st.george
Corporate & Business Bank

PayWay

API Developer's Guide

Version 1.3 6 May 2013

Document History

Date	Version	Description
26 Aug 2009	1.0	Initial Version
26 Sep 2010	1.1	New feature: registration of customers
13 Mar 2011	1.2	CVN is required for ecommerce transactions
06 May 2013	1.3	Minor Update
13 May 2015	1.4	Renamed Recurring Billing to Recurring Billing and Customer Vault
10 May 2016	1.5	Added test MasterCard starting with 2
30 Oct 2017	1.6	order.ipAddress is required for e-commerce transactions

Table of Contents

1 Introduction

St.George is utilising technology developed by our partner Qvalent in conjunction with existing market leading transactional banking products to provide comprehensive receivables management solutions.

This document describes the solution offered to meet the business needs of customers requiring on-line credit card processing through a software API in Australia. This is provided as a component of PayWay.

1.1 What is the PayWay API?

The PayWay solution allows customers to process credit cards using an Application Programming Interface (API). All communications between the customer's system and API takes place in a secure manner. The API offers the customer a range of credit card processing features including:

- Capture – debit funds from a nominated credit card
- Refund – credit funds to a nominated credit card
- Pre-authorisations – reserve funds from a nominated credit card and capture later
- Query – query the result of a previously attempted transaction
- Echo – check the status of the API service

The API supports multiple connectivity options (Internet, Leased Line) and technologies (Java, COM, ASP, .NET, PHP and SOAP) so there is a solution to suit most customers. All major credit and charge card types such as Visa, Mastercard, Bankcard, Amex, JCB and Diners[1] may be processed through PayWay. EFTPOS debit card transactions can not be processsed.[2]

The actual concept of the API is simple. The Customer System will send a Credit Card API Request to PayWay, containing either:

- A unique order number, credit card details and amount, OR
- A unique order number, recurring billing customer number[3] and amount

PayWay will send back an API Response, containing details on whether the transaction

---

[1] A separate merchant agreement is required for accepting American Express and Diners Club and Japanese Credit Bureau (JCB) charge card transactions.

[2] In order to process a EFTPOS Debit Card, the card must be physically swiped by a approved device.

[3] This mode of operation requires that your PayWay facility also have the Recurring Billing and Customer Vault module.

was successful or not.

For ad-hoc transactions it is recommended that the customer use the virtual terminal facility that is available through the PayWay customer web interface (https://payway.stgeorge.com.au).

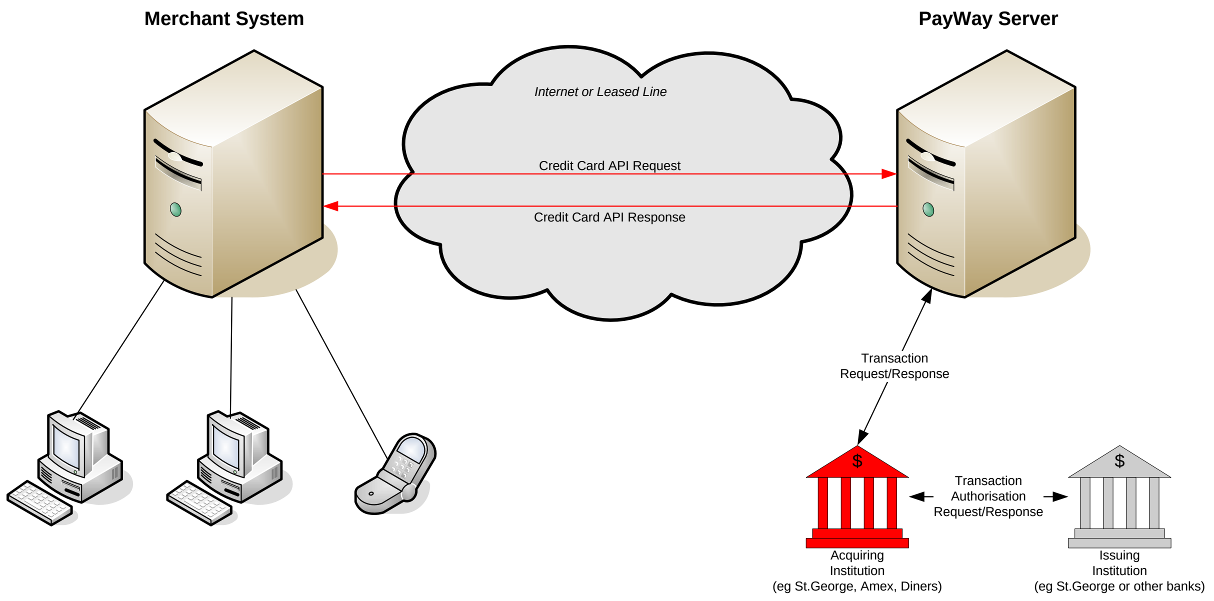
Merchant System
PayWay Server
Internet or Leased Line
Credit Card API Request
Credit Card API Response
Transaction Request/Response
$
Transaction Authorisation Request/Response
Acquiring Institution (eg St.George, Amex, Diners)
Issuing Institution (eg St.George or other banks)

Figure 1.1 – Solution Overview

1.2 Is it Secure?

For a solution of this nature, security is critical. St.George must be absolutely confident that they are receiving a credit card processing request from an authorised source. To ensure the source of the request is valid:

- The Merchant must be registered before credit card processing will begin. Part of this registration will be to issue the customer with a username and password. This username/password combination must be passed in with every request.

- PayWay will only accept requests from pre-configured IP addresses.

- For high volume customers Qvalent will recommend that a leased line (i.e. Frame Relay) be installed between the customer's site and Qvalent's data centre. If you intend to perform more than 100,000 transactions per month through the Credit Card API, please discuss this option with your St.George implementation manager.

- For added security a card verification number (CVN) can be supplied with the credit card API call.

- All transaction data will be communicated via HTTPS with 128-bit encryption and each message digitally signed (MAC). A digital certificate is provided to each customer for this purpose.

- Using the API in conjunction with the PayWay Recurring Billing and Customer Vault module means that the merchant does not need to store or handle credit card details in their system.

The connectivity options are as follows:

- Leased line. Dedicated link between the customer's site and St.George's credit card server. In this scenario, no data would be transmitted over the Internet. Username / password and digital certificate are required.

- Internet. Credit card transactions are passed over the Internet. Username / password and digital certificate are required.

In both cases above, Qvalent will provide you with a digital certificate (via download), username and password. This certificate will be in the form of a file that you will deposit into a directory on your server. Qvalent also will ask you to set up the IP addresses which are allowed to send transactions for your company.

All attempts to use the credit card service are logged. This includes Internet connections, API calls and the success or failure of those calls. Logs will be written to permanent storage. Logging will include the IP address of the caller and all the information sent and received to that destination.

Credit card details are protected as follows:

- Credit card numbers and expiry dates are stored in encrypted form in the database. Credit card numbers are recorded in truncated form in system logs (ie first few and last few digits only)

- Card verification number (CVN) is not stored in the database or in system logs. The value is passed on as is to the banking network and only a record of whether a value was provided or not is retained.

Finally, refunds are restricted to only be allowed against previous captures so there is no risk of fraudulent activity within your business.

1.3 Getting Started

First of all … **DON'T PANIC!**

This document has lots of pages and sections. We have tried to include as much detail as possible for customers wishing to implement the PayWay Credit Card API solution so you have all the information you need at your fingertips. Due to its contents, it can be quite daunting at first read, however not all sections are relevant to all customers.

We recommend that the document be examined by your IT staff with a quick skim through at first and then start working through the steps documented in Section 1 in order to implement the solution. If you have any questions that cannot be answered from the document, please direct these to your St.George implementation contacts.

Section	Why You Should Read It
2	This section provides a step-by-step guide to implementing the CC API. This is where all customers should start.
3	This section is where most of the technical specification of the CC API is included along with tips on how to develop your client software. There are also details on our web-based screens for administration and reconciliation processes. As such, this section is used as a reference by all customers.
Appendix A	This contains a list of common response codes in Australia.
Appendix B	This contains a glossary of terms used in the document.

2 Implementing the Credit Card API

This section details the process of implementing the PayWay Credit Card API for your business. This includes how you will be assisted in the implementation by St.George, and by our subsidiary Qvalent, from the initial testing through to production usage.

2.1 Setup

Follow these steps:

1. Sign-in to the PayWay website using the login name and password provided to you by your implementation manager.

2. Download the API software for your technology from the PayWay web site. The supported technologies are Java, Microsoft .NET, Microsoft COM/ASP, PHP and SOAP. Click on "Setup API" in the menu and then "Downloads".

3. Extract the software bundle (which is a ZIP file) to a folder, and review the "PayWay API for …" PDF document in that folder. Follow the instructions in that document to install the API software.

4. In necessary, configure your firewall to allow outbound SSL traffic (port 443) and determine if you need to use a proxy server to communicate with the PayWay server.

5. Get your API username and password from the PayWay web site and record these details for use in your application. This is information listed on the "Security" page under the "Setup API" heading. Note that this username and password is for your application to connect with the PayWay API server and is different from the username and password that you use to access the PayWay web site.

6. Download your certificate from the PayWay web site on the "Certificate" page under the "Setup API" heading. You must first select the technology you are using and press the "Go" button, since different certificate formats are required depending on your application technology. Record the location of the certificate file for use in the initialisation of the PayWayAPI object.

7. Develop your API solution by integrating the API into your application (section 3 provides more detail on this).

8. To determine your IP address, perform a test transaction with the correct username, password and certificate. You will receive a QU response code, and the response description will tell you your IP address. Add this IP address to the security access list on the "Security" page under the "Setup API" heading.

Any questions or issues regarding development should be emailed to PayWay Customer Care (payway@stgeorge.com.au). Please include your name, client number, description of the issue and a copy of the log file in the email. The log file can be found in the *logDirectory* you selected (see section 3.1.3). A member of our support team will assist you as soon as possible. **Never send credit card numbers in an email.**

2.2 Testing

For any customer wishing to implement our credit card API, we provide a test merchant which simulates responses rather than sending the transaction to the live banking network. This test merchant is accessed through the normal production URL, but the customer merchant specified is "TEST".

When using the test merchant, only the card numbers in Table 2.1 are valid. All other card numbers will return a response of "42 No Universal Account". Each card number will return a specific response as detailed in Table 2.1, so if you want to test a card which has low funds, you would use card number 4564710000000020 with an amount higher than $10. Note that if you enter an incorrect expiry date for one of the test cards, you will get a response of 54. If you enter an incorrect CVN, you will get a response of 01 or 05 depending on the card type.

The test merchant simulates a live gateway but may be used without any risk of transactions actually being processed through the banking system.

Card Number	Expiry	CVN	Response	Description
4564710000000004	02/19	847	08	Visa Approved
5163200000000008	08/20	070	08	MC Approved
2221000000000009	01/20	009	08	MC Approved
4564710000000012	02/05	963	54	Visa Expired
4564710000000020	05/20	234	51	Visa Low Funds ($10 credit limit)
5163200000000016	12/19	728	04	MC Stolen
4564720000000037	09/19	030	05	Visa invalid CVV2
376000000000006	06/20	2349	08	Amex
343400000000016	01/19	9023	62	Amex Restricted
36430000000007	06/22	348	08	Diners
36430000000015	08/21	988	43	Diners Stolen
5163200000000024	02/19	847	If Fraud Guard is active 34 otherwise 08	Fraud Guard
5163200000000032	02/19	847	If Fraud Guard is active 34 otherwise 05	Fraud Guard

Table 2.1 – Allowed test card numbers

Refer to the technology specific API documentation provided on how to carry out a test transaction.

2.2.1 Viewing Test Transactions

You can view test transactions in PayWay as follows:-

1. Sign-in to PayWay
2. Click **Search and Refund** under the **Transactions** menu item
3. Select the **Settlement Date**[4] of the transaction and **Test – Test Merchant**
4. Click **Search**

2.2.2 Recurring Billing and Customer Vault

If you wish to use the credit card API in conjunction with the recurring billing and customer vault module, you will need to setup a test customer as follows:

1. Sign-in to PayWay
2. Click "Add Customer"
3. Choose a customer number (e.g. "TESTCUSTOMER1") and enter a name
4. Select **Variable Debit**
5. Click **Next**
6. Choose **Credit Card**
7. Click **Next**
8. Enter one of the test card numbers listed above.
9. Click **Next**
10. Confirm the details and click **Save**

You can now call the Credit Card API and pass in the customer number you selected instead of card details (see Using Recurring Billing and Customer Vault to hold Credit Card Details, page 24). In your API request specify:

```
customer.mercant=TEST&customer.customerReferenceNumber=TESTCUSTOMER1
```

If you wish to disable the test customer, you can do this as follows:-

1. Sign in to PayWay
2. Click **Search and Edit** in the menu
3. Enter the customer number of the test customer (e.g. TESTCUSTOMER1)
4. Click **Search**
5. Click **Edit Customer**
6. Choose **Recurring Billing Setup** and click **Go**
7. Click **Stop Remaining Payments**
8. Confirm that you wish to **Stop Remaining Payments**

The customer is now stopped and can not be charged through the API.

2.3 Production

2.3.1 Going Live

Once you have developed and tested your API solution, you are ready to go live. Before this can happen, your implementation manager must set up billing and provide you with

[4] Transactions processed after 6pm Sydney time settle on the following day.

a live merchant id. When these two steps are completed, you must press the "Go Live" button on the "Go Live" page under the "Setup API" heading of the PayWay web site.

Once you press this button, you may perform live transactions against your live merchant id. However, you may still use your TEST merchant id to perform test transactions. Transactions to your TEST merchant will not appear on the cardholder statement or your bank account.

We recommend that you process one live transaction before you start sending all your transactions to the new system. The day after you perform this test transaction, you should check your settlement account to make sure that the funds from this transaction arrived in your account. Once this verification is done, you can then switch all your live transactions to the new system.

2.3.2 Support

- For issues relating to your Merchant agreement with St.George, contact Business Direct on **1300 781 605**.

- For issues relating to your Merchant agreement with American Express, contact Amex on **1300 363 614**.

- For issues relating to your Merchant agreement with Diners Club, contact Diners on **1300 360 060**.

- For issues relating to the Payway website or Credit Card API, contact your Implementation Manager or Payway Customer Care by phone on **1300 395 501** (available Monday to Friday, 8:30 am. to 5:30 pm AEST)

- For issues relating to Credit Card API development, email Payway Customer Care (payway@stgeorge.com.au). Please include your name, client number, description of the issue and a copy of the log file in the email. The log file can be found in the *logDirectory* you selected (see section 3.1.3).

Never send credit card numbers in an email.

3 Application Programming Interface

The PayWay API provides a real-time interface to execute credit card transactions. This interface uses the familiar concept of a remote procedure call (RPC). The customer invokes the remote method with the credit card details then the PayWay server processes the transaction and returns the result in real-time.

This transaction contains the following steps:

1. Customer calls the API initialise method. This loads the URL, loads the certificate and initialises SSL.

2. The Customer uses the 'processCreditCard' client command to submit a credit card transaction. The API digitally 'signs' the request to allow the PayWay server to verify the request originator and sends the request to the PayWay server.

3. The Customer waits for a response to be returned through the API connection.

4. The PayWay server verifies the digital 'signature', merchant username / password and IP address of the originating request. If all these are valid, the server verifies the other credit card request parameters.

5. The PayWay server contacts the issuing bank and requests authorisation for the credit card transaction.

6. The PayWay server returns the Response to the Customer through the HTTPS connection established in step 1.

7. The API reads the Response and returns it to the process that initiated the API request.

Depending on the technology used, this process is then can the repeated from step 2 for further Request/Response cycles to process additional credit cards. Refer to your technology specific API document for more details on whether the API object can be re-used for additional transactions.

3.1 Qvalent Provided Software

3.1.1 Supported Technologies

Qvalent provides an API software package for the following technologies:

- Java
- Microsoft Component Object Model (COM) and Active Server Pages (ASP)
- Microsoft .NET
- PHP

This software API provides an object in the relevant programming language which performs all the communication with the St.George PayWay server. The objects contain the same methods/functions regardless of the technology, and these are listed below. The method names start with lower case for Java and PHP, and upper case for Microsoft technologies, according to the relevant conventions

Method Name: initialise

Description: Initialise this object so that it can process CCAPI transactions. Configuration information is read from the provided parameters string.

Arguments:

> `parameters` - the initialisation parameters (delimited with an ampersand (&)) to use. These parameters must contain at a minimum the log directory and the certificate file. See section 3.1.3 for the available parameters.

Return Value: None.

Method Name: isInitialised

Description: Returns true if this client object has been correctly initialised or false otherwise.

Arguments:

> None.

Return Value: True if this client object has been correctly initialised or false otherwise.

Method Name: processCreditCard

Description: Main credit card processing method. Pass the request parameters into this method, then the current thread will wait for the response to be returned from the server.

Arguments:

> `parameters` - The parameters string containing all the request parameters (delimited with an ampersand (&)) to send to the server. See section 3.2.1 for the available parameters.

Return Value: The response string containing all the response parameters (delimited with an ampersand (&)) from the server. See section 3.2.2 for details on the returned parameters.

Table 3.1 – PayWay API Method Details

3.1.2 Other Technologies

If your application does not use one of the supported technologies, there is another option available – SOAP. Using this integration method, you develop your application to send the credit card requests to Qvalent using SOAP. More details about how to do this are provided in the document entitled "PayWay API for SOAP" which can be downloaded from the PayWay web site.

The parameters are still sent and received in the same format as for all other technologies (i.e. one long string containing all request parameters, delimited by an ampersand (&)). Section 3.1.3 on initialising the API will not be relevant to you, but all other sections of this document still apply.

The SOAP method to process a credit card is called "processCreditCard" and has the same signature and behaviour as described in Table 3.1.

3.1.3 Initialising the API

The Qvalent PayWay API object must always be initialised before it can be used. The object instance should be created in the normal way for your technology, and you must then call the "initialise" method on this object. The parameters for the "initialise" method are listed in the table below.

Name	Description	Required
logDirectory	The directory to place the log files in. If the directory does not exist, it will be created.	Yes
certificateFile	The location of the file containing your certificate. Use a fully qualified file name including the drive letter and full directory information.	Yes
proxyHost	The host name of the proxy server that your application must connect through to access the Qvalent PayWay server. Do not include this parameter if you do not require a proxy server.	No
proxyPort	The port number the proxy server listens on. Do not include this parameter if you do not require a proxy server.	No
proxyUser	The username required to connect to the proxy server. Do not include this parameter if you do not require a proxy server.	No
proxyPassword	The password required to connect to the proxy server. Do not include this parameter if you do not require a proxy server.	No

Table 3.2 – The parameters used by the "initialise" method

Where you store this information in your application is up to you. The best solution is to make this information of your application's standard configuration. For example, if your configuration is stored in an XML file, add this information to that XML file and read it from your application. It is unwise to "hard-code" values for these parameters in your software, since you may wish to change them in the future.

An example parameters string for the initialise method is

```
certificateFile=d:\payway\payway.q0&logDirectory=d:\payway
```

Note that there should be no spaces or line breaks in the parameters string.

Any errors during initialisation will be reported through your technology's usual mechanisms (e.g. the Java and .NET objects will throw an exception, the PHP object will raise an error). These errors normally relate to invalid parameters being specified (such as a non-existent certificate file), or invalid system configuration (such as missing required libraries). By default, a 60 second timeout will apply for the initialisation.

Once the object has been initialised, you may call the “processCreditCard” method to process credit card transactions. It is possible to call “processCreditCard” multiple times on the same object from multiple threads depending on your technology. Refer to your technology specific API document for more details.

3.2 Processing Credit Card Transactions

To process a credit card transaction, your application sends the PayWay server a string of request parameters, and it returns you a string of response parameters. Both these strings are of the same format:

- Each parameter is of the form: *name=value*
- Parameters are separated by ampersand characters (&)

The same format applies regardless of whether you are using Qvalent PayWay API software or the SOAP interface.

Only use standard ASCII characters in your parameter values. **Do not** use any of the following special characters in your parameter values:

- Ampersand (&) - ASCII character number 38
- Plus sign (+) – ASCII character number 43
- Percentage sign (%) – ASCII character number 37

3.2.1 Request Parameters

Below are the valid request parameters that your application can provide.

Parameter Name	Data Type	Description	Required
order.type	See Description for valid order types	The type of processing required: • capture • refund • query • echo • preauth • captureWithoutAuth	Yes
customer.username	At most 32 chars	The customer's username (from PayWay web site). The PayWay server uses the username parameter to identify your company within the system. You will always use the same username and password regardless of which merchant ID you select.	Yes
customer.password	At most 32 chars	The customer's password (from PayWay web site).	Yes
customer.merchant	At most 32 chars	The merchant ID to use. Enter either 'TEST' for test transactions, or your live St.George merchant ID. Do not use your Amex or Diners merchant ID here.	Yes
card.PAN	At most 19 chars	The credit card number.	Only for capture / refund / preauth / captureWithoutAuth (Alternatively, provide customer.customerReferenceNumber)
card.CVN	At most 4 digits	The card verification number. This is a 3 or 4 digit code that provides extra security for online payments. For Visa, MasterCard, BankCard and Diners, this is the last 3 digits on the back of the signature panel. For Amex, this is the 4 digit number on the front of the card above the embossed card number. Under no circumstances should the CVN be stored in any system.	Optional for capture / refund / preauth / captureWithoutAuth
card.expiryYear	2 digits	The year the card expires.	Required for capture / refund / preauth / captureWithoutAuth (Alternatively, provide customer.customerReferenceNumber)

Parameter Name	Data Type	Description	Required
card.expiryMonth	2 digits	The month the card expires – must be between 1 and 12.	Required for capture / refund / preauth / captureWithoutAuth (Alternatively, provide customer.customerReferenceNumber)
card.cardHolderName	At most 60 chars	The name on the credit card being processed. Note this is for information only – the card holder name will not be checked by the card issuer. This field will appear on the reports that can be downloaded from the PayWay website. Note: Do not use any of the following characters in the card holder name: & % +	Optional (Alternatively, provide customer.customerReferenceNumber)
order.amount	At most 12 digits	The amount to apply to the card based on the order.type in cents. This amount should be a positive value for all order types. For example, enter "1295" for an amount of "$12.95".	Required for capture / refund / preauth / captureWithoutAuth
customer.customerReferenceNumber	At most 20 chars	The customer reference number to record against this transaction. This field will appear on the reports that can be downloaded from the PayWay website. Only letters, numbers, dashes, underscores and full-stops are accepted in this field. This field can be provided as an alternative to card.cardHolderName, card.expiryYear, card.expiryMonth and card.PAN if the customer reference number is registered in the PayWay application and has associated credit card details. Customers can be registered by you by choosing "Add" under the "Customers" menu item. Alternatively, you can have them self-register on a St.George hosted web page (See "Internet Sign Up for Recurring Billing" in the PayWay User Guide, which is available from the Downloads section of the PayWay website). A list of registered customers can be found by signing into PayWay and selecting "Search and Edit" under the "Customers" menu item.	May be provided for capture / refund / preauth / captureWithoutAuth as an alternative to card.PAN, card.expiryYear and card.expiryMonth

•

Parameter Name	**Data Type**	**Description**	**Required**
customer.orderNumber	At most 20 chars	Your unique transaction number. This number can be used at a later date to query a transaction. This is your own internal tracking number for this transaction, which can also be used in the PayWay search screens. This number must be used by your systems to prevent duplicate transaction processing. Note 1: This must be unique for a given merchant id. Note 2: Do not use any of the following characters in your order number: & % + Note 3: Include at least one letter or use a maximum of 15 numeric characters to ensure the order number is displayed correctly in Microsoft Excel.	Required for capture / refund / query / preauth / captureWithoutAuth
card.currency	3 chars	The currency to apply to this card transaction. Always enter 'AUD' for Australian Dollar Payments. The PayWay API only accepts transactions in Australian Dollars. However, you can charge credit cards from outside Australia. You will receive funds in Australian Dollars. The transaction will appear on the cardholder statement in their currency using an exchange rate determined by the card scheme (Visa, Bankcard, American Express, etc).	Required for capture / preauth / captureWithoutAuth Optional for refund
order.ECI	3 chars	The Electronic Commerce Indicator (ECI) to apply to this card transaction. Please refer to Section 3.3.1.1 for more details on this parameter.	Required for capture / refund / preauth / captureWithoutAuth
customer.originalOrderNumber	At most 20 chars	This parameter is used when performing a Refund to link the refund with the original Capture. See Section 0. Also used when capturing a previous preauth. In this case, this parameter must be populated with the original order number of the preauth transaction. See section 3.3.3. Note that this field was previously named customer.captureOrderNumber	Required for refund / captureWithoutAuth
order.authId	6 chars	The authorisation Id returned from the corresponding preauth transaction. See section 3.3.3. This should be the value returned in response.authId for the preauth transaction.	Optional for captureWithoutAuth Must not be present for other transaction types.
order.includedSurchargeAmount	At most 12 digits	The amount of the order total that is considered to be the surcharge in cents (for example, enter "129" for a surcharge amount of "$1.29"). Only specify this field if you apply surcharges to credit card payments. This field defaults to 0 if not specified. This amount must be less than the order total. This field will appear on the reports that can be downloaded from the PayWay website.	Optional

Parameter Name	**Data Type**	**Description**	**Required**
order.ipAddress	At most 15 chars	The IP address of the card holder performing the transaction (if the transaction is an internet payment). This field must not be populated for mail, telephone or recurring payments. This information is used in fraud checking if you have opted for the Fraud Guard module. E.g. 10.101.101.101	Required for e-commerce transactions.

3.2.2 Response Parameters

Below are the possible response parameters that the PayWay server can send back to your application. Note that St.George may add additional parameters to the response at any time, so you should not assume that these will be the only parameters present in the future.

Parameter Name	Data Type	Description	Always Present
response.summaryCode	number	The summary code in numeric format: 0 – Approved 1 – Declined 2 – Erred 3 – Rejected	Yes
response.responseCode	2 chars	The AS2805 response code of the transaction. See Appendix A	Yes
response.text	String (At most 200 characters)	A textual description of the transactions response or failure.	Yes
response.settlementDate	yyyymmdd	The settlement date for the transaction. See Section 3.3.2.2.	Returned when the transaction is stored in the PayWay database. Returned for all approved transactions and most declined transactions.
response.receiptNo	String (At most 32 characters)	Internal PayWay reference number for transaction.	Returned for all approved transactions and most declined transactions.
response.cardSchemeName	String (At most 20 characters)	This is the card scheme of the card used in the transaction (see section 3.3.2.3 for more details). Values returned are: ➢ AMEX ➢ BANKCARD ➢ DINERS ➢ MASTERCARD ➢ VISA	Only if card scheme is known

Parameter Name	Data Type	Description	Always Present
response.creditGroup	String (At most 20 characters)	This is the card scheme grouping of the card used in the transaction (see section 3.3.2.3 for more details). Values returned are: ➢ AMEX ➢ DINERS ➢ VI/BC/MC	Only if card scheme is known
response.transactionDate	dd-mmm-yyyy hh24:mi:ss	The date/time the transaction occurred. Refer to section 3.3.2.1.	Only if transaction passes initial parameter checks.
response.authId	String (At most 6 characters)	The authorisation Id returned from the issuing bank for preauth transactions. This value must be passed in with any subsequent captureWithoutAuth transaction for the same card.	Only for approved preauth transactions

3.2.3 API Examples

3.2.3.1 Request Parameters Example

```
customer.username=Q00000&
customer.password=Ahl2jfi8n&
customer.merchant=TEST&
order.type=capture&
card.PAN=4564710000000004&
card.CVN=847&
card.expiryYear=19&
card.expiryMonth=02&
order.amount=1000&
customer.orderNumber=1136346832577&
card.currency=AUD&
order.ECI=SSL
```

The above parameters string has been split at the ampersand (&) markers for readability. When you construct your parameters string, you should not include any spaces or line breaks.

3.2.3.2 Response Parameters Example

```
response.summaryCode=0&
response.responseCode=08&
response.text=Honour with identification&
response.receiptNo=505228832&
response.settlementDate=20060125&
response.transactionDate=25-JAN-2006 14:09:49&
response.cardSchemeName=VISA&
response.creditGroup=VI/BC/MC
```

The above parameters string has split at the ampersand & markers for readability. The response parameters you receive will not contain these extra line breaks.

3.3 Further Details

3.3.1 API Request

3.3.1.1 Using Recurring Billing and Customer Vault to hold Credit Card Details

If you have both the PayWay API and Recurring Billing and Customer Vault modules, you can:

- register a customer number and associated credit card details via a web page or via the API (see section 3.5),
- charge the customer by passing the customer number and amount using the Application Programmer Interface (API).

Your software has full control over transaction processing without needing to store credit card details. This solution can potentially reduce your obligations under Payment Card Industry Data Security Standards (PCI DSS) compliance.

When the card-holder registers his/her credit card details using the Recurring Billing and Customer Vault module, the details are recorded against a customer reference number that you supply. The customer reference number must be unique – each card-holder may only register one credit card.

Customers can be registered by you. Sign in to PayWay and click "Add" under the "Customers" menu item. Alternatively, you can have them self-register on a St.George hosted web page (See "Internet Sign Up for Recurring Billing" in the PayWay User Guide, which is available from the Downloads section of the PayWay website).

Alternatively, customers can be registered via the API. See section 3.5 for more details.

A list of registered customers can be found by signing into PayWay and selecting "Search and Edit" under the "Customers" menu item.

To use the pre-registered cardholder information, use the following parameters:

- order.type – capture, preauth, refund, captureWithoutAuth (cardholder details are not required for other order types)
- card.PAN, card.expiryYear, card.expiryMonth, card.CVN – these parameters must not be present, otherwise the pre-registered account will not be used
- customer.customerReferenceNumber – the unique reference number for the pre-registered customer
- customer.merchant – optional since the customer is registered against a merchant
- other parameters as normal

If the card-holder's details cannot be located, a QA Invalid Parameters response will be returned. Otherwise, the response you receive will be the same as if you had specified the card details yourself.

3.3.1.2 Electronic Commerce Indicator

The Electronic Commerce Indicator (ECI) is used by acquirers/issuers to determine the type of transaction being processed. The ECI value should represent the *source* of the transaction request. That is, the environment that the cardholder used to provide the payment card details to the merchant. It is important that merchants set the correct ECI value during transaction processing to ensure that appropriate merchant service rates are received.

An Electronic Commerce Transaction is one where the cardholder enters their card details into a merchant's website and the transaction is processed immediately. According to Card Scheme rules, merchants must collect the Card Verification Number (CVN) for all electronic commerce transactions. This means that if you provide an ECI value from the table below that represents an Electronic Commerce Transaction, **you must also provide the CVN in the card.CVN parameter**.

Note: PCI DSS requires that the CVN must never be stored in any card processing system.

ECI Value	Description	Electronic Commerce Transaction
CCT	Call Centre Transaction	No
IVR	IVR Transaction	No
MTO	MOTO Transaction	No
SSL	Channel Encrypted Transaction (SSL or other)	Yes
REC	Recurring payment[1]	No
5	3D Secure transaction. This is the value returned from your MPI (Merchant Plugin Interface) software for 3D Secure transactions[2]	Yes
6		Yes
7		Yes

Table 3.3 – Possible ECI values

[1] The REC ECI value is only applicable to Visa and MasterCard transactions that satisfy the Visa Recurring and MasterCard Recurring scheme rules.

[2] The numeric ECI values are only available when the transaction contains the 3D Secure order.xid and order.cavv fields. Depending on your 3D Secure MPI software, you may need to convert the MasterCard ECI values to the Visa ECI values depicted above.

3.3.1.3 Refund Orders

Refunds are always matched against an original capture using the originalOrderNumber parameter. The original order number will be checked to ensure that it was an approved Capture. Finally, the refund will be checked to determine if the amount is less than or equal to the original capture less other approved refunds against the same capture. If all of these checks are passed, the transaction will be processed as per normal. If any of the checks fail, a 'QV' response will be returned with response text differing based on which condition failed.

Although all parameters should be supplied as per normal, the following includes an explanation of the key API parameters required for a refund against an original capture:

- order.type=refund
- card.PAN, card.expiryYear, card.expiryMonth – if supplied they must match the details given on the original capture. If they are not supplied then the values given in the original capture will be reused.
- order.amount - amount being refunded, must not exceed amount originally captured
- customer.orderNumber - new, unique transaction reference for the refund (i.e. a different value from the order number used for the original capture)
- customer.originalOrderNumber – order number of original capture

3.3.1.4 Query Orders

In order to execute a 'query', the request must include:

- order.type=query
- your identification details (customer.username, customer.password & customer.merchant)
- customer.orderNumber of order you are querying

For most transactions, the response will include the previously returned response code. However, if this was previously a summary code of 2 (ie Transaction Erred), the API will attempt to determine an updated response code for the transaction if available. Since order numbers are only unique per merchant, make sure you are using the same merchant ID as for the original transaction.

If you perform a query and receive a Q2 response, the original transaction is still being processed. You should re-attempt the query after 60 seconds.

When executing a Query order, it is important to note that it is possible that the response code indicates a failure with the query itself rather than the original transaction itself. For example, the original transaction may have been successful but the subsequent query fails due to a network issue. If this has occurred, a QI is returned for the query.

One special case for queries is that a QG is returned if the order number supplied is unknown. This may occur if your system has attempted a transaction that failed prior to being attempted (eg network issues between your server and PayWay). In this case, the original transaction was not attempted, and can be safely retried if required.

3.3.1.5 Echo Orders

The Echo order type allows your application to interrogate the status of the API server. If you include order.type=echo, a response code of '00' will be returned if the PayWay server is accessible. This is essentially a network test from your server to PayWay. However, this does not confirm that your merchant account is available.

3.3.1.6 Preventing Duplicate Payments

When implementing the API, it is <u>your responsibility</u> to ensure that your application prevents customers from performing duplicate payments. However, this section is provided as a guide of best practices to avoid duplicate payments.

Firstly, the most common causes of customers performing duplicate payments via ecommerce applications are:

- Customer double-clicks on Make Payment button on web application and the payment is executed twice.
- Customer does not wait sufficient time for the transaction response to be returned and they retry the transaction.
- Due to a system or network issue, the customer does not receive a response from your system and they retry the transaction.

In each case, safeguards can be put in place which virtually eliminate duplicate payments.

With the API, it is important to remember that the customer.orderNumber must be unique for each distinct transaction attempt. This means that if a transaction is retried with the identical customer.orderNumber, it will only be processed once (any further attempts will be rejected with a Q6 Duplicate Transaction response). However, if transactions are submitted with the same details except with a different customer.orderNumber, they will both be processed.

The following recommendations should be considered when designing your application to prevent duplicate payments:

- If providing a web or screen interface, ensure that the Make Payment button can only be clicked once and double-clicking will not attempt two transactions.

- Prior to executing an API transaction, commit this to permanent storage in your system in a pending state. When the response is returned, update the details with the response. If there is a network failure mid-transaction or your application crashes, you may then later use the Query operation to determine the success or failure of the first attempt.

- Once a customer has confirmed that they wish to pay, check if you have any approved (summary code 0), pending or incomplete (summary code 2) transactions for the same customer number, credit card number[5] and amount within a recent time period (eg 24 hours). If you do find a match, inform the customer of the situation (eg "Possible duplicate payment attempt. Please check your statement and confirm whether transaction has been processed before retrying.") but optionally give them the opportunity to proceed with executing the payment if they wish.

3.3.2 API Response

3.3.2.1 System Times

All times are based on Sydney time. This is either Australian Eastern Standard Time (AEST) or Australian East Daylight Time (AEDT) depending on the time of year.

3.3.2.2 Settlement Date

The settlement date represents the date that the transaction will be settled by the acquiring bank. The settlement date cut-off is always 6pm (refer to section 3.3.2.1). If you process transactions after this time, the settlement date will be the following day. For example, if you process a transaction at 7pm on 24 Jan 2006, the settlement date will be 25 Jan 2006 (represented as "20060125" in the response).

Furthermore, the settlement date does not necessarily mean that the funds will be

[5] For customers that do not wish to store credit card details in their system, we recommend that a one-way hash algorithm instead be used for check if the same card is being used.

credited on the settlement date returned. Again, this is up to the relevant acquiring bank (see 3.3.2.2 for a list). However, for reconciliation purposes, all successful transactions through a given acquirer that return the same settlement date will be credited together on the same day.

St.George credits your account the same day, except on weekends and public holidays when the settlement is delayed until the next banking day. The amount credited is the total of approved transactions. Any merchant service fees will be deducted as separate transactions per your service agreement.

American Express and Diners Club may credit your account a number of days later and may be a net amount (ie approved transactions less the merchant service fees), depending on your contract with them. Although St.George facilitates the processing of the transaction, we do not control settlement for these schemes and therefore any queries should be made to American Express and Diners Club directly.

The settlement date will be returned for all approved transactions. It will often be returned for declined transactions but not all declined transactions. It will be returned only if the transaction is stored in the PayWay database, but not transactions rejected prior to this (for example: QJ - Incorrect Customer Password). For customers receiving a transaction log, the settlement date may be used to reconcile the log with your own records.

3.3.2.3 Card Scheme vs Credit Group

In order to aid reconciliation, the API response includes details on the cardScheme and creditGroup. The differences between these two parameters are explained below.

Every transaction will be using a card from a particular card scheme, such as Amex or Visa. However, the creditGroup indicates which transactions will be grouped together in a single credit to the customer’s bank account. This occurs for Visa, Mastercard and Bankcard transactions where St.George will group all such transactions into a single credit. However, Amex and Diners will be credited separately.

The following table summarises the relationship:

cardScheme	creditGroup	Acquiring Bank
AMEX	AMEX	American Express
DINERS	DINERS	Diners Club
MASTERCARD	VI/BC/MC	St.George
VISA	VI/BC/MC	St.George

Table 3.4 – Card scheme, credit group and acquiring bank relationships

3.3.3 Pre-Authorisations

Most of the transactions you perform using the Credit Card API will be “capture” transactions. These transactions are equivalent to a purchase using an EFTPOS device (i.e. someone in a shop purchasing something and paying by credit card).

Behind the scenes, a capture is made up of two parts: a pre-authorisation, and a clearance. The pre-authorisation is the message that is sent to the issuing bank to check whether the transaction can be processed (i.e. the card number is correct, and the account has enough funds etc). Once the pre-authorisation transaction is approved, the clearance is sent and at the end of the day, the transaction will appear on the card holder's statement. The pre-authorisation can be thought of as "reserving" the funds, and the clearance can be thought of as a trigger message that causes the actual transaction to take place.

Capture transactions consist of these two parts, and they happen seamlessly to the API user. However, the API allows customers to split the process into its two separate components when this is required by a customer's business model. This is the purpose of the "preauth" and "captureWithoutAuth" transactions. The "preauth" transaction reserves the funds, and the "captureWithoutAuth" transaction adds the transaction to the list of clearances that will occur at the end of the day.

An example usage would be if a company had separate order processing and shipping systems. The order processing system could first check the available funds using the "preauth" transaction, and then send the order to the shipping system. When the goods are ready to be shipped, the order processing system would send the "captureWithoutAuth" transaction to complete the purchase. Separating these two functions allows for the case where the goods are out of stock, or discontinued, since no money has been taken from the card holder's account.

An important consideration is the length of time that a pre-authorisation will last. This time depends solely on the issuing bank and cannot be controlled by St.George. Most issuers will keep the funds reserved for at least three banking days. You should certainly not assume that you can safely perform a "preauth" then perform the "captureWithoutAuth" 4 weeks later.

This short lifetime of pre-authorisations is the main reason that most companies do not use this feature. Most companies interested in pre-authorisations have a much longer time between ordering and shipping (such as several weeks). The danger in sending the "captureWithoutAuth" after the "preauth" has expired is that the issuing bank will raise a charge back for the transaction.

Each pre-authorisation is given an identifier by the issuing bank so that the capture can be matched to the reserved funds. To ensure that this matching occurs, you must specify the order number of the "preauth" transaction in the "customer.originalOrderNumber" parameter for the "captureWithoutAuth" transaction request. You must generate a new order number for the "captureWithoutAuth" transaction.

3.3.3.1 Performing Preauth Transactions

To perform a "**preauth**" transaction, use the following request parameters:

- order.type = preauth
- customer.orderNumber - unique transaction reference for the pre-authorisation
- All other fields as normal (see Section 3.2.1).

To perform a "**captureWithoutAuth**" transaction, use the following request parameters:

- order.type = captureWithoutAuth
- card.PAN, card.expiryYear, card.expiryMonth – not present. You are not required to store card details.
- customer.orderNumber - new, unique transaction reference for the "captureWithoutAuth" (i.e. a different value from the order number used for the original pre-authorisation)
- order.authId – not present
- order.amount – the amount to charge the card-holder
- customer.originalOrderNumber – order number of original preauth transaction

Remember:

1. You must perform a "captureWithoutAuth" transaction to complete the purchase. If you do not, the funds will not appear in your account.
2. You cannot perform a "captureWithoutAuth" without a corresponding "preauth".
3. Pre-authorisations do not last for very long (typically around 3 days).
4. The "preauth" and "captureWithoutAuth" transactions must have unique order numbers in the "customer.orderNumber" parameter.

3.4 Error Handling

The API will inform your system of any errors that occurred in the processing of the credit card request. Error information will be returned in summary form in the response.summaryCode and a detailed description of the error in the response.responseCode and response.text. As in most software, much of the code you write will be to handle error conditions that occur infrequently.

You can think of your system like a physical EFTPOS terminal. A terminal transactions on behalf of a user and displays the response. It must also deal with error conditions like communications link failures.

Where a response is received, the response.summaryCode may be used to determine the appropriate system behaviour. Recommended system actions based on these responses is included in the following table.

Summary Code	Description	Recommended System Action
0	Transaction Approved	Transaction is successful. No further action required.

Summary Code	Description	Recommended System Action
1	Transaction Declined	Transaction has been declined by the financial institution. Some of the more common reasons are invalid credit card details (QQ), expired card (54) or insufficient funds (51). In many cases, the problem can be addressed by either: ➢ Carefully checking the card details and correcting any mistakes before retrying under a new orderNumber; or ➢ Retrying the transaction under a new orderNumber with an alternative card of the card holder. The reason for the decline may not always be obvious from the detailed response code eg Do Not Honour (05). In this case, ask for an alternative means of payment from the card holder.
2	Transaction Erred	Transaction is of an unknown status. Please refer to section below for details on how to handle this status.
3	Transaction Rejected	Transaction request has been rejected by the St.George Credit Card API, often due to invalid parameters or system configuration. This is similar to the Transaction Declined (Summary Code of 1) in terms of error handling. Refer to the detailed response code and either: ➢ Correct the transaction details if required and retry under a new orderNumber; or ➢ Handle offline through St.George/Qvalent CustomerCare if you are unable to resolve the issue.

Table 3.5 – Summary Codes and appropriate actions

Summary Code 2 – Transaction Erred

When implementing the API, you should take care to correctly handle summary code 2 responses. In an ideal world, all transactions would simply succeed or fail. Unfortunately, certain circumstances can mean that your system will not know the status of the transaction. For example, a transaction may be received and processed by PayWay but a network error may occur such that a response is not returned to your system. In this case, your system will receive the "Transaction Erred" summary code or some other network/timeout error. The network outage may last for some time and therefore your exception handling process must handle this rare but possible situation.

Your system needs to perform a query transaction to ensure that the payment status is always known. Your system should also alert your support staff that a network error has occurred so that they can investigate the root cause. If they cannot identify the cause of the issue and the issue is ongoing, they should contact PayWay customer care as per

Appendix B.

Firstly, your system should record the transaction status as QI "Transaction Incomplete". This status should also be presented to the person using your system (e.g. the card holder or one of your operators). You should inform the user that they should not retry the payment, but instead contact you to confirm the status of the transaction.

Secondly, your system must perform a query to discover the transaction status. You have two options to accomplish this:

1. Your system can keep sending the query transaction to PayWay until it is successful.

2. Your system can inform an operator that a transaction must be queried and they can manually perform the query through the search screens available on the PayWay web site.

The manual option makes your system development simpler, but makes your support processes more complicated. In either case, you must record the order number of the original transaction for future reference. Pseudo-code for both options is provided below.

Manual Querying

```
// Initial transaction attempt
ccResponse = processCreditCard ( 'capture'/'refund', other API parameters )
if ccResponse.summaryCode != '2' or network error
   // This is the normal condition - transaction attempt is complete
   Store transaction response and handle response
else
{
   // Transaction status is unknown
   Store transaction status as 'QI'
   Alert support personnel to perform a manual query
   Alert support personnel to investigate network connectivity
}
```

Your support personnel will search for the transaction by order number, so ensure that the order number is included in the alert that is sent to them.

Automatic Querying

Building a system to reliably query a transaction can be a complex task. You will need to permanently store a record to indicate that the transaction requires querying. These records need to be added to a query queue and this queue must be processed by a separate background thread in your application. When a query is successful, your system will update the query record accordingly and remove it from the queue. It will also update the status of the original transaction. You will also need to ensure that when your system starts up, any outstanding query records are added to the queue.

```
// Initial transaction attempt
ccResponse = processCreditCard ( 'capture'/'refund', other API parameters )
if ccResponse.responseCode != '2' or network error
   // This is the normal condition - transaction attempt is complete
   Store transaction response and handle response
else
{
   // Transaction status is unknown
   Store transaction status as 'QI'
   Alert support personnel to investigate network connectivity
   Store query record and add to query queue
}
```

The query queue processing logic is below:

```
Repeat
   Wait 60 seconds
   Remove next query from the queue
   ccResponse = processCreditCard ( 'query', other API parameters )
   if ccResponse.summaryCode = '0', '1' or '3'
      // Query was successful
      Record that the query succeeded
      Update the original transaction status
   else
      // Query must be retried
      Add this query back on to the queue
End Repeat
```

When creating the query request, you must use the same order number as the original transaction. See section 3.3.1.4 for more information. The query record that you store should include this information.

3.5 Registering Customers

This section provides information on using PayWay API to register, manage and deregister credit card customers. This feature requires that you have both the API and Recurring Billing and Customer Vault modules.

3.5.1 Creating or Updating a Customer

To register a new credit card customer or modify an existing customer, you must provide the following parameters in your request. All parameters are mandatory unless otherwise noted.

Parameter Name	Value
order.type	"registerAccount"
customer.username	As described in section 3.2.1.
customer.password	
customer.merchant	
customer.customerReferenceNumber	The unique reference number for the pre-registered customer. To create a new account, provide a new customer reference number. To modify an existing account, provide an existing customer reference number.

Parameter Name	Value
card.PAN	The card number to register against this customer reference number
card.expiryYear	The expiry date for this credit card
card.expiryMonth	
card.cardHolderName	The name on the credit card (optional)

The response will contain one of the following response codes:

Response Code	Description	Account Registered
00	The customer has been registered in PayWay for future use in credit card transactions. See section 3.2.1 under customer.customerReferenceNumber for more information.	Yes
QE	A system error occurred while registering the account. The customer has not been saved and can be registered again later.	No
14	The card number provided does not pass the check digit routine.	No
QY	Your PayWay facility does not accept this card type.	No
54	The expiry date provided is in the past.	No
QA	Invalid parameters were provided in your request. See the response text for more information.	No

3.5.2 Transacting with Registered Customers

The registered credit card details can be used in any of the following processing requests:

- capture
- refund
- preauth
- captureWithoutAuth
- reversal

To use the registered card details, you must provide the parameter customer.customerReferenceNumber. This field can be provided as an alternative to card.cardHolderName, card.expiryYear, card.expiryMonth, card.CVN and card.PAN.

3.5.3 Stopping Remaining Payments

To stop remaining payments for a registered customer, you must provide the following parameters in your request. All parameters are mandatory.

Parameter Name	Value
order.type	"deregisterAccount"

Parameter Name	Value
customer.username	As described in section 3.2.1.
customer.password	
customer.merchant	
customer.customerReferenceNumber	The unique reference number for the pre-registered customer. A customer must have been previously registered with this customer reference number.

The response will contain one of the following response codes:

Response Code	Description	Account Deregistered
00	The customer has had remaining payments stopped in PayWay and cannot be used in future credit card transactions.	Yes
QE	A system error occurred while stopping the customer's remaining payments. Check that the customer reference number has actually been registered in PayWay previously.	No
QA	Invalid parameters were provided in your request. See the response text for more information.	No

Appendix A – Common Response Codes

These response codes have been included for your reference and are derived from the message format defined in Australian Standard 2805.2 (1997).

The table below lists the most commonly received response codes. As a general rule you should use the summary response code, which is supplied to determine whether a transaction is approved or declined. The actual reason for a decline is often not important, and the situation can usually be resolved by verifying the card details with the customer, or asking them for a different card number.

Valid response codes are of a two digit alphanumeric format.

If an unknown response code is returned please contact St.George with the appropriate transaction details.

Please note that there are no response codes specific to card verification number mismatches. This is because no financial institutions in Australia currently return any such information if declining a transaction.

Both the code and description of a response will be supplied by the API.

Summary Code	Description
0	Transaction Approved
1	Transaction Declined
2	Transaction Erred
3	Transaction Rejected

Response Code	Description	Summary Code
00	Approved or completed successfully	0
01	Refer to card issuer	1
03	Invalid merchant	1
04	Pick-up card	1
05	Do not honour	1
08	Honour with identification	0
12	Invalid transaction	1
13	Invalid amount	1
14	Invalid card number (no such number)	1
30	Format error	1

Response Code	Description	Summary Code
36	Restricted card	1
41	Lost card	1
42	No universal account	1
43	Stolen card, pick up	1
51	Not sufficient funds	1
54	Expired card	1
61	Exceeds withdrawal amount limits	1
62	Restricted card	1
65	Exceeds withdrawal frequency limit	1
91	Issuer or switch is inoperative	1
92	Financial institution or intermediate network facility cannot be found for routing	1
94	Duplicate transmission	1
Q1	Unknown Buyer	1
Q2	Transaction Pending	2
Q3	Payment Gateway Connection Error	3
Q4	Payment Gateway Unavailable	1
Q5	Invalid Transaction	1
Q6	Duplicate Transaction – requery to determine status	3
QA	Invalid parameters or Initialisation failed	3
QB	Order type not currently supported	3
QC	Invalid Order Type	3
QD	Invalid Payment Amount - Payment amount less than minimum/exceeds maximum allowed limit	1
QE	Internal Error	3
QF	Transaction Failed	3
QG	Unknown Customer Order Number	3
QH	Unknown Customer Username or Password	3
QI	Transaction incomplete - contact St.George to confirm reconciliation	2
QJ	Invalid Client Certificate	3
QK	Unknown Customer Merchant	3
QL	Business Group not configured for customer	3

Response Code	**Description**	**Summary Code**
QM	Payment Instrument not configured for customer	3
QN	Configuration Error	1
QO	Missing Payment Instrument	3
QP	Missing Supplier Account	3
QQ	Invalid Credit Card \ Invalid Credit Card Verification Number	1
QR	Transaction Retry	2
QS	Transaction Successful	0
QT	Invalid currency	3
QU	Unknown Customer IP Address	3
QV	Invalid Original Order Number specified for Refund, Refund amount exceeds capture amount, or Previous capture was not approved	1
QW	Invalid Reference Number	1
QX	Network Error has occurred	2
QY	Card Type Not Accepted	1
QZ	Zero value transaction	0

Common Response Code Descriptions

00 – Approved. This indicates that the transaction has been authorised.

What authorisation DOES mean:-

- The card number is valid
- The card has not been reported lost or stolen (although it may in fact be lost, stolen or compromised [card details improperly obtained or copied] and the card owner is unaware)
- There are sufficient funds available to cover the transaction.

What authorisation DOES NOT mean:-

- An authorisation does NOT confirm that the person providing the card number is the legitimate cardholder. The risk remains that the person providing the credit card number has either stolen or improperly obtained the card.
- There is also the risk that the purchaser has compromised (improperly obtained) the card number, without being in posession of the card.

Although it is imported to obtain an authorisation for each transaction, it does not protect you from the risk of fraud or chargeback. The risk of fraud remains even though authorisation has been obtained.

01 - Refer to Issuer
This indicates an error or problem on the issuer's side. The problem may be related to the card holder's account. In general the reason for this response code may be any of the following:-

- Suspected Fraud
- Insufficient Funds
- Stolen Card
- Expired Card
- Invalid CVN
- Any other rule imposed by the card issuer that causes a decline (e.g. daily limit exceeded, duplicate transaction suspected, etc).

03 - Invalid Merchant
This can be returned by St.George when there is a problem with the merchant configuration. This can also be returned for AMEX transactions when there is a problem with the setup at American Express. This code can be returned from an issuing bank if they don't like the acquiring bank. An example of this would be someone trying to pay their speeding fine with an overseas credit card. The overseas issuing bank would return a 03, indicating that they wouldn't allow the transaction over the internet for an Australian bank.

04 - Pickup Card
Error code 04 normally means that the card has been reported as lost or stolen. In all cases where this response code is being returned and the customer does not know why they need to follow this up with the issuing bank.

05 - Do Not Honour
This code is usually returned from St.George for St.George issued cards for similar reasons that other issuers return 01. It can indicate any of the following:-

- Suspected Fraud
- Insufficient Funds
- Stolen Card
- Expired Card
- Invalid CVN
- Any other rule imposed by the card issuer that causes a decline (e.g. daily limit exceeded, duplicate transaction suspected, etc).

08 – Honour with identification. This indicates that the transaction has been authorised.

What authorisation DOES mean:-

- The card number is valid
- The card has not been reported lost or stolen (although it may in fact be lost, stolen or compromised [card details improperly obtained or copied] and the card owner is unaware)
- There are sufficient funds available to cover the transaction.

What authorisation DOES NOT mean:-

- An authorisation does NOT confirm that the person providing the card number is the legitimate cardholder. The risk remains that the person providing the credit card number has either stolen or improperly obtained the card.
- There is also the risk that the purchaser has compromised (improperly obtained) the card number, without being in posession of the card.

Although it is imported to obtain an authorisation for each transaction, it does not protect you from the risk of fraud or chargeback. The risk of fraud remains even though authorisation has been obtained.

12 - Invalid Transaction
This code is often returned from the issuer when they do not accept the transaction. This can possibly be when a transaction for the same amount and merchant is attempted multiple times quickly for the same card. The best approach is for the card holder to contact their issuing bank.

14 - Invalid card number (no such number)
This code indicates that the card number either did not pass the check digit algorithm, or is not an account that exists at the issuing bank. St.George returns this code if the card number passes the check digit algorithm, but is not an existing card. St.George also returns this code if an AMEX card is used, but the merchant is not setup for AMEX cards at the St.George end.

22 - Suspected Malfunction
St.George returns this code if the card number does not pass the check digit algorithm. This is considered a malfunction, since St.George expect the terminal to check the card number before transmission.

42 - No Universal Account
This error is returned from some issuers when the credit account does not exist at the issuing bank. This situation is similar to the 14 response code - the card number passes the check digit algorithm, but there is no credit account associated with the card number.

51 – Not sufficient funds
61 – Exceeds withdrawal amount limits
This error is returned when the card holder does not have enough credit to pay the specified amount. Ask the card holder if they have another card to use for the payment.

54 – Expired Card
This error is returned when the wrong expiry date has been entered for the credit card. Check that the expiry date is correct and attempt the transaction again. If the transaction still does not work, check with the card holder to see if they have a new card with a new expiry date.

91 - Issuer or switch is inoperative
This code is used to indicate that the next party in a credit card transaction timed out and the transaction has been reversed. This may happen between PayWay and St.George, or further down the chain.

92 - Financial institution or intermediate network facility cannot be found for routing
The card number is incorrect. The first 6 digits of the credit card number indicate which bank issued the card. These are used for routing credit card requests through the credit card network to the issuing bank. This error indicates that there is no bank that corresponds to the first 6 digits of the card number.

QA - Invalid Parameters
Invalid parameters passed to API, for example, missing credit card number or customer number. The response text will contain more detail about which parameters are invalid.

QI - Transaction incomplete
This response code indicates that a request message was sent to the PayWay server but no response was received within the timeout period. This could be caused by incorrect proxy configuration, so you should verify that your proxy information is correct. This could also be caused by a network connectivity issue between your server and the PayWay server. See Appendix B for more information.

QK - Invalid Merchant
The merchant Id passed in the request is not recognised by PayWay. Check that you are specifying either 'TEST' or your 8-digit St.George merchant Id in the 'customer.merchant' request parameter. You can check your 8-digit merchant Id by clicking the 'Merchants' link on the PayWay website. If the correct Merchant Id does not appear, contact your implementation manager.

Note: You must press the 'Go Live' button in the PayWay website in order to enable live transactions against your 8-digit St.George merchant Id (see section 2.3.1). After you have pressed the 'Go Live' button, you may continue to use you TEST merchant to develop and test your implementation. Transactions to your TEST merchant will not appear on the cardholder statement or your bank account.

QH – Unknown Customer Username or Password
The customer.username and/or customer.password parameters that you are passing are incorrect. Note that this username and password is for your application to connect with the PayWay API server and is different from the username and password that you use to login to the PayWay web site. To find the correct values, login to the PayWay website. The username and password are listed on the "Security" page under the "Setup API" heading.

QT – Invalid Currency
You must always pass "AUD" for the card.currency parameter.

The PayWay API only accepts transactions in Australian Dollars. However, you can charge credit cards from outside Australia. You will receive funds in Australian Dollars. The transaction will appear on the cardholder statement in their currency using an exchange rate determined by the card scheme (Visa, Bankcard, American Express, etc).

QQ - Invalid Card
This error code indicates that the credit card details (card number, expiry date or CVN) are invalid. This could be because the card number does not meet check digit validation, an invalid expiry date was entered or an invalid CVN was entered.

QY - Card Type not accepted
The Merchant is not enabled for the particular Card Scheme (normally returned for American Express and Diners Club cards). To register for American Express or Diners Club, click the **Register to accept Amex or Diners through PayWay** link on the "Merchants" page. Alternatively, you may have entered a bad card number with too many or too few digits.

Other Response Codes
If you receive a numeric response code other than those listed in this section, you should check that the card details are correct. If they are, ask the card holder for an alternative credit card. If this still does not resolve the problem, the card holder should contact their issuing bank.

If you receive a response code starting with 'Q' that you do not understand, you should contact Customer Care as per section 2.3.2.

Appendix B – Dealing with QI Responses

A QI response from the API most often indicates that there was a network error between your server and the PayWay server. Your server cannot know the transaction status because either the request did not arrive at the PayWay server, or the response did not make it back to your server.

If you have followed the recommendations in section 3.4, your software should correctly handle the QI response code using one of the following options:

1. Querying the transaction via the API to find out its status (see section 3.3.1.4), or

2. Raising an alert for your support staff to handle

Your support staff can always determine the final status of the transaction using the search pages in the PayWay web site at https://payway.stgeorge.com.au/[6]. Depending on your system, your support staff may be able to update the transaction status in your system manually. If a transaction does not appear on these pages, the transaction was not received or processed by the PayWay server.

Network errors can be almost impossible to investigate after the fact, so it is vital that you investigate while you are experiencing an issue. You should try these steps:

1. If you use a proxy, check that your proxy settings are correct, and check that there are no other issues with your proxy server.

2. If you do not use a proxy server, login to your server and telnet to payway.stgeorge.com.au on port 443. Record the results of this test for use by your network administrator.

3. Contact your network administrator and ask him/her to investigate the issue. The results of the telnet test can help him/her to begin the investigation.

4. If you still cannot locate the cause of the QI responses you should contact PayWay support for assistance on **1300 395 501**. Your network administrators will need to be available to work through the problem with the PayWay network administrators.

[6] It may take a few minutes for the transaction to appear on the PayWay search pages.