the online store application connects directly to the Virtual Payment Client using a form POST operation that directly returns a response. Since the Payment Server cannot collect cardholder card details, they must be collected on your site and sent to the Virtual Payment Client. During Merchant-Hosted Payments, session variables do not need to be sent to the Payment Server because the merchant's session is not broken as it is in Server-Hosted Payments, where the cardholder's Internet browser is disconnected from the merchant's site and redirected to the Payment Server. This means that in

Merchant-Hosted Payments, the cardholder browser is not redirected, so advanced functionality such as Verified-by-Visa and MasterCard Secure Code cannot be used.

### Transaction Request Fields

The transaction request contains the required information for a cardholder's order that is sent via the Virtual Payment Client to the Payment Server.

#### Required Transaction Request fields for Merchant-Hosted Payment Request

The required fields that must be included in a transaction request when using Merchant-Hosted Payments are:

Field Name	Required Optional Input	Field Type	Length	Example Value
vpc_Version	The version of the OByte Virtual Payment Client API being used. The current version is 1.			
	Required	Numeric	1,8	1
vpc_Command	Indicates the type of transaction type. It must be equal to 'pay'			
	Required	Alpha	1,16	pay
vpc_MerchTxnRef	A unique value created by you to identify the transaction request. It is used to track the progress of a transaction and allows it to be identified on the Payment Server should a communications failure occur and the transaction response is not received. It can contain similar information to the vpc_OrderInfo field, but it must be unique. It may be in part an order number or invoice number, but it should also reflect the transaction attempt. For example, if a cardholder has insufficient funds on their card and you allow them to repeat the transaction with another credit card. The value may be test1234/1 on the first attempt, test1234/2 on the second attempt and test1234/3 on the third attempt. It can use text made up of any of the base US ASCII characters in the range, hexadecimal 20 to 126.			
	Required	Alphanumeric -Special characters	1,40	test1234/1
vpc_AccessCode	The access code is used to authenticate you on the Payment Server so that a merchant cannot access another merchant's MerchantId. The access code is provided to you when you registered your merchant profile with the Payment Provider.			
	Required	Alphanumeric	8	6ab89f3
vpc_Merchant	The unique merchant ID assigned to you by your Payment Provider.			
	Required	Alphanumeric	1,16	TESTMERCHANT01
vpc_OrderInfo	An identifier provided by you to identify the transaction with the cardholder. It can be a shopping cart number, an order number, or an invoice number.			
	Required	Alphanumeric -Special characters	1,34	test1234
vpc_Amount	The amount of the transaction in the smallest currency unit expressed as an integer. For example, if the transaction amount is \$49.95 then the amount in cents is 4995.			
	Required	Numeric	1,10	4995
vpc_CardNum	This field is used to bypass the card details page on the Payment Server. It is the number of the card to be used for processing the payment. It can only be a long integer value with no white space or formatting characters.			
	Required	Numeric	15,40	5123456789012346

Field Name	Required Optional Input	Field Type	Length	Example Value
vpc_CardExp	The expiry date of the card to be processed for payment. The format for this is YYMM, for example, for an expiry date of May 2009, the value would be 0905. The value must be expressed as a 4-digit number (integer) with no white space or formatting characters			
	Required	Numeric	4	0504

Required Transaction Request fields for Merchant-Hosted Payments

#### Optional Transaction Request fields for Merchant-Hosted Payment Request

The optional fields that can be included in a transaction request to the Virtual Payment Client when using Merchant-Hosted Payments are:

Field Name	Required	Optional Input	Field Type	Length	Example Value
Card Security Code (CSC) – Optional Transaction Request Fields					
The Card Security Code (CSC) is a security feature used for card not present transactions that compares the Card Security Code on the card with the records held in the card issuer’s database.					
vpc_CardSecurityCode	The Card Security Code (CSC) is a security feature used for card not present transactions that compares the Card Security Code on the card with the records held in the card issuer’s database. For example, on Visa and MasterCard credit cards, it is the three digit value printed on the signature panel on the back following the credit card account number. For American Express, the number is the 4 digit value printed on the front above the credit card account number. Once the transaction is successfully processed and authorised, the card issuer returns a result code (CSC result code) in its authorisation response message verifying the CSC level (vpc_CSCLevel) of accuracy used to match the card security code.				
	Optional		Numeric	1,4	123
vpc_CSCLevel	You may set this value to the minimum CSC level that you are willing to accept for this transaction. If you do not set a value, your default value will be used.				
	Optional		Alpha	1	M
	Optional		Alpha	3	AUS
	Optional		Numeric	1	M
Transaction Source – Optional Transaction Request Fields					
vpc_TxSource	Allows the merchant to specify the source of the transaction. Valid values are: <b>INTERNET</b> - indicates an Internet transaction. <b>MOTOCC</b> - indicates a call centre transaction. <b>MOTO</b> - indicates a mail order or telephone order. <b>MAILORDER</b> - indicates a mail order transaction. <b>TEORDER</b> - indicates a telephone order transaction. <b>CARDPRESENT</b> - indicates that the merchant has sighted the card. <b>VOICERESPONSE</b> - indicates that the merchant has captured the transaction from an IVR system.				
Ticket Number – Optional Transaction Request Fields					
vpc_TicketNo	This allows the merchant to include a ticket number, such as an airline ticket number in the transaction request. The ticket number is stored on the Payment Server database for that transaction. The ticket number value is not returned in the transaction response.				
	Optional		Alphanumeric -Special characters	1,16	AB1234

Optional Transaction Request fields for Merchant-Hosted Payments

## Sending a Transaction Request for Merchant-Hosted Payments

### Post Method

The Post Method is used when you collect the cardholder's card details. The data is collected in a secure form and included in the transaction request that is sent directly to the Payment Server.

## Sending a Transaction Request using the Post Method

The following post method example shows the minimum number of fields required to complete a transaction using the Merchant-Hosted Payment integration model:

```
<form method="POST" action="https://www.<vpc_name>/vpcdps">
<input type="hidden" name="vpc_Version" value="1">
<input type="hidden" name="vpc_Command" value="pay">
<input type="hidden" name="vpc_AccessCode" value="6ab89f3">
<input type="hidden" name="vpc_MerchTxnRef" value="test1234/1">
<input type="hidden" name="vpc_MerchantId" value="TESTWEBANZ01">
<input type="hidden" name="vpc_OrderInfo" value="test1234">
<input type="hidden" name="vpc_Amount" value="4995">
<input type="hidden" name="vpc_CardNum" value="5123456789012346">
<input type="hidden" name="vpc_cardExp" value="0405">
<!-- submit -->
<input type="submit" value="Pay Now">
</form>
```

## Transaction Response Fields

### Required Transaction Response fields for Merchant-Hosted Payment Response

The transaction response contains the results of the transaction request that was processed by the Payment Server.

The fields that are included in a transaction response from the Virtual Payment Client when using Merchant-Hosted Payments are:

Field Name	Required Optional Input	Field Type	Length	Example Value
Merchant-Hosted Payments - Transaction Response fields				
vpc_Version	The value of the <b>vpc_Version</b> input field returned in the transaction response.			
	Input	Numeric	1,2	1
vpc_Command	The value of the <b>vpc_Command</b> field returned in the transaction response.			
	Input	Alpha	1,16	pay
vpc_MerchTxnRef	The value of the <b>vpc_MerchTxnRef</b> field returned in the transaction response.			
	Input	Alphanumeric - Special characters	1,40	test1234/1
vpc_Merchant	The value of the <b>vpc_Merchant</b> input field returned in the transaction response.			
	Input	Alphanumeric - Special characters	1,16	TESTMERCHANT01
vpc_OrderInfo	The value of the <b>vpc_OrderInfo</b> input field returned in the transaction response.			
	Input	Alphanumeric -Special characters	1,34	test1234

Field Name	Required Optional Input	Field Type	Length	Example Value
vpc_Amount	The value of the <b>vpc_Amount</b> input field returned in the transaction response.			
	Input	Numeric	1,10	4995
vpc_Locale	Locale is not supplied in the transaction request but is returned in the transaction response. It is not used in Merchant-Hosted Payments.			
	Input	Alphanumeric	2,5	en
vpc_TxnResponseCode	A response code that is generated by the Payment Server to indicate the status of the transaction. A <b>vpc_TxnResponseCode</b> of "0" (zero) indicates that the transaction was processed successfully and approved by the acquiring bank. Any other value indicates the transaction was declined.			
	Required	Alphanumeric	1	0
vpc_TransactionNo	A unique number generated by the Payment Server. It is the reference value of the transaction in the Payment Server. This is the Shopping Transaction number that must be used for a Refund or Capture operation.			
	Required	Numeric	1,12	3465
vpc_Message	This is a message to indicate what sort of errors the transaction encountered.			
	Optional	Alphanumeric	10,200	Merchant [TESTCORE23] does not exist.
vpc_AcqResponseCode	Acquirer's Response Code is generated by the financial institution to indicate the status of the transaction. The results can vary between institutions so it is advisable to use the vpc_TxnResponseCode as it is consistent across all acquirers. It is only included for fault finding purposes.			
	Optional	Alphanumeric	2,3	00
vpc_ReceiptNo	This is also known as the Reference Retrieval Number (RRN), which is a unique identifier. This value is passed back to the cardholder for their records if the merchant application does not generate its own receipt number.			
	Optional	Alphanumeric -Special characters	1,12	RP12345
vpc_BatchNo	A date supplied by an acquirer to indicate when this transaction will be settled. If the batch has today's date then it will be settled the next day. When the acquirer closes the batch at the end of the day, the date will roll over to the next processing day's date.			
	Optional	Alphanumeric	1,8	20021021
vpc_AuthorizeId	A code issued by the acquiring bank to approve or deny the transaction. This may not always be supplied by all acquirers.			
	Optional	Alphanumeric	1,12	ABC12345
vpc_Card	A code issued by the Payment Server for the card type used by the cardholder for the transaction.			
	Optional	Alphanumeric	0,2	MC

Transaction Response Fields for Merchant-Hosted Payments

### Optional Transaction Response fields for Merchant-Hosted Payment Response

If you integrate advanced functionality when using Merchant-Hosted Payments, then optional fields that can be included in a transaction response from the Payment Server are:

Field Name	Required Optional Input	Field Type	Length	Example Value
<b>Card Security Code (CSC) - Transaction Response fields</b>				
<b>vpc_CSCResultCode</b>	The result code generated by the Payment Server in relation to the Card Security Code.			
	Optional	Alpha	1	S
<b>vpc_CSCRequestCode</b>	The CSC level that was requested in the Payment Server for the transaction. If the CSC Level value was not sent, then this will be the merchant's default CSC level set in the Payment Server.			
	Optional	Alpha	1	M
<b>vpc_AcqCSCRespCode</b>	The result code generated by the acquiring bank in relation to the Card Security Code.			
	Optional	Alpha	1	M
	Optional	Alpha	1	S

Table 1 Optional Transaction Response Fields for Merchant-Hosted Payments

# Advanced Functionality Fields

## Capture

### Transaction Request Fields - Capture

The fields that can be included in a transaction request to the Virtual Payment Client when using capture are:

Field Name	Required Optional Input	Field Type	Length	Example Value
vpc_Version	The version of the 0Byte Virtual Payment Client API being used. The current version is 1.			
	Required	Numeric	1,8	1
vpc_Command	Used to indicate the type of payment. The value, capture is used.			
	Required	Alpha	1,16	Capture
vpc_MerchTxnRef	A unique value created by the merchant to identify the transaction request. It is used to track the progress of a transaction and allows it to be identified on the Payment Server should a communication's failure occur and the transaction response is not received. It can contain similar information to the vpc_OrderInfo field, but it must be unique. It may be in part an order number or invoice number, but it should also reflect the transaction attempt. For example, if a cardholder has insufficient funds on their card and you allow them to repeat the transaction with another credit card. The value may be test1234/1 on the first attempt, test1234/2 on the second attempt and test1234/3 on the third attempt. It can use text made up of any of the base US ASCII characters in the range, hexadecimal 20 to 126.			
	Required	Alphanumeric - Special characters	1,40	test1234/1
vpc_AccessCode	The access code authenticates a merchant on the Payment Server so that a merchant cannot access another merchant's MerchantId. The access code is provided to you when you registered your merchant profile with the Payment Provider.			
	Required	Alphanumeric	8	6ab89f3
vpc_Merchant	The unique merchant ID assigned to you by your Payment Provider.			
	Required	Alphanumeric	1,16	TESTMERCHANT01
vpc_TransactionNo	The transaction reference number of the original authorisation or purchase transaction.			
	Required	Numeric	1,12	123
vpc_Amount	The amount of the transaction in the smallest currency unit expressed as an integer. For example, if the transaction amount is \$49.95 then the amount in cents is 4995.			
	Required	Numeric	1,10	4995
vpc_User	This field is a special AMA user created to allow this function to operate.			
	Required	Alphanumeric	1,16	amauser
vpc_Password	The password used to authorise the AMA user to access this function.			
	Required	Alphanumeric	1,16	Password12

Transaction Request Fields for a Capture



### Transaction Response Fields - Capture

The fields included in a transaction response from the Virtual Payment Client when using captures are:

Field Name	Required Optional Input	Field Type	Length	Example Value
vpc_Version	The value of the <b>vpc_Version</b> input field returned in the transaction response.			
	Input	Numeric	1,8	1
vpc_Command	The value of the <b>vpc_Command</b> field returned in the transaction response.			
	Input	Alpha	1,16	capture
vpc_MerchTxnRef	The value of the <b>vpc_MerchTxnRef</b> field returned in the transaction response.			
	Input	Alphanumeric -Special characters	1,40	test1234/1
vpc_Merchant	The value of the <b>vpc_Merchant</b> input field returned in the transaction response.			
	Input	Alphanumeric -Special characters	1,16	TESTMERCHANT01
vpc_Amount	The value of the <b>vpc_Amount</b> input field returned in the transaction response.			
	Input	Numeric	1,10	4995
vpc_TxnResponseCode	A response code that is generated by the Payment Server to indicate the status of the transaction. A <b>vpc_TxnResponseCode</b> of "0" (zero) indicates that the transaction was processed successfully and approved by the acquiring bank. Any other value indicates the transaction was declined.			
	Required	Alphanumeric	1	0
vpc_TransactionNo	A unique number generated by the Payment Server and is the reference value of the transaction in the Payment Server. This is the value that must be used for a Capture.			
	Required	Numeric	1,12	3465
vpc_Message	A message to indicate an error the transaction encountered.			
	Optional	Alphanumeric	10,200	Merchant [TESTCORE23] does not exist
vpc_AcqResponseCode	Acquirer's Response Code is generated by the bank to indicate the status of the transaction. The results can vary between institutions so it is advisable to use the vpc_TxnResponseCode as it is consistent across all acquirers. It is only included for fault finding purposes.			
	Optional	Alphanumeric	2,3	00
vpc_ReceiptNo	This is also known as the Reference Retrieval Number (RRN), which is a unique identifier. This value is passed back to the cardholder for their records if your online store does not generate its own receipt number.			
	Optional	Alphanumeric	1,12	RP12345
vpc_BatchNo	A date supplied by an acquirer to indicate when this transaction will be settled. If the batch has today's date then it will be settled the next day. When the acquirer closes the batch at the end of the day, the date will roll over to the next processing day's date.			
	Optional	Alphanumeric	1,8	20021021
vpc_AuthorizeId	A code issued by the acquiring bank to approve or deny the transaction. This may not always be supplied by all acquirers.			
	Optional	Alphanumeric	1,12	ABC12345

Field Name	Required Optional Input	Field Type	Length	Example Value
vpc_Card	A code issued by the Payment Server to detail the type of card the cardholder used for the transaction.			
	Optional	Alphanumeric	0,2	MC
vpc_ShopTransactionNo	The transaction reference number of the original authorisation or purchase transaction.			
	Optional	Numeric	1,21	3DS
vpc_AuthorisedAmount	The total amount of the original authorisation transaction.			
	Optional	Numeric	1,10	N
vpc_CapturedAmount	The amount of the capture transaction in the smallest currency unit expressed as an integer. For example, if the transaction amount is \$49.95 then the amount in cents is 4995.			
	Optional	Numeric	1,10	4995
vpc_TicketNo	Allows you to include a ticket number, such as an airline ticket number in the transaction request. The ticket number is stored on the Payment Server database for that transaction and returned in the transaction response for capture transactions.			
	Optional	Alphanumeric -Special characters	1,16	Any data

Transaction Response Fields for a Capture

## Refund

### Transaction Request Fields - Refund

The fields that can be included in a transaction request to the Virtual Payment Client when using refund are:

Field Name	Required Optional Input	Field Type	Length	Example Value
vpc_Version	The version of the OByte Virtual Payment Client API being used. The current version is 1.			
	Required	Numeric	1,8	1
vpc_Command	Used to indicate the type of payment. For refunds, the value 'refund' is used.			
	Required	Alpha	1,16	Refund
vpc_MerchTxnRef	<p>A unique value created by the merchant to identify the transaction request. It is used to track the progress of a transaction and allows it to be identified on the Payment Server should a communications failure occur and the transaction response is not received.</p> <p>It can contain similar information to the vpc_OrderInfo field, but it must be unique.</p> <p>It may be in part an order number or invoice number, but it should also reflect the transaction attempt. For example, if a cardholder has insufficient funds on their card and you allow them to repeat the transaction with another credit card. The value may be test1234/1 on the first attempt, test1234/2 on the second attempt and test1234/3 on the third attempt.</p> <p>It can use text made up of any of the base US ASCII characters in the range, hexadecimal 20 to 126.</p>			
	Required	Alphanumeric -Special characters	1,40	test1234/1
vpc_AccessCode	The access code authenticates a merchant on the Payment Server so that a merchant cannot access another merchant's MerchantId. The access code is provided to you when you registered your merchant profile with the Payment Provider.			
	Required	Alphanumeric	8	6ab89f3
vpc_Merchant	The unique merchant ID assigned to you by your Payment Provider.			
	Required	Alphanumeric	1,16	TESTMERCHANT01
vpc_TransactionNo	The transaction reference number of the original Authorisation or Purchase transaction.			
	Required	Numeric	1,12	123
vpc_Amount	The amount of the refund transaction in the smallest currency unit expressed as an integer. For example, if the transaction amount is \$49.95 then the amount in cents is 4995.			
	Required	Numeric	1,10	4995
vpc_User	This field is a special AMA user created to allow this function to operate.			
	Required	Alphanumeric	1,16	amauser
vpc_Password	The password used to authorise the AMA user access to this function.			
	Required	Alphanumeric	1,16	password12

Transaction Request Fields for a Refund

## Transaction Response Fields - Refund

The fields included in a transaction response from the Virtual Payment Client when using refunds are:

Field Name	Required Optional Input	Field Type	Length	Example Value
vpc_Version	The value of the <b>vpc_Version</b> input field returned in the transaction response.			
	Input	Numeric	1,8	1
vpc_Command	The value of the <b>vpc_Command</b> field returned in the transaction response.			
	Input	Alpha	1,16	refund
vpc_MerchTxnRef	The value of the <b>vpc_MerchTxnRef</b> field returned in the transaction response.			
	Input	Alphanumeric -Special characters	1,40	test1234/1
vpc_Merchant	The value of the <b>vpc_Merchant</b> input field returned in the transaction response.			
	Input	Alphanumeric -Special characters	1,16	TESTMERCHANT01
vpc_Amount	The value of the <b>vpc_Amount</b> input field returned in the transaction response.			
	Input	Numeric	1,10	4995
vpc_TxnResponseCode	A response code that is generated by the Payment Server to indicate the status of the transaction. A <b>vpc_TxnResponseCode</b> of "0" (zero) indicates that the transaction was processed successfully and approved by the acquiring bank. Any other value indicates the transaction was declined.			
	Required	Alphanumeric	1	0
vpc_TransactionNo	A unique number generated by the Payment Server. It is the reference value of the transaction in the Payment Server. This is the value that must be used for a Refund.			
	Required	Numeric	1,12	3465
vpc_Message	A message to indicate any errors the transaction may have encountered.			
	Optional	Alphanumeric	10,200	Merchant [TESTCORE23] does not exist
vpc_AcqResponseCode	Acquirer's Response Code is generated by the financial institution to indicate the status of the transaction. The results can vary between institutions so it is advisable to use the vpc_TxnResponseCode as it is consistent across all acquirers. It is only included for fault finding purposes.			
	Optional	Alphanumeric	2,3	00
vpc_ReceiptNo	This is also known as the Reference Retrieval Number (RRN), which is a unique identifier. This value is passed back to the cardholder for their records if the merchant application does not generate its own receipt number.			
	Optional	Alphanumeric	1,12	RP12345
vpc_BatchNo	A date supplied by an acquirer to indicate when this transaction will be settled. If the batch has today's date then it will be settled the next day. When the acquirer closes the batch at the end of the day, the date will roll over to the next processing day's date.			
	Optional	Alphanumeric	1,8	20021021
vpc_Authorizeld	A code issued by the acquiring bank to approve or deny the transaction. This may not always be supplied by all acquirers.			
	Optional	Alphanumeric	1,12	ABC12345

Field Name	Required Optional Input	Field Type	Length	Example Value
vpc_Card	A code issued by the Payment Server to detail the type of card the cardholder used for this transaction.			
	Optional	Alphanumeric	0,2	MC
vpc_ShopTransactionNo	The transaction reference number of the original authorisation or purchase transaction.			
	Optional	Numeric	1,19	3DS
vpc_AuthorisedAmount	The total amount of the original authorisation transaction.			
	Optional	Numeric	1,10	N
vpc_RefundedAmount	The amount of the refund transaction in the smallest currency unit expressed as an integer. For example, if the transaction amount is \$49.95 then the amount in cents is 4995.			
	Optional	Numeric	1,10	4995
vpc_TicketNo	Allows you to include a ticket number, such as an airline ticket number in the transaction request. The ticket number is stored on the Payment Server database for that transaction. The ticket number is stored on the Payment Server database for that transaction and returned in the transaction response for refunds.			
	Optional	Alphanumeric -Special characters	1,16	Any data

Table 2 Transaction Request Fields for a Refund

## QueryDR Transaction

### Transaction Request Fields – Query DR

The QueryDR command allows you to search for a transaction response, which has been lost. The search is performed on the primary key – **MerchTxnRef**, which is why the **vpc\_MerchTxnRef** field needs to be a unique value.

If there are transactions with duplicate **vpc\_MerchTxnRef** numbers, the query will only return the most recent

transaction encrypted transaction response, but a flag is raised to indicate there is more than one transaction that meets the criteria.

If the query result returned is not the correct one, you must use Merchant Administration on the Payment Server to search for the correct transaction.

The fields that are included in a transaction request when using QueryDR are:

Field Name	Required Optional Input	Field Type	Length	Example Value
vpc_Version	The version of the OByte Virtual Payment Client API being used. The current version is 1.			
	Required	Numeric	1,8	1
vpc_Command	This indicates the type of transaction.			
	Required	Alpha	1,16	QueryDR
vpc_AccessCode	The access code authenticates a merchant on the Payment Server so that a merchant cannot access another merchant's MerchantId. The access code is provided to you when you registered your merchant profile with the Payment Provider.			
	Required	Alphanumeric	8	6ab89f3
vpc_Merchant	The unique merchant ID assigned to you by your Payment Provider.			
	Required	Alphanumeric	1,16	TESTMERCH ANT01
vpc_MerchTxnRef	It is the primary key used to search the progress of a transaction in the event of a communication's failure where no transaction response is received.			
	Required	Alphanumeric -Special characters	1,40	test1234/1
vpc_User	This field is a special AMA user created to use this function.			
	Required	Alphanumeric	1,16	amauser
vpc_Password	The password used to authorise the AMA user access to this function.			
	Required	Alphanumeric	1,16	password12

Transaction Request fields for a QueryDR transaction

### Transaction Response Details – Query DR

The fields returned in the transaction response are the same as the original transaction, but includes two additional transaction response fields. The fields that are included in a transaction response when using QueryDR are:

Field Name	Required Optional Input	Field Type	Length	Example Value
vpc_DRExists	This key is used to determine if the QueryDR command returned any search results. If the value is "Y", then there is at least one vpc_MerchTxnRef number result matching the search criteria.			
	Optional	Alpha	1	Y
vpc_FoundMultipleDRs	This is used after the previous command to determine if there are multiple results. If the value is "Y", then there are multiple MerchTxnRef numbers matching the search criteria.			
	Optional	Alpha	1	N

Transaction Response fields for a QueryDR transaction

## Bypass Card Selection Page on the Payment Server

This is used in Server-Hosted Payments to bypass the Payment Server payments page that displays the logos of all the cards the payment processor will accept.

### Transaction Request Fields - Bypass Card Selection Page

The fields that are included in a transaction request when using Bypass Card Selection are:

Field Name	Required Optional Input	Field Type	Length	Example Value
vpc_Gateway	This field determines the VPC gateway that will be used. The field is case sensitive, and must comply with the gateways that are valid in the Payment Server. The value used here will always be ssl.			
	Optional	Alpha	3	ssl
vpc_Card	A code issued by the Payment Server for the card type used by the cardholder in the transaction.			
	Amex	American Express Credit Card		
	Dinersclub	Diners Club Credit Card		
	JCB	JCB Credit Card		
	Mastercard	MasterCard Credit Card		
	Visa	Visa Credit Card		
	Optional	Alphanumeric	3,16	VISA

Transaction request fields to Bypass Card Selection Page on the Payment Server

### Transaction Response Fields - Bypass Card Selection Page

The Bypass Card Selection page functionality does not return any extra fields in the transaction response.

# Troubleshooting and FAQs

## Troubleshooting

### What happens if a Transaction Response fails to come back?

To deal with a transaction response that fails to come back:

- Flag the transaction as having an error, so that it needs to be manually checked using Merchant Administration on the Payment Server. Or,
- Use the Advanced Merchant Administration (AMA), QueryDR command to search the Payment Server database for the transaction. The vpc\_MerchTxnRef is used as the transaction identifier when searching using QueryDR command.

Since the transaction response has failed to come back, there is no transaction number available from the Payment Server to identify the transaction in question, and this is why you use the **vpc\_MerchTxnRef**. It is important to have a unique **vpc\_MerchTxnRef** for every transaction otherwise the query could return multiple results. Only the most recent transaction is returned in the QueryDR command if there are multiple results, but this may not be the transaction you are concerned with.

### If the transaction response code was successful

When you find the required vpc\_MerchTxnRef in the QueryDR, check the vpc\_TxnResponseCode field to see if it is successful (should be equal to '0'). If the vpc\_TxnResponseCode is 0, then the transaction is successful and you just need to extract the relevant data details from the QueryDR results for your records.

### If the transaction response code was not successful

If the vpc\_TxnResponseCode is not 0, you need to determine the next course of action based on what you would do if the vpc\_TxnResponseCode were not 0 in a normal transaction response coming back from the Payment Server.

### If you did not find the transaction response code

If you query the Payment Server for the vpc\_MerchTxnRef using the QueryDR call and you do not receive any results, then it is safe to repeat the transaction. It is safe to use the same vpc\_MerchTxnRef, as the existing one does not show up in the Payment Server's database and was therefore never processed.

### If you find multiple transaction response code results

If the QueryDR is flagged as having multiple results (returns 'Y' in the MultipleResults field), then the MerchTxnRef is not unique. This is the primary reason for implementing a unique vpc\_MerchTxnRef for every transaction. This solution requires more data capture and processing, but it is only necessary when you don't have a unique vpc\_MerchTxnRef number. This solution requires more data capture and processing, but it is only necessary when you don't have a unique vpc\_MerchTxnRef number.

## What happens if a transaction response fails to come back

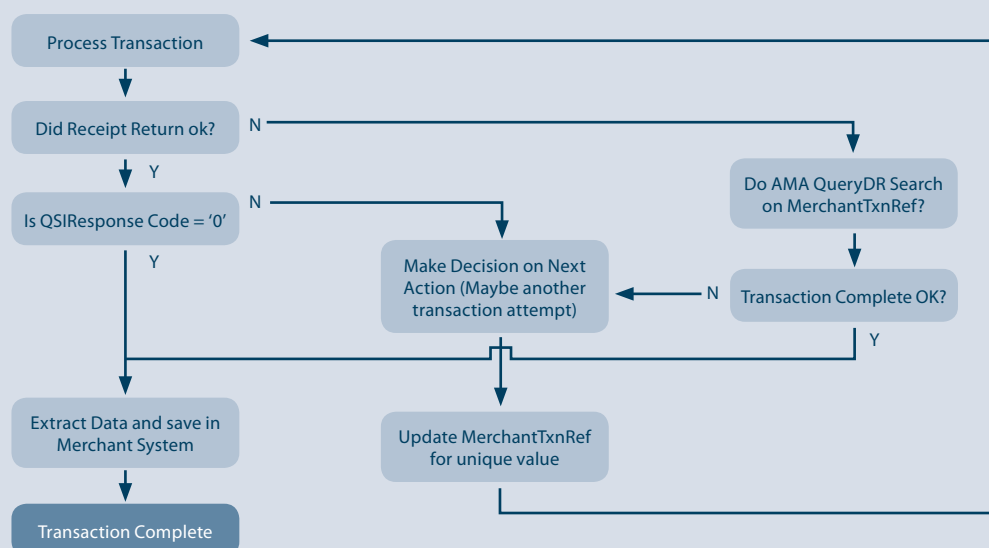


Figure 6: What happens if a transaction response fails to come back



### What to do if a Session Timeout occurs?

It is possible that while a cardholder is entering their card details at the Payment Server, the session is broken (say a communication failure due to a modem connection dropping off). If this occurs, a cardholder will lose their session. Even if they come back to your site, they will have a new session, and their old session will never be completed. To determine the status of the lost transaction, you will need to perform a QueryDR transaction based on the original vpc\_MerchTxnRef.

### Does the Cardholders Internet browser need to support cookies?

Yes. The Virtual Payment Client interface requires a cardholder's browser to support cookies for Server-Hosted Payments.

### How do I know if a transaction has been approved?

All approved transactions are represented with a response code of zero "0" from the Payment Server. All other codes represent declined transactions.

## Frequently Asked Questions

### Can the Payment Servers payment pages be modified for a Merchant?

No. The Payment Servers payment pages are branded using either the Payment Provider or Banks branding to assure cardholders of the security of the transaction. If you do not wish to display the Payment Provider's branded pages to your cardholders then you need to implement the Merchant-Hosted Payments Integration Model.

### How often can I reconcile?

Reconciliation is performed automatically by the MIGS Payment Server. It is always done around the same time each day. Your bank will be able to inform you of the cut-over time.

### Is a Shopping Cart required?

It is not necessary to have a shopping cart. All that is required is that the transaction information is within the transaction request passed to the Payment Server.

### Does the Payment Server handle large peaks in transaction volumes?

The Payment Server queues pending transactions so transactions are not lost.

### How long will an authorisation be valid on a cardholder account?

This depends on the Financial Institution who issued the card to the cardholder. Each card Issuer defines the authorisation expiry period in which they hold the funds on the cardholder's account, while they wait for the arrival of the capture transaction. Generally it is 5-8 processing days, before the authorisation purges from the cardholder account and access to the funds are released back to the cardholder.

### What is the RRN and how do I use it?

The RRN (Reference Retrieval Number) is a unique number generated by the bank for a specific bank merchantId.

It is generated by using the following formula:

The RRN is a reference used to retrieve the original transaction data and it is useful when your online store does not provide a receipt number. The RRN can be viewed in Merchant Administration.

### RRN, MerchTxnRef, OrderInfo, Authorizeld and TransactionId

**RRN (Reference Retrieval Number)** is a unique number for a particular MerchantId. This is the value that is passed back to the cardholder for their records. You cannot search for this field in Merchant Administration, but it is displayed in Merchant Administration on the transaction details pages as the Reference Retrieval Number (RRN). It is one of the fields returned in a queryDR and the transaction result (captures, refunds).

**MerchTxnRef** is generated by your online store. Ideally it should be a unique value for each transaction and you should retain this number so that transactions can be searched for in your online store and the Payment Server.

**OrderInfo** is also generated by your online store. It should also be a unique value for each transaction, which you should retain so that you can search for the transaction in your online store and the Payment Server.

**Authorizeld** is an identifier from the Acquiring Bank, which is in the transaction response for the authorisation. This field cannot be searched for in Merchant Administration, but it is displayed in Merchant Administration as the Authorisation Code. It is one of the fields returned in an AMA query and the AMA transaction result (captures, refunds).

**TransactionID** is a unique number generated by the Payment Server that matches the shopping transaction number. The shopping transaction number is the key reference value for transactions when using AMA transactional functions like captures and refunds.

## Appendix 3 – Test Environment

### Test Cards

The following table shows the test card numbers and associated expiry dates configured for each card scheme on the MIGS Payment Server.

Card Type	PAN	Expiry Date
MasterCard	5123456789012346	05/21
MasterCard	5313581000123430	05/21
Visa	4005550000000001	05/21
Visa	4557012345678902	05/21
Amex	345678901234564	05/21
Bankcard (Australian Domestic)	5610901234567899	05/21
Diners Club	30123456789019	05/21

### Response Codes

The test bank simulator is configured to allow the user to change the response received against the above test card numbers by varying the amount after the decimal point for the transaction.

The following table shows how the various response codes can be triggered varying the amount after the decimal point.

QSI Resp.	Name	Amount
0	Transaction approved	XXX.00
1	Transaction could not be processed	XXX.10
2	Transaction declined - contact issuing bank	XXX.05
3	No reply from Processing Host	XXX.68
4	Card has expired	XXX.33
5	Insufficient credit	XXX.51
6	Error Communicating with Bank	Not Mapped

QSI Resp.	Name	Caused by
7	Message Detail Error	Invalid PAN, Invalid Expiry Date
8	Transaction declined – transaction type not supported	Not Mapped
9	Bank Declined Transaction – Do Not Contact Bank	Not Mapped

For example, to obtain a response of 1 on a MasterCard, simply send a transaction for \$xxx.10 against one of the above MasterCard numbers.

Developers should use these response codes in exception handling. For further detail on the reason for decline, the issuer response code should be checked. See 'Issuer Response Code Mapping'.

### Issuer Response Code Mapping

The Payment Server returns both a summary result code generated by the Payment Server as well as the raw issuer response code as received from the bank.

Digital Receipt	Field Description
DigitalReceipt.QSIResponseCode	Summary result code as returned from the Payment Server
DigitalReceipt.AcqResponseCode	Issuer response code as returned from the bank

The following table is a list of relevant issuer response codes:

Issuer Resp.	Description
00	Approved
01	Refer to Card Issuer
02	Refer to Card Issuer
03	Invalid Merchant
04	Pick Up Card
05	Do Not Honor
07	Pick Up Card
12	Invalid Transaction
14	Invalid Card Number (No such Number)
15	No Such Issuer
33	Expired Card
34	Suspected Fraud
36	Restricted Card
39	No Credit Account
41	Card Reported Lost
43	Stolen Card
51	Insufficient Funds
54	Expired Card
57	Transaction Not Permitted
59	Suspected Fraud
62	Restricted Card
65	Exceeds withdrawal frequency limit
91	Cannot Contact Issuer

The following table shows how the bank simulator maps the issuer response code to response codes.

Issuer Resp.	MIGS Resp.	Issuer Resp.	MIGS Resp.	Issuer Resp.	MIGS Resp.
00	0	34	2	68	3
01	2	35	1	69	1
02	2	36	2	70	1
03	2	37	1	71	1
04	2	38	1	72	1
05	2	39	2	73	1
06	2	40	1	74	1
07	2	41	2	75	1
08	0	42	1	76	1
09	1	43	2	77	1
10	1	44	1	78	1
11	1	45	1	79	1
12	1	46	1	80	1
13	1	47	1	81	1
14	2	48	1	82	1
15	2	49	1	83	1
16	0	50	1	84	1
17	1	51	5	85	1
18	1	52	1	86	1
19	2	53	1	87	1
20	1	54	4	88	1
21	1	55	1	89	1
22	1	56	1	90	2
23	1	57	1	91	2
24	1	58	1	92	2
25	2	59	2	93	1
26	1	60	1	94	1
27	1	61	2	95	1
28	1	62	1	96	1
29	1	63	1	97	1
30	1	64	1	98	2
31	2	65	2	99	2
32	1	66	1		
33	4	67	1		

