Westpac Banking Corporation ABN 33 007 457 141

QuickGateway

Technical Implementation Guide

Document History

Date	Version	Description
2-August-2004	8.0	Added recommendations for preventing duplicate payments (Section 3.3.1.6). Corrected details on when SettlementDate is returned (Section 3.3.2.1) Provided clarification on query orders and previousTxn field (Section 3.3.1.4) Provided additional details on Echo order (Section 3.3.1.5) Added upgrade to Java client v7.4.3 Added upgrade to Java client v8.0 (JDK1.4 compliant) Added reference to SOAP WSDL files Changed KeyToolGUI details due to licensing updates. Added more information on client certification tools
15-April-2005	8.0.1	Added directive to trust root certificate in Section 2.1.
20-May-2005	8.0.2	Reorganised introductory sections to assist customer understanding. Removed references to old Java client versions from Section 4. Removed SOAP examples from Section 3 as these will vary between technologies. Removed Appendix B on managing certificates as future SSL certificates will be issued by Qvalent. Corrected QH and QJ response descriptions in Appendix A.
7-June-2005	8.1	Added details about pre-authorisations. Separated technology specific sections into separate documents.
19 Dec 2007	8.2	Added recurring payment information
12 Feb 2008	9.0	Added pre-registered accounts information Added 3D Secure information
5th Jan 2009	9.1	Changed document to new product name Added some further documentation for the 9.0 clients
12th April 2011	9.2	Added pre-registration code as parameter to 3.4.3 section on pre-registered cards.

Table of Contents

1 Introduction

Westpac is utilising technology developed by our Qvalent subsidiary 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 and New Zealand. This is provided as part of the Quickstream product suite and is called QuickGateway.

1.1 What is QuickGateway?

The QuickGateway solution allows customers to process credit cards using an Application Programming Interface (API). This API is network based and does not necessarily require any software installed on the customer's site. All communications between the customer's system and API will take place in a secure manner.

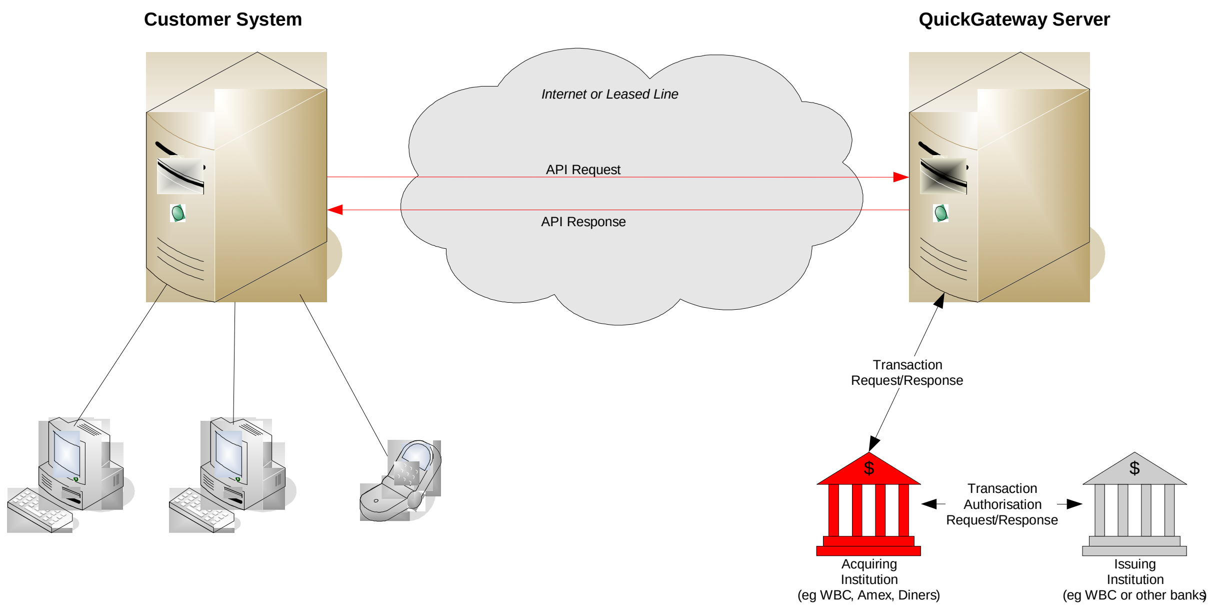
The API will offer the customer a range of credit card processing features including:

- 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 (HTTPS POST, Java, SOAP, ASP, .NET) so there is a solution to suit any customer. All major card types such as Visa, MasterCard, Amex, JCB and Diners may be processed through QuickGateway.

The actual concept of the API is simple. The Customer System will send a API Request to QuickGateway, containing the transaction details and will receive back an API Response containing details on whether the transaction was successful or not.

Solution Overview


Customer System
QuickGateway Server
Internet or Leased Line
API Request
API Response
Transaction Request/Response
$
Transaction Authorisation Request/Response
Acquiring Institution (eg WBC, Amex, Diners)
Issuing Institution (eg WBC or other banks)

1.2 Is it Secure?

For a solution of this nature, security is critical. Westpac 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 following measures will be adopted:

- The Customer 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.
- Qvalent will only accept requests from certain 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.
- For added security a card verification number (CVN) can be supplied with the QuickGateway API call.
- All transaction data will be communicated via HTTPS with 128-bit encryption

The connectivity options are as follows:

- Leased line. Dedicated link between the customer's site and Westpac's credit card server. In this scenario, no data would be transmitted over the Internet. Username / password are still mandatory. SSL certificates are not mandatory but recommended in case of failure of leased line requiring a failover to the Internet.

- Internet. SSL client certificate exchange with username / password over HTTPS. With client certificates, Westpac can be assured of the source of the request. We will supply Test and Production 128-bit SSL client certificates for your use.

Where a customer system interfaces with the QuickGateway API, the customer system will establish a trust relationship with the QuickGateway Server. The QuickGateway Server holds an SSL certificate issued by Verisign to Qvalent. This is different to the SSL client certificate issued by Qvalent and supplied to you. The latter is provided to be installed on your system, whereas the former should not. Customers **must trust the root level certificate (ie that issued by Verisign) and not the certificate issued to Qvalent**. The certificates issued to Qvalent expire and are replaced each year, whereas the Verisign root certificates have a much longer life (ie expiry in 2028). Any customers that trust the certificate issued to Qvalent directly will cease to operate as replacement certificates are installed and therefore this must be avoided.

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 nor 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 may be 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 Westpac QuickGateway 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 at first a quick skim through and then start working through the steps documented in Section 2 in order to implement the solution. If you have any questions that cannot be answered from the document, please direct these to your Westpac implementation contacts.

Section	**Why You Should Read It**
2	This section provides a step-by-step guide to implementing the QuickGateway API. This is where all customers should start.

3	This section is where most of the technical specification of the QuickGateway 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 full list of possible response codes in Australia and New Zealand.
Appendix B	This contains a guide to troubleshooting connectivity problems using Internet Explorer.
Appendix C	This contains information on dealing with network errors when processing transactions.
Glossary	This contains a glossary of terms used in the document.

If you think there is something missing in this document that we should include in the future or you discover an error that we should correct, please bring this to the attention of your Westpac technical contact and we will include in a future version.

2 Implementing the QuickGateway API

This section details the process of implementing the Westpac QuickGateway API into your business. This includes how you will be assisted in the implementation by Westpac, and by our subsidiary Qvalent, from the initial testing through to production usage.

2.1 Testing

For any customer wishing to implement the QuickGateway API, we provide a test environment that is accessible via the internet[1]. This environment is able to simulate a live gateway but may be used without any risk of transactions actually being processed through the banking system.

The typical steps that might be involved in testing are as follows:

1. You will be provided with this document to provide background information about the API.

2. One of our staff will be nominated to help you through the testing and we will put them in contact with you. They will create a test merchant for your use in our test environment and provide the username, password and merchant ID to use. In order to set this up correctly, we will need to know some details, such as what your accepted card types are and the IP address of your test server(s). We will ask these questions as part of the setup process.

3. You must use an SSL certificate to connect to QuickGateway. This allows our server to verify that the transactions requests are coming from an authorised source. We will provide a test SSL certificate for your use during testing. We will also provide a production SSL certificate once testing is complete.

4. The URL you will use to connect depends on the programming environment you intend to use:

   a. If you are using a HTTPS Post to connect to the API, the URL is https://ccapi.client.support.qvalent.com/post/CreditCardAPIReceiver.

   b. If you are using SOAP to connect to the API, the URL is https://ccapi.client.support.qvalent.com/axis/services/CardsAPI.

5. You should then attempt to connect to our API using your chosen mechanism. You may receive a "QU" for Unknown Customer IP Address response if you have not provided us with your current IP address. You may also test connectivity using Internet Explorer as described in Appendix B - Connectivity Troubleshooting with Internet Explorer.

[1] May also be accessed via a leased line for customers wishing to do this in Production, however via the internet is a more cost-effective approach for the testing phase.

6. You may try further transactions (again, if using Java the CreditCardAPITester application may be used for this) and should now receive other responses.

7. You should now develop your client application using your chosen programming environment to access the API. As stated earlier, it is important to move as soon as possible to SSL certificate testing if you are intending to conduct API calls over the internet.

Some additional options during the testing phase might be:

- You may request a web user/password to access online facilities (see Section 3.4). During testing this may be just a single account.

- You may wish to receive the daily reconciliation email (see Section 3.5) during your testing. If so, you may provide your email address and we will configure this for you.

2.1.1 Testing Exception Handling

You may use our test environment, to perform unit and acceptance testing of your application. The most critical element of your testing should be to test your application's exception handling as documented in Section 3.3.3.4.

The test environment assists you in performing these testing by providing various ways of simulating transaction responses, as per the list in Appendix A – Response Codes. Your test facility may be setup with any of the following options, some of which are driven by what you include in the API request.

Simulator Option	**Response Returned**
Cents	The transaction response is based on the cents portion of the order.amount included in the request. This only allows testing of numeric responses. Unless otherwise specified, this will be the default option set up. See following text for an example of this.
Order Number Prefix	The transaction response is based on the first two characters of the customer.orderNumber included in the request. This allows testing of alphanumeric response codes.
Order Number Suffix	The transaction response is based on the last two characters of the customer.orderNumber included in the request. This allows testing of alphanumeric response codes.
Success	The transaction response is always set to success.

The simulator can also simulate delays in responding to transactions. This can be used to test your application's handling of timeouts. Please ask your Westpac technical contact if you wish to include an extended delay in responses.

Please note: the response codes operate differently for the NZ gateway. More details will be provided during the implementation for customers wishing to accept transactions in NZ Dollars.

<u>How does the "Cents" simulator option work?</u>

In order to generate a given response code, the required response code should be included as the rightmost two digits of the order.amount field. For example:

- If order.amount = 10000 (ie $100.00), the response code returned will be '00' (ie successful) with a summary code of '0'
- If order.amount = 10054 (ie $100.54), the response code returned will be '54' (ie expired card) with a summary code of '1'

This will allow testing of the numeric response codes. If modifying the amount is not possible or testing of alphanumeric responses if required, then the Order Number Prefix/Suffix options may be selected instead.

2.1.2 Other Key Tests

It is also critical that you ensure that your front-end application prevents duplicate payments. See Section 3.3.2.3 for recommendations on how this may be performed.

2.1.3 Test Environment Availability

We will endeavour to ensure that the test environment is available as often as possible, during business hours and after hours. However, the environment is unavailable from time-to-time for upgrade purposes. If you believe that the test environment may be unavailable, you may contact your support contacts (see Section 2.2.2) to see when the service will be restored.

Also, if you are planning significant test periods or load testing, please contact your support contacts to ensure that the test environment will be available during these periods.

2.2 Production

2.2.1 Migration

The migration to Production will be planned in conjunction with you to ensure that it is as seamless as possible. Before commencing the migration to Production, the following must occur:

- Connectivity to be setup, either over the internet or leased line. If using the internet, we will provide your Production SSL Certificate to you via a secure means.
- Your production Merchant IDs are to be provided ie. Visa/Mastercard (to be arranged by Westpac Implementation Manager) plus Amex and Diners if accepted.

- You should indicate if you expect any transaction limits to be applied or if refunds will be possible with/without reference to a capture.

- You should provide a reconciliation email address if required in Production.

- You should provide administration user details if required in Production ie User name, email address, access required (Credit Card Payments, Refunds and/or Searches)

The typical steps that might be involved in the production migration are as follows:

1. We will configure your merchant details in Production and setup your Production merchant account.

2. In order to ensure that the banking side is correctly configured prior to live testing, we will initiate a low-value capture transaction (ie less than $2.00) through each merchant facility to confirm that the transaction is accepted. The following day, the Westpac Implementation Manager will confirm that the transactions have appeared correctly on the bank statements. Following confirmation, we will reverse the transactions through a refund transaction and confirm that this has been registered on the bank statement.

3. We will provide the Production URL, username, password and merchant as well as web user/passwords for online facilities. The Production URL is the same as what is used in testing with ".support" removed. For example, https://ccapi.client.qvalent.com/... instead of https://ccapi.client.support.qvalent.com/...

4. Prior to connectivity testing, your account will be temporarily pointed to the test gateway to ensure that transaction attempts are not actually processed through the banking network. You should then attempt to call the API from your production server using the Production URL and merchant details.

5. Following successful connectivity testing, we will reconfigure your account to be completely live. It is recommended that you perform a production test for each accepted card scheme (eg Visa/Mastercard, Amex and Diners) to confirm that the setup is complete before commencing full live usage. This would include reconciling the transactions on your bank statement as correct.

2.2.2 Support

During the testing phase, all support will be provided by the person nominated to assist during this phase. However, after migrating to Production, support will be provided via a more formal process. Prior to contacting support, we recommend that some basic troubleshooting be performed:

- Check the most recent 10 transactions or query the online transaction facility to determine if the issue is an isolated, transient issue or a more serious outage.

- Check if there are any scheduled or unscheduled outages noted on the Qvalent web site (http://www.qvalent.com/technicalAdv.jsp). You may subscribe your email address onto the site to receive notification of scheduled maintenance changes.

If the troubleshooting does not resolve a production issue, contact support as described on http://www.qvalent.com/services/support.jsp:

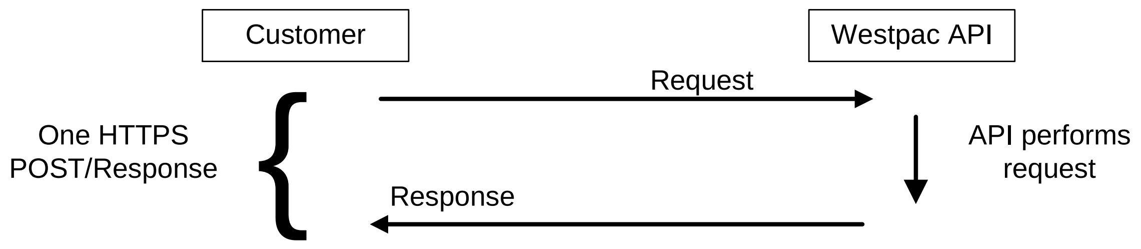
- For urgent issues where you are unable to process transactions through the API for a sustained period, contact **Quickstream Customer Care** on **1300 726 370** (24 hrs/7 days). They will ensure you are able to get in contact with the appropriate person to resolve the issue as soon as possible.
- For less urgent issues, including reconciliation issues, that may be handled during normal business hours contact **Quickstream Customer Care** either via email (**presentandpay@qvalent.com**) or phone (**1300 726 370**).

3 Application Programming Interface

The QuickGateway API provides a real-time, synchronous interface to execute credit card transactions. That is, only one method call is needed to request processing against a credit card. This method will complete in a synchronous manner. This removes the complexities of issuing a request then having to process a separate response message sometime in the future. In this model, a customer will issue the credit card request, then receive the results of the request back in the same call, all in real-time.

The following figure illustrates the steps in a Request-Response interaction between a customer using the API:

Message Flows



This transaction contains the following steps:

1. Customer initiates a HTTPS connection to the provided API URL.

2. The Customer uses a POST operation to send the API request through the HTTPS connection.

3. The Customer waits for a response to return through the HTTPS connection.

4. The API 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 API server contacts the issuing bank and requests authorisation for the credit card transaction.

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

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

8. The Customer closes the HTTPS connection established in step 1.

This process is then repeated for further Request/Response cycles to process additional credit cards.

3.1 Connecting to the API

Qvalent provides software packages for some technologies to assist you in the task of connecting to the API. This software provides an object in the relevant programming language which performs all the communication with the Westpac API server. Software packages are available for the following technologies:

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

These software packages include additional documentation on how to install and use the API software in your application.

If your application does not use one of the supported technologies, you can develop your own connectivity using either SOAP or HTTPS POSTs. You can also choose this option if you do not wish to install extra software on your servers. More details about how to do this are provided in the document entitled "Custom Integration" which can be provided on request.

Regardless of the method you use to connect to the API, the parameters you send and receive are the same. These parameters are described in the following section.

3.2 API Messages

3.2.1 API Request

To make a request to process a credit card, the body of the message would include the following fields:

Parameter Name	Value	Description	Required
order.type	See Description for valid order types	The type of processing required: capture refund query echo preauth captureWithoutAuth reversal	Yes
customer.username	At most 32 chars	The username that identifies your company.	Yes
customer.password	At most 32 chars	The password provided to you with your username.	Yes
customer.merchant	At most 32 chars	The merchant identifier to use for the transaction. This allows you to have multiple merchant accounts. Values for this field will be provided to you by Westpac.	Yes
card.PAN	At most 19 chars	The credit card number	Required for capture / preauth, optional for captureWithoutAuth / refund / reversal
card.CVN	4 digits	The card verification number. This is a 3 or 4 digit code that provides extra security for online payments. For Visa, MasterCard 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.	No
card.expiryYear	2 digits	The year the card expires	Required for capture / preauth, optional for captureWithoutAuth / refund / reversal

Parameter Name	Value	Description	Required
card.expiryMonth	2 digits	The month the card expires	Required for capture / preauth, optional for captureWithoutAuth / refund / reversal
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. Note: Do not use any of the following characters in the card holder name: & % +	Optional
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". Limits may be applied to these amounts for security purposes (see Section 3.3.2.2).	Required for capture / preauth / captureWithoutAuth, optional for refund / reversal
customer.orderNumber	At most 40 chars	A unique customer generated number for this transaction. 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 Quickstream 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: & % +	Only for capture / refund / query
card.currency	3 chars	The currency for the transaction amount. If no currency is specified, your pre-configured default currency is used.	Optional for capture / refund
order.ECI	3 chars or 1 digit	The Electronic Commerce Indicator (ECI) to apply to this card transaction. Please refer to Section 3.3.2.1 for more details on this parameter.	Only for capture / refund

Parameter Name	Value	Description	Required
customer.originalOrderNumber	At most 40 chars	This field is used to reference a previous transaction in the system when performing a refund, reversal or captureWithoutAuth. This field is used when performing a Refund to link the refund with the original Capture. See Section 3.3.1.2. Also required when performing a reversal of a preauth/capture. In this case, this parameter must be populated with the original order number of the preauth/capture transaction. See Section 3.4.2. 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 3.4.1. Note that this field was previously named customer.captureOrderNumber	Required for refund / reversal / captureWithoutAuth
order.authId	6 chars	The authorisation Id returned from the corresponding preauth transaction. Note that this field may contain spaces, so be careful not to trim spaces from this field.	Optional for captureWithoutAuth, must not be present for other transaction types.
customer.customerReferenceNumber	At most 20 chars	The customer reference number to record against this transaction. This field will appear in any transaction reports that you receive. Only letters, numbers, dashes, underscores and full-stops are accepted in this field. This field is used to identify the customer's account if you are using pre-registered accounts. See section 3.4.3.	Optional
customer.preregistrationCode	At most 20 chars	This field is used to identify the customer's account if you are using pre-registered accounts. Can be used as well as customerReferenceNumber or instead of it. See section 3.4.3.	Optional
order.xid	20 chars	The transaction ID (XID) value for the 3D Secure authentication for this transaction.	No
order.cavv	28 chars or 32 chars	The base 64 encoded Cardholder Authentication Verification Value (CAVV) for the 3D Secure authentication for this transaction. This value is returned from your Merchant Plugin Interface (MPI) software.	No
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	Yes if the Fraud Guard module is used, No otherwise.
message.end	No value	Must be the last property in the message	Yes

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

3.2.2 API Response

In addition to the standard HTTPS response code (i.e. 200, 404), after processing the following will be sent back to the Customer:

Parameter Name	Value	Description	Always
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.RRN	String (At most 12 characters)	The retrieval reference number provided by the banking network. See Section 3.3.3.6.	May be returned for approved transactions
response.settlementDate	yyyymmdd	The settlement date for the transaction. See Section 3.3.3.1.	May be returned for most approved or declined transactions.
response.previousTxn	number	Indicates if transaction response is returned based on previous transaction response for same order number or if this is a new order (see section 3.3.3.3). 0 = new transaction response 1 = previous transaction response	Only if transaction has been processed (either now or previously)
response.referenceNo	String (At most 32 characters)	Internal Quickstream reference number for transaction.	Only if transaction is successful
response.orderNumber	String	Order number as supplied in transaction request.	Only if supplied in transaction request.

Parameter Name	Value	Description	Always
response.cardSchemeName	String (At most 20 characters)	This is the card scheme of the card used in the transaction (see section 3.3.3.5 for more details). Values returned are: ➢ AMEX ➢ DINERS ➢ MASTERCARD ➢ VISA	Only if card scheme is known
response.creditGroup	String (At most 20 characters)	This is the card scheme grouping of the card used in the transaction (see section 3.3.3.5 for more details). Values returned are: ➢ AMEX ➢ DINERS ➢ VI/BC/MC[2]	Only if card scheme is known
response.transactionDate	dd-mmm-yyyy hh24:mi:ss	The date/time the transaction occured.	Only if transaction passes initial parameter checks.
response.authId	String (At most 6 characters)	The authorisation Id returned from the issuing bank for the preauth transactions. This value may be passed in with any subsequent captureWithoutAuth transaction for the same card.	Only for approved preauth transactions
response.end	No value	Must be the last property in the message	Yes

[2] BC stands for Bankcard which is a discontinued Australian-only card scheme

3.2.3 HTTPS POST Example

3.2.3.1 Request Example

A HTTPS transmission of the API may include the following MIME and HTTPS headers:

```
POST /post/CreditCardAPIReceiver HTTPS/1.0
Content-type: application/x-www-form-urlencoded
Content-length: 328
Host: ccapi.client.support.qvalent.com:443

order.type=refund&customer.username=COMPANYA&customer.password=insurance&cu
stomer.merchant=companya&card.PAN=4242424242424242&card.CVN=564&card.expiry
Year=10&card.expiryMonth=06&order.amount=3400&customer.orderNumber=AB1322-
refund&card.currency=AUD&order.ECI=IVR&customer.originalOrderNumber=AB1322&
order.priority=1&message.end=
```

3.2.3.2 Response Example

```
response.summaryCode=0
response.responseCode=00
response.text=Transaction processed successfully
response.RRN=13238
response.settlementDate=20050103
response.previousTxn=0
response.referenceNo=12922
response.orderNumber=1234567
response.cardSchemeName=VISA
response.creditGroup=VI/BC/MC
response.transactionDate=03-JAN-2005 13:24:41
response.end
```

3.3 Further Details

The following section provides further details on the request and response parameters.

3.3.1 Order Types

This section describes how to perform transactions of the various order types.

3.3.1.1 Capture Orders

A capture represents a purchase transaction by a card-holder. To perform a simple capture, provide the following parameters:

- order.type=capture
- card.PAN, card.expiryYear, card.expiryMonth, card.CVN – credit card details
- customer.username, customer.password, customer.merchant – as provided by Westpac
- order.amount – amount being debited to the card-holder's account
- customer.orderNumber – unique txn reference for the capture

- order.ECI – defined by how you received the card-holder's details (see 3.3.2.1)
- card.currency – provided if you have multiple currencies available
- message.end – to indicate the end of the request

3.3.1.2 Refund Orders

The way that refunds are handled depends on the configuration of a specific merchant. The two options are:

1. Merchant is configured to allow Refunds without reference to a Capture. In this case, the originalOrderNumber parameter does not require population and is ignored. This option is provided for merchants where option 2 is not appropriate and also for backward compatibility
2. Merchant is configured to only allow Refunds with a reference to the original Capture. In this case, the originalOrderNumber parameter must be populated otherwise the call will fail. If the originalOrderNumber is populated, the 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 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.

The 2nd option is recommended if appropriate to provide a higher level of security against fraudulent transactions. 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 txn reference for the refund
- customer.originalOrderNumber - order number of original capture

3.3.1.3 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 again query the banking network to determine an updated response code for the transaction if available.

When executing a Query order, it is important to note that it is possible that the response code may indicate 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. For this reason, the previousTxn field should be checked after receiving a query response. If:

- previousTxn = 1, the query has returned a response related to the original transaction and may be used.

- previousTxn = 0, the response is related to the query operation itself and typically indicates a failure in this query for some reason as further indicated in the response code. The response for the original transaction is not included in this response in this circumstance.

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 in QuickGateway (eg network issues between your server and QuickGateway).

3.3.1.4 Echo Orders

The Echo order type allows your application to interrogate the status of the API server at varying levels depending on the parameters supplied. If you include order.type=echo with:

- No other parameters, a response code of ‘00’ will be returned if the Quickstream server is accessible. This is essentially a network test from your server to Quickstream. However, this does not confirm that your merchant account or the payment gateway is available.

- Your identification details (customer.username, customer.password & customer.merchant) but no currency, your community’s default currency (eg AUD or NZD) will be used. A ‘00’ response returned if your identification details are valid, your IP address is registered and the payment gateway for the default currency is available. Otherwise an appropriate error response is returned.

- Your identification details (customer.username, customer.password & customer.merchant) and a currency is supplied, a ’00’ response is returned if your identification details are valid, your IP address is registered and the payment gateway for the supplied currency is available. Otherwise an appropriate error response is returned.

3.3.2 API Request

3.3.2.1 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.

ECI Value	Description
CCT	Call Centre Transaction
IVR	IVR Transaction
MTO	MOTO Transaction
SSL	Channel Encrypted Transaction (SSL or other)
REC	Recurring payment[1]
5	3D Secure transaction. This is the value returned from your MPI (Merchant Plugin Interface) software for 3D Secure transactions[2]
6	
7	

Table 3.1 – Valid ECI values

[1] The REC ECI value is only available to customers that have applied for the Visa and MasterCard recurring schemes. Please consult your implementation engineer before attempting to use this value.

[2] The numeric ECI values are only available when the transaction is 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. Please consult your implementation engineer before attempting to use ECI 5, 6 or 7.

3.3.2.2 Limits

In order to restrict the amount of transactions, minimum and maximum limits may be configured on request for your merchant. If a transaction request falls outside of these limits, a 'QD' response will be returned.

For example, a $10,000 maximum transaction limit may be configured for your merchant to reduce the risk associated with transactions conducted through the gateway.

3.3.2.3 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 (see Section 0 for more details on previousTxn field in the 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 in 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 otherwise of the first attempt.

- Once a customer has confirmed that they wish to pay, check if you have any approved, pending or incomplete (eg summary code of 2) transactions for the same customer number, credit card number[3] 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.

3.3.3 API Response

3.3.3.1 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.

3.3.3.2 Settlement Date

The settlement date represents the date that the transaction will be settled by the

[3] For customers that do not wish to stored 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.

acquiring bank. The settlement date cut-off is always 6pm (refer to section 3.3.3.1). 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 credited on the settlement date returned. Again, this is up to the relevant acquiring bank (see Table 3.3 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.

Westpac 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 Westpac 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 Quickstream 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.

3.3.3.3 PreviousTxn

The previousTxn field returned in the response indicates whether the response is based on a previous response for a transaction with the same order number or is a new transaction response. This field will assist the client application in determining if a transaction has been duplicated. On receiving a response for a transaction expected to be attempted for the first time, your application should check that previousTxn=0 and handle the exception if this is not the case.

For example, a Capture is performed with orderNumber=1234 and receives a responseCode=03 (also previousTxn=0). If a transaction is attempted with the same orderNumber (ie 1234), then the same responseCode (ie 03) will be returned and previousTxn=1. The transaction will not be processed again.

Although it can produce the same result, it is better to perform a Query operation than to re-attempt the Capture/Refund if a transaction is suspected to have been processed previously. Please refer to 3.3.1.3 for details on Query orders and how the previousTxn field is used in these cases. Also, refer to 3.3.3.4 for more details on error handling.

3.3.3.4 Error Handling

The API will inform the customer's 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.

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.
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 customer. The reason for the decline may not always be obvious from the detailed response code eg Do Not Honour (05) and may require offline followup with the financial institution that issued the card.
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 Westpac QuickGateway 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 Westpac/Qvalent CustomerCare if unable to be resolved.

Table 3.2 – Suggested error handling

<u>Summary Code 2 – Transaction Erred</u>

When implementing the API, the summary response of 2 causes the most concern to our customers. In an ideal world, all transactions would simply succeed or fail. Unfortunately, the complexity of credit card transactions and the interaction with acquiring and issuing financial institutions means it is possible in rare circumstances for a transaction to return with an unknown status. For example, a transaction may be received and processed by a financial institution but a system or network error may occur such that a response it not returned. Given that we do not know the status but it may have been successfully approved and the card debited, we must return the

"Transaction Erred" status. The outage with the financial institution may last for some time and therefore the exception handling process must handle this rare but possible situation.

The recommended action is as follows:

1. Perform a Query order via the API against the original orderNumber and wait for the response. The Quickstream system will return the most up-to-date response of the original transaction attempt. This may be different to the response originally received.

2. If the summaryCode from the Query is 0, 1 or 3 and previousTxn = 1 then the transaction may be handled as if the response was received on the first attempt. If previousTxn = 0, the response is not related to the original transaction but instead relates to the actual Query order. A special case is if a 'QG' (Unknown Customer Order Number) response is received, then the transaction is not known to Quickstream and therefore may be retried under the original or new orderNumber.

3. If the status of the transaction cannot be determined through a single Query, the transaction should be marked as held within the client system. As the transaction may have been potentially processed, it is <u>not</u> recommended that the capture/refund be retried under a new customer order number. Your application should indicate to the end user that the status of the transaction is unknown and that they should contact your customer service personnel during applicable hours or provide a receipt number and contact the end user if the transaction is subsequently found to fail.

4. If the summaryCode remains at a 2 after initial Query attempts, we recommend re-querying after 6:30pm to try to automatically determine the status. Westpac have processes in place to try to update Erred transactions with the final status and will endeavour to have statuses updated by 6:30pm. This means that in many cases, the status will be available automatically if re-queries are attempted after this time.

5. If the summaryCode remains at a 2, we recommend re-querying at the same time on the following day as it may take up to 48 hours for the final status of transactions to be updated under some circumstances.

6. For transactions that remain unresolved, customer service would then try to resolve the situation by first using the online transaction query facility (see Section 3.5) to query the transaction status. If this does not resolve the issue, the exception details should be communicated to Westpac CustomerCare for resolution. This would require details on the original transaction (particularly the customer.orderNumber, time, amount, card number), responses and exception handling actions attempted thus far.

Some customers, with a need for minimal customer impact of outages, have implemented additional business logic that tries to protect the end customer from such issues. As this may be useful in your implementation, a typical process is as follows:

- If an API call returns a summaryCode of 0, 1 or 3 then the customer application handles this as per normal.

- If an API call returns a summaryCode of 2 then the customer application handles this by returning a receipt number to the end customer and then tries to determine the transaction result in the background. This ensures that the end customer is not impacted by an outage period. If the transaction subsequently fails, this will require handling through an internal customer care process and may require contact with the end customer.

Error-handling Pseudocode

The following pseudocode provides an example of how your code may be structured in order to perform the recommended error-handling logic.

```
// Initial transaction attempt
ccResponse = ccRequest.processRequest( capture/refund, other API parameters )
if ccResponse.summaryCode != 2 and ccResponse.previousTxn != 1
   // This is the normal condition - transaction attempt is complete
   Store transaction response and handle response
else if ccResponse.previousTxn = 1
   // Transaction has been previously attempted – order number should not be reused
   Handle this exception
else
{
   // Transaction attempt is incomplete – try to re-query
   Wait x seconds
   ccResponse = ccRequest.processRequest( query, other API parameters )
   if ccResponse.summaryCode != 2 and ccResponse.previousTxn = 1
         // Query has determined final status so no more queries required
      Store transaction response and handle response
   else if ccResponse.responseCode = ‘QG’ (unknown order number)
      // Transaction has not been previously attempted
      Indicate to retry transaction using original (or new) order number
   else
      // Transaction queries have not determined the final status
      Set transaction status to incomplete and query again later or followup manually
}
```

Timeouts

In the case of a timeout error, no summary code or response code may be received. This may occur due to a communications failure between the customer and Westpac. If the customer is unsure if Westpac received the transaction, then it should be handled in the same way as a Summary Code 2.

If repeated timeout errors are received, contact Westpac CustomerCare for resolution.

3.3.3.5 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 fields may be explained in the following.

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 and Mastercard transactions where Westpac 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	Westpac
VISA	VI/BC/MC	Westpac

Table 3.3 – Card scheme, credit group and acquiring bank relationships

3.3.3.6 RRN

The Retrieval Reference Number (RRN) is returned from the banking network and may differ in its implementation for each financial institution and card scheme. As such, it is not guaranteed to be unique and may be null in some cases. In Australia, it should not be used as a unique reference and, as it may be null, cannot be used as a receipt number. If you wish to report a "receipt number" to your customers, the recommended alternative is to use the orderNumber included in the request or the response.referenceNo.

3.4 Advanced QuickGateway API Features

This section describes how to perform pre-authorisation transactions, and how to reverse previous transactions. If you are not already familiar with the terms "pre-authorisation" and "reversal" and how they apply to your business, then you do not need to read this section.

3.4.1 Pre-Authorisations

Most of the transactions you perform using the QuickGateway 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 the 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.

This functionality is not required by most customers and is only made available on request. Customers not setup for these transactions will receive a "QC – Invalid Order Type" response if they attempt to use them.

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 Westpac. Most issuers will keep the funds reserved for at least three days. You should certainly not assume that you can safely perform a "preauth" then perform the "captureWithoutAuth" 4 weeks later. If this short lifetime of pre-authorisations does not fit your business model, you should reconsider your proposed usage of pre-authorisations.

Each pre-authorisation is given an identifier by the issuing bank so that the clearance can be matched to the reserved funds. This identifier is returned in the "response.authId" field from the "preauth" transaction response. It must be passed in the "order.authId" field in the "captureWithoutAuth" transaction request. Every "captureWithoutAuth" must have either an "order.authId" or a "customer.originalOrderNumber", otherwise it will be rejected.

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.

3.4.1.1 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

3.4.1.2 Historical Note

In previous versions of the API, matching of the "preauth" transaction depended on the "order.authId" parameter in the "captureWithoutAuth". This matching depends on the "customer.originalOrderNumber" parameter (as described above). This is now the preferred method since it means that you do not have to store the credit card details. You should upgrade to the new method where possible.

The old method of performing a "captureWithoutAuth" transaction is still accepted by the system. This old method uses the following request parameters:

- order.type = captureWithoutAuth
- card.PAN, card.expiryYear, card.expiryMonth – must be supplied and must match the details given on the original transaction.
- 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 – the value from the response.authId parameter in the response to the original pre- authorisation transaction
- order.amount – the amount to charge the card-holder
- customer.originalOrderNumber – not present

3.4.2 Reversals

As described in section 3.4.1, each capture transaction is comprised of two parts behind the scenes. Since the transaction value does not actually appear on the card holder's statement until the end of that day, it is possible to cancel, or reverse, a transaction before it is cleared. Declined transactions do not need to be reversed.

The cut-off time for reversals is 6pm EST each day. For example, the settlement day of 15 Nov 2005 starts at 6pm on 14 Nov 2005, and ends at 6pm on 15 Nov 2005. Thus, any transaction processed with during the 15 Nov 2005 settlement day can be reversed until 6pm 15 Nov 2005. You cannot reverse any such transactions after this time. Instead, you should perform a refund if the original transaction was a capture, or perform a capture if the original transaction was a refund.

To reverse a transaction, you create the request message and specify the order type as "reversal". You must also place the order number of the transaction you wish to reverse in the capture order number field. You should also specify the card number and expiry date of the original transaction if you have them, since extra checking of the transaction will be performed in this case. Below are the details of the parameters for a reversal transaction:

- card.PAN, card.expiryYear, card.expiryMonth, order.amount – if supplied they must match the details given on the original transaction. If they are not supplied then the values given in the original transaction will be reused.
- customer.orderNumber - new, unique txn reference for the reversal
- customer.originalOrderNumber - order number of original transaction (capture, refund or preauth)

You may reverse capture, refund or pre-authorisation transactions. If you attempt to reverse any other transaction type, you will receive an error response.

If the reversal succeeds, a response code of 00 will be returned, and the response code of the original transaction will be updated to 91. If you attempt to query the original transaction, or view it through the administration screens, this is the status that you will see. Your software should also update the response code of the original transaction to 91 for consistency.

Because of the limited time you have to reverse a transaction, the main use of reversal transactions is to ensure that both your system and the QuickGateway system have a consistent status for a transaction. For example, if you perform a capture and before you receive a response, a network error is encountered, you could then reverse the original transaction so that both systems mark the transaction with a 91 response code.

The following table lists the various reasons that a reversal will not be accepted, and the response code you can expect to receive in that situation.

Situation	**Response Received**
Original transaction not found, i.e. no transaction with the specified capture order number exists in the system.	21 - no action taken
Original transaction was not approved, i.e. the original transaction had a summary code other than 0.	21 - no action taken
Original transaction is not reversible, i.e. a reversal was submitted for a transaction that was not a capture, captureWithoutAuth, refund or preauth.	12 - invalid reversal

Situation	Response Received
Reversal details do not match original transaction details, i.e. different card details were specified in the original transaction and the reversal	12 - invalid reversal
Current Settlement date is not the same as the original transaction, i.e. the original transaction's settlement date is different from the current system settlement date	12 - invalid transaction
Original transaction was already reversed, i.e. a reversal was submitted for a transaction that had previously been reversed	00 - approved or completed successfully
Reversal accepted, i.e. a valid reversal was submitted and processed	00 - approved or completed successfully
Error occurred during processing	QE – Internal Error
Communications error occurred between client and Qvalent	QI/QX - Incomplete or Network error

Processing these response codes is relatively simple. If the response is 00 or 21, then either the transaction has been reversed, or does not need to be reversed. If the response is 12, you should check the details of the transaction you are trying to reverse. For example, you may be specifying the wrong capture order number. If you receive a QE, QI or QX response or a network error, you should re-attempt your reversal.

3.4.3 Pre-registered Card-holder Accounts

The API now provides the option to store card-holder information against a customer reference number or preregistration code (Card Alias/Token), or both. The registering of these details normally takes place on a Westpac-hosted web page. This solution can potentially reduce your obligations under PCI DSS compliance.

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

- order.type – capture, preauth, refund, captureWithoutAuth, reversal (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.preregistrationCode – this is often referred to as the 'card token' and can be used to lookup the stored card details instead of or in combination with the customerReferenceNumber.
- 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.

3.5 Administration

Complementing the QuickGateway is a set of administration features that:

- Allow ad-hoc credit card payments to be made
- Allow refunds to be made against previously successful captures
- Provide searches and export of QuickGateway transactions
- Provide a daily reconciliation email of payment attempts

The online features are accessible via a provided URL and web-based user/password. The site may be accessed over the Internet using recommended web browsers (eg Internet Explorer) and operates securely over HTTPS with 128-bit encryption. To further enhance security, particular features may be turned on or off for individual users.

Refer to the separate "Quickstream User Guide" document for more information.

Appendix A – 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).

It is highly unlikely that you will receive many of these 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.

Valid response codes are of a two digit alphanumeric format.

If an unknown response code is returned please contact Westpac 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

Code	Description	Summary Code
00	Approved or completed successfully	0
01	Refer to card issuer	1
02	Refer to card issuers special conditions	1
03	Invalid merchant	1
04	Pick-up card	1
05	Do not honour	1
06	Error	1
07	Pick-up card, special condition	1
08	Honour with identification	0
09	Request in progress	1
10	Approved for partial amount	0
11	Approved VIP	0

Code	Description	Summary Code
12	Invalid transaction	1
13	Invalid amount	1
14	Invalid card number (no such number)	1
15	No such issuer	1
16	Approved, update Track 3	0
17	Customer cancellation	1
18	Customer dispute	1
19	Re-enter transaction	1
20	Invalid response	1
21	No action taken	1
22	Suspected malfunction	1
23	Unacceptable transaction fee	1
24	File update not supported by receiver	1
25	Unable to locate record on file	1
26	Duplicate file update record, old record replaced	1
27	File update field edit error	1
28	File update file locked out	1
29	File update not successful, contact acquirer	1
30	Format error	1
31	Bank not supported by switch	1
32	Completed partially	1
33	Expired card	1
34	Suspected fraud	1
35	Card acceptor contact acquirer	1
36	Restricted card	1
37	Card acceptor call acquirer security	1
38	Allowable PIN tries exceeded	1
39	No credit account	1
40	Request function not supported	1
41	Lost card	1
42	No universal account	1
43	Stolen card, pick up	1
44	No investment account	1
45-50	Reserved for ISO use	1
51	Not sufficient funds	1
52	No cheque account	1
53	No savings account	1
54	Expired card	1
55	Incorrect PIN	1

Code	Description	Summary Code
56	No card record	1
57	Transaction not permitted to cardholder	1
58	Transaction not permitted to terminal	1
59	Suspected fraud	1
60	Card acceptor contact acquirer	1
61	Exceeds withdrawal amount limits	1
62	Restricted card	1
63	Security violation	1
64	Original amount incorrect	1
65	Exceeds withdrawal frequency limit	1
66	Card acceptor call acquirers security department	1
67	Hard capture (requires that card be picked up at ATM)	1
68	Response received too late	1
69-74	Reserved for ISO use	1
75	Allowable number of PIN tries exceeded	1
76-89	Reserved for private use	1
90	Cutoff is in process (Switch ending a days business and starting the next. The transaction can be sent again in a few minutes).	1
91	Issuer or switch is inoperative	1
92	Financial institution or intermediate network facility cannot be found for routing	1
93	Transaction cannot be completed. Violation of law	1
94	Duplicate transmission	1
95	Reconcile error	1
96	System malfunction	1
97	Advises that reconciliation totals have been reset	1
98	MAC error	1
99	Reserved for national use	1
EA	*response text varies depending on reason for error*	2
EG	*response text varies depending on reason for error*	2
EM	Error at the Merchant Server level	2
N1	Unknown Error (NZ Only)	1
N2	Bank Declined Transaction (NZ Only)	1
N3	No Reply from Bank (NZ Only)	1
N4	Expired Card (NZ Only)	1
N5	Insufficient Funds (NZ Only)	1
N6	Error Communicating with Bank (NZ Only)	1
N7	Payment Server System Error (NZ Only)	1
N8	Transaction Type Not Supported (NZ Only)	1
N9	Bank declined transaction (NZ Only)	1
NA	Transaction aborted (NZ Only)	1

Code	Description	Summary Code
NC	Transaction cancelled (NZ Only)	1
ND	Deferred Transaction (NZ Only)	1
NF	3D Secure Authentication Failed (NZ Only)	1
NI	Card Security Code Failed (NZ Only)	1
NL	Transaction Locked (NZ Only)	1
NN	Cardholder is not enrolled in 3D Secure (NZ Only)	1
NP	Transaction is Pending (NZ Only)	2
NR	Retry Limits Exceeded, Transaction Not Processed (NZ Only)	1
NT	Address Verification Failed (NZ Only)	1
NU	Card Security Code Failed (NZ Only)	1
NV	Address Verification and Card Security Code Failed (NZ Only)	1
Q1	Unknown Buyer	1
Q2	Transaction Pending	2
Q3	Payment Gateway Connection Error	3
Q4	Payment Gateway Unavailable	1
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 Westpac to confirm reconciliation	2
QJ	Invalid Client Certificate	3
QK	Unknown Customer Merchant	3
QL	Business Group not configured for customer	3
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 Capture Order Number specified for Refund, Refund amount exceeds capture amount, or Previous capture was not approved	1

Code	**Description**	**Summary Code**
QW	Invalid Reference Number	1
QX	Network Error has occurred	2
QY	Card Type Not Accepted	1
QZ	Zero value transaction	0
RA	*response text varies depending on reason for rejection*	3
RG	*response text varies depending on reason for rejection*	3
RM	Rejected at the Merchant Server level	3

Appendix B - Connectivity Troubleshooting with Internet Explorer

This section describes how to use Internet Explorer to perform connectivity troubleshooting. The steps described in this appendix can be used to isolate problems if your application is unable to connect to the QuickGateway API server. These instructions were written using Internet Explorer 5.5.

In order to perform these steps, you will require the HTML Client (a file named CCAPI_support_tester.htm). This is available on request.

1) Login to the server that will be making the API call as the same user that your application will run under.

2) Save the file as a CCAPI_support_tester.htm file on the machine.

3) Close any existing Internet Explorer windows

4) Open the saved .HTM file in Internet Explorer.

5) Enter the details of the API call that you wish to perform. Ensure that the username, password and merchant are correct. For the purposes of this test, the default details will usually suffice (you can use any credit card number e.g. 4242424242424242 and the amount must be in cents eg 100 = $1.00)

6) Click "Make Payment". This will cause IE to do a POST to the QuickGateway API server.

7) If you receive "The page can not be displayed" or a DNS error then your server is unable to make a network connection to the QuickGateway API server. Check your proxy, DNS and firewall settings.

9) Internet Explorer should prompt, "The Web site you want to review requests identification. Select the certificate to use when connecting." A list of certificates will be displayed. Select your client certificate. This is the private key used to digitally sign the request from your server to the QuickGateway API server. It is validated against your server's public key which is held on the QuickGateway API server. If it does not appear in the list it is not installed correctly.

10) The QuickGateway API should respond to the QuickGateway API request. This will look like something like this: -

```
response.summaryCode=0 response.responseCode=00 response.text=Approved or
completed successfully response.referenceNo=8499901 response.orderNumber=
X12345112 response.RRN=231232 response.settlementDate=20051216
response.previousTxn=0 response.paymentInstrumentId=1
response.cardSchemeName=VISA response.creditGroup=VI/BC/MC
response.transactionDate=16-Dec-2005 14:12:11 response.end
```

If the QuickGateway API works correctly from Internet Explorer, but your application does not work correctly, check the following :-

- The proxy server settings used by your application are identical to those specified in Internet Explorer
- The POST URL used by your application is identical to those specified on the web page (NB. The web page uses the HTTPS POST interface, a different URL applies if you are using the SOAP interface).
- The client certificate (your private key) you selected using Internet Explorer is being used when sending the POST

Appendix C – Dealing with QI Responses

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

If you have followed the recommendations in section 3.3.3.4, your software should correctly handle the QI response code using one of the following options:

1. Repeatedly querying the transaction via the API at a later stage to determine the status (see section 3.3.3.4)

2. Reversing the transaction via the API to ensure that the card holder is not debited (see section 3.4.2), or

3. 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 section 3.5[4]. Depending on your business procedures, your support staff can either enter this status into your system, or contact Customer Care to void the transaction. If a transaction does not appear on these pages, the transaction was not received or processed by the QuickGateway 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 ccapi.client.qvalent.com 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 Customer Care support for assistance on **1300 726 370**. Your network administrators will need to be available to work through the problem with the Qvalent network administrators.

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

Appendix D – Glossary

CA-XCOM
CA-XCOM is a cross-platform, value-added data transport solution, providing high-performance unattended file transfer with complete audit trails and reporting. CA-XCOM provides a single solution for sending and receiving files, as well as sending reports and jobs, to a wide range of platforms. This is Qvalent's standard file transfer mechanism.

Certificate
An electronic document that identifies an entity (e.g. a person, computer or company). Each certificate contains the entity's public key, along with details about which encryption algorithms the entity can use. Certificates are issued by Certificate Authorities (CAs) when the CA verifies the entity requesting the certificate.

Each certificate contains a subject, describing who the certificate is for, and an issuer, describing the organisation that signed the certificate.

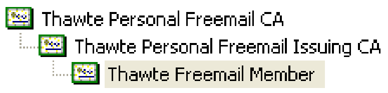
The certificate contains the entity's public key, as well as the digital signature of the CA. This signature is like a hologram on a credit card, verifying that the CA has authenticated the entity's identity.

Certificates can be marked for various purposes, including SSL client, SSL server and CA. See also *Certificate Authority*, *Digital Signature*, *SSL* and *Public Key Encryption*.

Certificate Authority
A trusted third party that signs certificates for other parties. Often in internet communications, the two parties will not trust each other, but will trust a third party. Party A can trust party B's certificate if it is signed by that third party (the certificate authority or CA).

Certain CAs (e.g. Verisign, Thawte) are automatically trusted by all certificate software. See also *Certificate* and *Certificate Hierarchy*.

Certificate Hierarchy
The chain of certificates for an entity consisting of that entity's certificate and any CAs which signed the certificate. All certificates are signed by another certificate, generating a hierarchy. This hierarchy terminates at a root certificate, which is **self-signed**. This type of certificate contains an identical issuer and subject.


Thawte Personal Freemail CA
Thawte Personal Freemail Issuing CA
Thawte Freemail Member

A certificate is trusted by a party if the certificate chain terminates at a CA which is trusted by that party. Each party maintains a list of trusted root CAs. See also *Certificate*, *Certificate Authority and Self-signing*.

Digital Signature
A process of signing a message electronically. Normally, the sender of a message will calculate a message digest, then encrypt that digest value with the sender's private key. This resulting value is the digital signature.

The receiver can verify the signature by calculating the message digest, and comparing it to the value obtained by decrypting the digital signature with the sender's public key. See also *Message Digest* and *Public Key Encryption*.

Encryption/Decryption
The process of scrambling a message so that it cannot be read by a third party while in transit. The sender encrypts a

message before sending, and the receiver decrypts the received message before reading it.

Many algorithms are available to encrypt data. Examples include RSA, RC4 and DES. The algorithm is generally well-known, but a number (called a **key**) must be used with the algorithm to produce an encrypted result or to decrypt previously encrypted information. Decryption with the correct key is simple, whereas without the key, decryption is almost impossible.

HTTP
Hypertext Transfer Protocol: The application level protocol that is used to transfer data on the web. A client sends a request message to the server, and the server sends a response message.

Each message consists of a start line (which is either a request line or a status line as appropriate), followed by a set of message headers and finally an optional message body.

The request line contains the method (usually GET or POST) used for the request. GET is a simple request for information, whereas POST allows the client to send data to the server in the request.

A web browser generally sends a GET request to the server for information, and the server responds with a HTML document in the response for the browser to display.

The HTTP protocol uses the TCP/IP protocol to transport the information between client and server. HTTP uses TCP port 80 by default. See also *TCP/IP*.

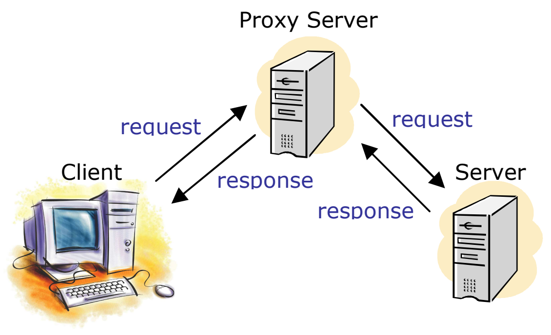
HTTPS
Hypertext Transfer Protocol, Secure: The HTTP protocol using the Secure Sockets Layer (SSL), providing encryption and non-repudiation. HTTPS uses TCP port 443 by default. See also *HTTP* and *SSL*.

Message Digest
A mathematical function which generates a number from a message (also called a one-way hash). The generated number is unique for the message, in that changing any part of the message changes the resulting number. The function is one-way in that it is, for all practical purposes, impossible to determine the message from the number. Common algorithms are MD5 and SHA-1.

Non-repudiation
Assurance the sender of data is provided with proof of delivery and the recipient is provided with proof of the sender's identity, so neither can later deny having processed the data.

Proxy Server
An intermediate server on the client side of a HTTP transaction which makes requests on behalf of the client. Proxy servers improve corporate security by only exposing the proxy server to the internet, rather than each individual computer in the organisation.

The client sends its request to the proxy server, which then sends the request (with any modifications) to the server. The server responds to the proxy, which then passes the response to the client.



System administrators can restrict which servers are accessible simply by configuring the proxy server. See also *HTTP*.

Public Key Encryption
An encryption method where different keys are used for encryption and decryption. Each party has two keys – a public key and a private key. Messages encrypted with the public key can only be decrypted with the private key, and messages encrypted with the private key can only be decrypted by with the public key. Each party publishes their public key and keeps their private key secret.

Encryption is accomplished by the sender encrypting the message with the receiver's public key. The message can then only be decrypted by the receiver with his private key.

Non-repudiation is accomplished by the sender encrypting the message with her private key. The message can then be decrypted by anyone with the sender's public key (which is published), but the receiver can be assured of the message's origin. See also *Symmetric Key Encryption* and *Encryption*.

Self-Signing
Self-signing occurs when the owner of a key uses his private key to sign his public key. Self-signing a key establishes some authenticity for the key, at least for the user IDs. The user ID of the signature must match the user ID of the key. (Where there are multiple user IDs, the ID of the signature must match the primary ID of the key.) Also, the key ID of the signature matches the key ID of the key. This verifies that whoever placed a user ID on a public key also possesses the private key and passphrase. Of course, this does not verify that the owner of the key is really who she says she is. That is done by the signatures of others on the public key (such as a root CA like Verisign).

SOAP
Simple Object Access Protocol: An XML-based protocol allowing remote procedure calls and asynchronous messaging. SOAP generally uses HTTP to transport the messages between computers. SOAP is becoming popular because of its use of standard internet protocols as its basis. See *XML* and *HTTP*.

SSL
Secure Sockets Layer: A protocol designed by Netscape to encrypt data, authenticate the client and server and ensure message integrity. SSL sits between the application layer protocol (e.g. HTTP) and above the TCP/IP network protocol.

The SSL handshake establishes the SSL connection, setting up the secure channel. In this process, the server presents its certificate to the client for authentication:

- The server encrypts some data with its private key and the client then checks this signature with the public key from the server's certificate.
- The client checks that the server DNS name is the same as that in the certificate.
- The client checks that the server certificate has not expired.
- The client checks that the server's certificate is signed by a trusted CA.

The server can also optionally require the client to present its certificate to the server for authentication.

The handshake also allows the client and server to agree on an encryption algorithm (a symmetric key algorithm for speed), and securely exchange the session key. This session key is used in the encryption algorithm which encrypts the data exchanged between the client and server after the handshake is finished. The session key length can be 40-bit, 56-bit or 128-bit, with the longer keys being more difficult to break. See also *TCP/IP*.

Symmetric Key Encryption
An encryption method where the sender and receiver use the same key to encrypt and decrypt the message. This method relies on the key being kept secret between the two parties. If the key is discovered, anyone can read the messages in transit, or send false messages to the receiver.

This type of encryption is often used for bulk encryption because it is much faster than public key encryption. See also *Encryption* and *Public Key Encryption*.

TCP/IP
Transmission Control Protocol over Internet Protocol. IP allows packets of data to be sent across the internet from one computer to another. TCP provides a reliable communication stream between the two computers, using the Internet Protocol.

XML
eXtensible Markup Language: A document formatting language which describes a standard syntax, but allowing many different document types. Business partners can then agree on the specific documents they will exchange, using the standard syntax. XML documents contain a hierarchical list of tags, some of which contain values.