

# Internet Payment Gateway

WebService API

# Table of Contents

<b>Introduction .....</b>	<b>3</b>
<b>Overview .....</b>	<b>4</b>
WebService API Endpoints .....	4
Transaction Formats .....	4
XML request example .....	4
JSON request example .....	5
<b>Transaction Parameters .....</b>	<b>6</b>
<b>Transaction Response Samples .....</b>	<b>8</b>
Transaction Response Details .....	9
Testing with cURL .....	9
XML cURL example .....	9
JSON cURL example .....	9
Transaction Response Analysis .....	10
XML response example .....	10
JSON response example .....	10
Response properties .....	11
Application Integration .....	11
<b>Troubleshooting .....</b>	<b>11</b>
Network Errors .....	11
Unexpected Errors While Testing .....	11
Common errors and solutions .....	12
<b>Authentication Tokens .....</b>	<b>13</b>
Managing Authentication tokens .....	13
<b>Additional Information .....</b>	<b>15</b>
Security .....	15
Test/Live Merchant Status .....	15

## Introduction

The St.George Bank WebService API provides application developers with a WebService interface for credit card transaction processing through the St.George Internet Payment Gateway. This document should be used in conjunction with the **Generic API Developer Guide** – located on the St.George IPG download page - <https://www.ipg.stgeorge.com.au/download.asp>

Communication between the merchant and St.George Bank WebServices API interface uses SSL connectivity.

**NOTE: If your application is located behind a firewall or proxy server, you should ensure that the relevant addresses and ports are not being filtered or blocked.**

It is important to note that the Gateway utilises two distinct transaction processing environments. These environments are defined as **LIVE** and **TEST**. The **LIVE** environment connects directly to the live banking processing system and performs real-time transactions. The **TEST** environment is a simulated environment controlled within the Gateway that effectively mimics the live processing environment; **TEST** is principally used for integration testing and training. Using the parameters provided by St.George Bank, you must define and test your application and integration against the **TEST** environment to ensure compliance before making the request to be made **LIVE**.

## Overview

The St.George Bank WebService API is a REST API which allows merchants to submit transactions for processing by sending XML or JSON via HTTP POST to a secure API endpoint.

## WebService API Endpoints

The St.George Bank WebService API environments are accessible at the following URLs.

- > Live URL: <https://www.ipg.stgeorge.com.au/WebServiceAPI/service/transaction>
- > Test URL: <https://www.ipg.stgeorge.com.au/WebServiceAPI/service/testtransaction>

### Note:

- > The test URL is designed to be used for integration testing only; it is not connected to the live banking network.
- > All transactions are submitted via a secure HTTPS protocol.

## Transaction Formats

All transaction data is submitted using a HTTP POST to a secure API endpoint. The following formats and examples are provided.

### XML request example

The following Headers **must** be set for each API request:

- > Content-Type: *application/xml*
- > Accept: *application/xml*

A Sample XML document is shown below:

```
<?xml version='1.0'?>
<transaction>
  <interface>CREDITCARD</interface>
  <transactiontype>PURCHASE</transactiontype>
  <clientid>10000000</clientid>
  <authenticationtoken>a50Dg4Z1NZq3skXq</authenticationtoken>
  <carddata>4111111111111111</carddata>
  <cardexpirydate>1213</cardexpirydate>
  <cvc2>132</cvc2>
  <totalamount>10.00</totalamount>
</transaction>
```

- > The XML document must have a root element called “*transaction*”
- > Each child node of the “*transaction*” root element then represents the required fields for a transaction.
- > Different transaction types will require different child nodes to be supplied in the XML document. These are described below in Transaction Properties.
- > The XML document is case sensitive. The “*transaction*” element and child nodes are required to be lower case

## JSON request example

The following Headers **must** be set for each API request:

- > Content-Type: *application/json*
- > Accept: *application/json*

A Sample JSON object is shown below:

```
{
  "interface": "CREDITCARD",
  "clientid": "10000008",
  "authenticationtoken": "i006vWJMCPVcCx99",
  "carddata": "4111111111111111",
  "cardexpirydate": "1222",
  "cvc2": "132",
  "totalamount": "10.00",
  "transactiontype": "PURCHASE"
}
```

- > Different transaction types will require different child nodes to be supplied in the JSON object. These are described below in Transaction Properties.

## Transaction Parameters

The following transaction parameters are used with the St.George WebService API. **Note: All parameters are case sensitive.**

Parameter	Valid values	Description
interface	CREDITCARD	Specifies the interface to be used to process the transaction.
transactiontype	PURCHASE PREAUTH COMPLETION REFUND STATUS	Specifies the type of transaction. Different transaction types have different transaction parameter requirements.
clientid	8 digits Numeric field	Client ID associated with the merchant account. This can be found on the original establishment paperwork or in the St. George Bank Merchant Administration Console
authenticationtoken	Alphanumeric field	A unique client specific value that is used to authenticate the client with the Gateway. This value can be found in the St. George Bank Merchant Administration Console
carddata	12-19 digits Numeric field	The credit card number that will be used in the transaction. The value must not contain any white space or dashes and all characters must be numerical.
cardexpirydate	4 digits Numeric field	This parameter is the expiry date of the credit card to which the transaction applies. This number must not contain any white space, dashes or slashes or any other non-numerical characters. Valid characters are numbers 0 through to 9, and must be exactly 4 characters in length. The format is MMY. Eg: A two digit month followed by a two digit year.
cvc2	3-6 characters Alphanumeric field	This value is a three or four digit number printed on a credit card, which provides additional security for online payments. Visa, MasterCard, Diners Club and JCB cards have a three digit CVC2 printed on the signature panel on the reverse side of the card. American Express cards have a four digit CVC2 printed above the card number on the front of the card.
totalamount	Decimal format	The total amount to be transacted, inclusive of any taxes. This value must be in decimal format and may contain numeric characters and a decimal point '.' character, for example: 20.00 for a \$20.00 transaction, 3099.95 for a transaction to the value of \$3099.95. Do not include the dollar symbol '\$'. NOTE: Using the test environment the RESPONSECODE returned may be varied by selecting a different cents value in this field. It is essential that you test for all responses and handle each appropriately within the merchant application.
currency	Alpha Currency Code	The CURRENCY field allows the input of a foreign currency. The St. George Gateway only supports Australian currency (AUD). The field is optional and, if not present, it will default to AUD.

txnreference	Numeric field	This parameter applies only to STATUS transaction types. It is used to hold the unique transaction reference for the original transaction for which the STATUS is intended.
originaltxnref	Numeric field	This parameter applies to REFUND transaction types. It is used to hold the unique transaction reference for the original transaction. (The value must correlate to a transaction originally processed by the merchant account)
preauthnumber	Numeric field	This parameter applies to COMPLETION transaction types. It is used to hold the unique transaction reference for the original PREAUTH transaction. (The value must correlate to a transaction originally processed by the merchant account)
clientref	0-50 Characters Alphanumeric field	This parameter can be used by merchants for storing reference numbers or data. Its use is optional, but is primarily used for transaction reconciliation purposes. Values larger than the specified character limit will be trimmed when stored in the St. George Gateway.
comment	0-255 Characters Alphanumeric field	Merchants can use this optional free-text field for storing additional data to help with transaction reconciliation. Values larger than the specified character limit will be trimmed when stored in the St. George Gateway.

## Transaction Response Samples

The St.George WebService returns an XML or JSON response. A Sample response is shown below:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<transaction>
  <carddata>456445XXXXXX4564</carddata>
  <clientid>10000001</clientid>
  <interface>CREDITCARD</interface>
  <responsecode>00</responsecode>
  <responsetext>APPROVED</responsetext>
  <settlementdate>2012-02-28 00:00:00</settlementdate>
  <stan>619829</stan>
  <totalamount>10.00</totalamount>
  <transactiontype>PURCHASE</transactiontype>
  <txnreference>223000000036285</txnreference>
</transaction>
```

```
{
  "theinterface": "CREDITCARD",
  "transactiontype": "PURCHASE",
  "totalamount": "10.00",
  "taxamount": null,
  "carddata": "411111XXXXX1111",
  "cardexpirydate": null,
  "currency": null,
  "txnreference": "2230000003103928",
  "originaltxnref": null,
  "authcode": null,
  "clientref": null,
  "clientid": "10000008",
  "comment": null,
  "terminaltype": null,
  "cvc2": null,
  "merchantip": null,
  "authenticationtoken": null,
  "preauthnumber": null,
  "authorisationnumber": null,
  "error": null,
  "responsecode": "00",
  "responsetext": "APPROVED (TEST TRANSACTION ONLY)",
  "stan": "431845",
  "settlementdate": "2018-05-03 00:00:00",
  "customerip": null,
  "interface": "CREDITCARD"
}
```

- > Please note that an error field will also be returned if there was a problem processing the transaction



## Transaction Response Details

The following transaction responses are returned by the St. George Gateway:

Response	Description
responsecode	<p>A two digit code used to determine transaction success or failure. The code number matches the response text field.</p> <p>“00”, “08” or “77” designate an <b>Approved</b> transaction.</p> <p>Response codes are contained in The <i>Transactions Specifications</i> documents that accompany this kit.</p> <p>A response code &lt; 0 means that a request was not processed properly and should be tried again. Users will need to consult the logs to determine what the error was that caused the transaction to fail.</p>
responsetext	A meaningful text message that explains the response code
error	A field outlining additional details as to why a transaction may have been declined.
txnreference	A unique reference number generated for each transaction.
stan	The Sequence Trace Audit Number (STAN) is an identifier assigned to the transaction. It is helpful to supply this value to the St. George Helpdesk with any transaction enquiries.
settlementdate	SETTLEMENTDATE is the settlement date of the transaction. Eg: the date the actual funds transfer will occur.
carddata	Masked version of the card number used to do the transaction. The value will be the first 6 and last 4 digits of a card number.
clientid	The Client ID associated with the merchant account which processed the transaction.
interface	The interface used to do the transaction.

### Testing with cURL

While the St.George WebService API interface is designed to be programmatically integrated into your application, initial testing can be done using cURL. For details see:

<http://en.wikipedia.org/wiki/CURL>

Once you have downloaded the cURL application you can start testing transactions.

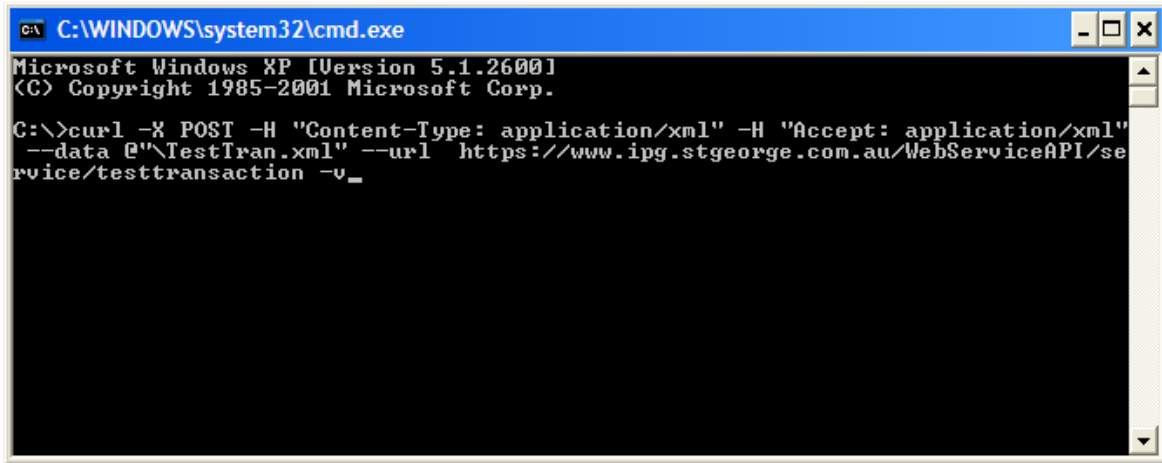
#### Sample cURL command:

#### XML cURL example

```
curl -X POST -H "Content-Type: application/xml" -H "Accept: application/xml" --data
@"\TestTran.xml" --url
https://www.ipg.stgeorge.com.au/WebServiceAPI/service/testtransaction -v
```

#### JSON cURL example

```
curl -X POST -H "Content-Type: application/json" -H "Accept: application/json" --data
@"\TestTran.json" --url
https://www.ipg.stgeorge.com.au/WebServiceAPI/service/testtransaction -v
```



**Explanation of command:**

Parameter	Description
Curl	The cURL executable.
-X POST	Transaction must be submitted as a HTTP POST
-H "Content-Type: <i>application/xml</i> " Or -H "Content-Type: <i>application/json</i> "	Set <i>Content-Type</i> header to <i>application/xml</i> or <i>application/json</i>
-H "Accept: <i>application/xml</i> " Or -H "Accept: <i>application/json</i> "	Set Accept header to <i>application/xml</i> or <i>application/json</i>
--data @\TestTran.xml Or --data @\TestTran.json	The data to be posted. This example loads the data from a file (using the "@" indicator) called TestTran.xml/ TestTran.json, you can also place the transaction details inline.
--url https://www.ipg.stgeorge.com.au/WebServiceAPI/service/testtransaction	The URL where the data will be sent
-v	Optional: Verbose output enabled

**Transaction Response Analysis**

**XML response example**

The response from cURL will be an XML document with a transaction root element. The following XML nodes should be reviewed to determine the result of the transaction.

- `<responsecode>00</responsecode>`
- `<responsetext>Approved (TEST TRANSACTION ONLY)</responsetext>`
- `<txnreference>2230000012345678</txnreference>`

If the transaction is in error there will also be an Error node.

- `<error>Required Field Client ID not set</error>`

**JSON response example**

The response from cURL will be a JSON object. The following keys should be reviewed to determine the result of the transaction.

- `responsecode: 00`
- `responsetext: Approved (TEST TRANSACTION ONLY)`
- `txnreference: 2230000012345678`

If the transaction has an error, there will also be an Error key with a message describing the error.

- `error: Required Field Client ID not set`

## Response properties

Response	Description
responsecode	A two digit code used to determine success or failure. The code number matches the response text field. “00”, “08” or “77” designate an <b>Approved</b> transaction. Response codes are contained in The <i>Transactions Specifications</i> documents that accompany this kit. A response code < 0 means that a request was not processed properly and should be tried again. Users will need to consult the logs to determine what the error was that caused the transaction to fail.
responsetext	A meaningful text message that explains the response code
error	Any errors received
txnreference	A unique reference number generated for each transaction.

## Application Integration

The St.George WebService API is designed to be integrated into your existing application environment using your application native support for REST WebServices. Programming languages such as Java, PHP, .Net, jQuery and more provide support for WebServices.

## Troubleshooting

If your output fails any of the tests above, please review the steps you have taken so far and ask yourself the following questions:

- Are you using the correct Client ID?
- Are you using the correct AuthenticationToken?
- Is the correct URL specified for the St.George Gateway? You may want to use “nslookup” or “dig (domain information groper)” to ensure that the address resolves correctly.
- Are you behind a firewall? You may need to contact your Security Administrator to ensure the appropriate ports have been defined through the firewall.

If you still can't identify the problem, please email the St.George IPG support email on: [ipgsupport@stgeorge.com.au](mailto:ipgsupport@stgeorge.com.au). Be prepared to e-mail the output of a cURL test.

## Network Errors

Commercial and freeware tools can be found on the Internet to assist you in testing your network connectivity to the St.George WebService server.

## Unexpected Errors While Testing

In some cases, the system may return errors when you are expecting an approved or other code during testing. Please check the response code you receive against those contained in The *Transactions Specifications* documents accompanying this kit.

Some response codes may be generated at any time and this behavior is perfectly normal. For example, if the system is being subjected to load testing the code 'A6' for 'Server Busy' may be expected.

Before reporting errors, ensure that you are able to replicate them, so that our support team can diagnose properly.

## Common errors and solutions

The following section outlines some common errors and their solutions.

Error Message	Comment
HTTP Status 405 - Request method 'GET' not supported	Transactions must be submitted using a HTTP POST
HTTP Status 404 The requested resource () is not available.	<p>The message can occur for a number of reasons. Check:</p> <p>The URL you have configured is incorrect</p> <p>The "Accept" Header has not been set correctly. The accept "Accept" should be set to: <i>"application/xml"</i> or <i>"application/json"</i></p>
HTTP Status 415 - The server refused this request because the request entity is in a format not supported by the requested resource for the requested method ()	Check that the <i>Content-Type</i> header has been set correctly. Content type should be set to: <i>"application/xml"</i> or <i>"application/json"</i>
HTTP Status 400 - The request sent by the client was syntactically incorrect ()	<p>The data sent to the server was incorrectly formatted.</p> <p>Check the transaction data being sent is valid.</p>
INVALID INPUT DATA	<p>Check the Error response field to determine which data is invalid:</p> <p>Required Field Interface not set</p> <p>Required Field Client ID not set</p> <p>Required Field AUTHENTICATIONTOKEN not set</p> <p>Missing required field for Refund transaction: ORIGINALTXNREF</p>
AUTHENTICATION FAILURE	AUTHENTICATIONTOKEN value is invalid for this client ID
UNABLE TO PROCESS	Check the Error response field to determine the cause of the problem.

## Authentication Tokens

Each API call will need the *authenticationtoken* value set to a valid authentication set within the St.George Merchant Admin Console system.

You can add, delete and invalidate Authentication tokens via the St.George Merchant Admin Console interface: <https://www.ipg.stgeorge.com.au/merchants>

Once logged in navigate to: Merchant Details > IPG Accounts (Client ID) > Authentication (tab) > Configure

### Managing Authentication tokens

You can generate a new token using the “Generate” button highlighted below:

#### Client Authentication Tokens

Client Authentication Tokens can be configured to provide enhanced security for online payments. Simply include a token generated below into the transaction bundle using the key AUTHENTICATIONTOKEN.

Once a token is generated the status is by default inactive. From the moment the token is generated all transactions from this client ID will enforce that a token is present before processing a transaction. To stop token authentication remove all tokens.

The system only allows two tokens to be generated. Only one can exist without an expiry date. After creating a new token an expiry date should be assigned to the old token. The new token can then be set to active.

<a href="#">← Back</a>	<a href="#">Refresh List</a>	
<b>Current Tokens</b>		
Token	Expiry Date (dd/mm/yyyy)	Active
		<a href="#">Generate</a>



Once generated the token will need to be activated by clicking on the “Edit” button highlighted below:

#### Client Authentication Tokens

Client Authentication Tokens can be configured to provide enhanced security for online payments. Simply include a token generated below into the transaction bundle using the key AUTHENTICATIONTOKEN.

Once a token is generated the status is by default inactive. From the moment the token is generated all transactions from this client ID will enforce that a token is present before processing a transaction. To stop token authentication remove all tokens.

The system only allows two tokens to be generated. Only one can exist without an expiry date. After creating a new token an expiry date should be assigned to the old token. The new token can then be set to active.

<a href="#">← Back</a>	<a href="#">Refresh List</a>	
<b>Current Tokens</b>		
Token	Expiry Date (dd/mm/yyyy)	Active
  097SbY34fo6Qo35u		<a href="#">false</a>
		<a href="#">Generate</a>

The token can then be set to Active = True and an optional Expiry Date set for the token. Once finished editing, save the changes with the “Update” button.

### Client Authentication Tokens

Client Authentication Tokens can be configured to provide enhanced security for online payments. Simply include a token generated below into the transaction bundle using the key AUTHENTICATIONTOKEN.

Once a token is generated the status is by default inactive. From the moment the token is generated all transactions from this client ID will enforce that a token is present before processing a transaction. To stop token authentication remove all tokens.

The system only allows two tokens to be generated. Only one can exist without an expiry date. After creating a new token an expiry date should be assigned to the old token. The new token can then be set to active.

<a href="#">← Back</a>	<a href="#">Refresh List</a>	
<b>Current Tokens</b>		
Token	Expiry Date (dd/mm/yyyy)	Active
097SbY34fo6Qo35u	<input type="text"/>	<span style="background-color: yellow;">false</span> ▼
		<input type="button" value="Cancel"/> <input type="button" value="Update"/>



Your token is now set. It can be deleted at any time using the “Delete” button highlighted below:

### Client Authentication Tokens

Client Authentication Tokens can be configured to provide enhanced security for online payments. Simply include a token generated below into the transaction bundle using the key AUTHENTICATIONTOKEN.

Once a token is generated the status is by default inactive. From the moment the token is generated all transactions from this client ID will enforce that a token is present before processing a transaction. To stop token authentication remove all tokens.

The system only allows two tokens to be generated. Only one can exist without an expiry date. After creating a new token an expiry date should be assigned to the old token. The new token can then be set to active.

<a href="#">← Back</a>	<a href="#">Refresh List</a>		
<b>Current Tokens</b>			
 	Token	Expiry Date (dd/mm/yyyy)	Active
	097SbY34fo6Qo35u		true
			<input type="button" value="Generate"/>

## Additional Information

### Security

The St.George Bank Gateway maintains secure transactions through the use of 256 bit SSL (Secure Sockets Layer). The SSL Handshake Protocol was originally developed by Netscape Communications Corporation to provide security and privacy over the Internet. Each transaction is secured with an AuthenticationToken which is unique to each merchant and can be regenerated using the Merchant Administration Console.

### Test/Live Merchant Status

St. George Bank provides a Test and a Live system for merchant use. The Test system is a mirror of the Live system, with the exception that it sends the requests to a *simulated* banking environment.

Initially, you will need to develop and test your code using the test system. When you believe you are ready to “Go Live”, you complete the Test checklist which is provided with your welcome email to begin the conversion to Live process.

The St.George Gateway produce a wide range of transactional responses. Within the testing phase, these responses are controllable through the allocation of the cents component in the amount value. To fully test your system’s ability to handle and control varying responses, you should test all cents values between 00 and 99.

A complete list of all response codes can be downloaded from the St.George IPG download page here: <https://www.ipg.stgeorge.com.au/download.asp>