

Technical Guide

Bancstac Paycenter Web



Table Of Contents

Table Of Contents	2
1. Introduction to Paycenter Web	3
2. Reference points for the intended audience	3
3. Paycenter Web features	3
4. Payment Configuration Options	5
4.1.Embedded Iframe	5
Why use an Iframe?	5
4.2 . Prepopulated Hosted Redirect Page (Prepopulated Hosted Payment Page)	6
Why Use a Prepopulated Hosted Redirect?	6
4.3. Open Field Hosted Redirect Page (OpenField Hosted Payment Page)	6
Why use an Open field Hosted Redirect?	6
4.4. Form Post	6
Why use a Form post?	6
5. Add-ons and Additional Features	7
5.1. Tokenization	7
5.2. 3D Secure	8
Use with Any payment method	8
Use with: Iframes or Hosted Pages	8
6. Transaction Processing	9
6.1. Iframe Page	9
6.2.Hosted Page	9
6.3.Form post	9
6.4.Payment Workflow	10
6.5. Payment Initialization message structure	11
6.6.Payment Init Request Fields	11
6.7. Payment init Response fields	13
6.8. Form Post Data Fields	14
6.9.Payment Complete Message Structure	14
6.9.1 Sample Payment_complete request generated by the API for you	14
6.10.PAYMENT_COMPLETE Response	15
6.10.1 Sample payment_Complete Response	16
7. Tokenization	17
7.1. Storing a card and obtaining a token	17
7.1.2. Performing a purchase and obtaining tokens simultaneously	18
7.1.3. Tokenization Responses	19
8. Making Payments Using Tokens	20
Support and Troubleshooting	21

1. Introduction to Paycenter Web

This guide provides technical information about integrating and configuring Bancstac's Paycenter Web Solution to your website or application. Bancstac Paycenter Web allows merchants to seamlessly integrate and process credit card payments in a PCI DSS-compliant manner.

2. Reference points for the intended audience

This guide is intended to act as a reference point for web/application developers seeking to implement Bancstac transaction functionality into merchant websites and for web applications using the Bancstac API for Bancstac's REST web services.

Our API will generate the JSON messages for you and handle the marshaling and unmarshaling of JSON messages so that the merchant does not have to spend time writing code for these tasks. Our sample codes are currently available in Java, .NET, and PHP. However, if you are using any other languages you may directly integrate into our REST web services.

It is assumed that the parties implementing this product will be experienced in the development and implementation of websites and web applications, this document is written principally for parties with such expertise. Therefore in particular you should understand:

- JSON-based Web Services
- HTTPS protocols

3. Paycenter Web features

- Fully customized, responsive payment pages.
- Card tokenization can be enabled here for token creation.
- Sample code provided for easy integration resources for your reference.

When setting up an instance of a Payment Page, you will be provided with the following information & resources;

- This Documentation
- Client ID
- Authtoken
- HMAC Secret
- End Point
- Response codes
- Sample Cards

We will require the following information from you:

- The location of any CSS you want to be applied to an Iframe or Hosted Page by default, an Advanced Payment Page instance is set up with the following configuration;

Parameter	Value
Permitted Card Types	Visa, MasterCard, UnionPay International, American Express, Discover Network, and Diners Club
Page Type	Iframe
Transaction Type	Purchase
CVV Required	True
Tokenize	True
3D Secure	Enabled
Surcharging	Disabled

PLEASE ENSURE YOU USE THE CORRECT CLIENT ID, AUTHTOKEN, HMAC SECRET AND ENDPOINT AS PROVIDED TO YOU BY BANCSTAC.

4. Payment Configuration Options

The Bancstac Paycenter Web supports a wide variety of configurations to suit most merchant implementations. Review the below options to consider which implementation style suits your requirements.

Before you do the integration, please check that the server is having TLS 1.2 protocol enabled and having the compatible curl version 7.3.0 or above.

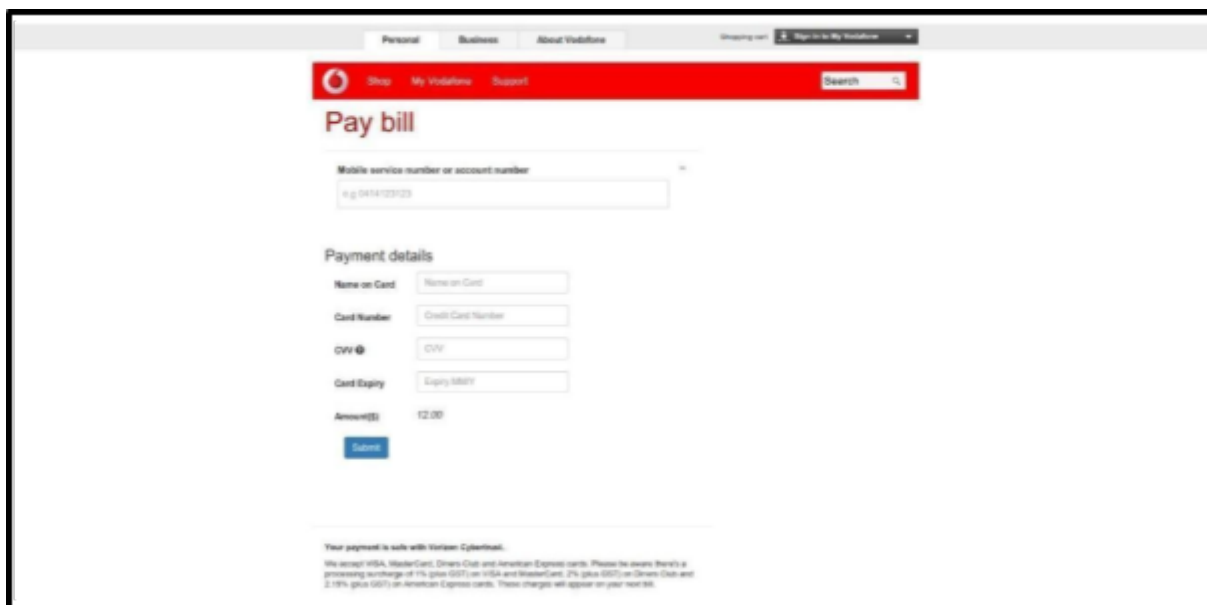
4.1. Embedded Iframe

Why use an Iframe?

Iframes are a good blend of security and compliance while causing minimal disruption to page layout. Iframe based payments are compliant according to PCI DSS, and can significantly reduce scope for required merchant side PCI compliance, unlike a Form Post or API. Iframe is the only method to keep your web service out of scope while maintaining a seamless appearance to your payment page.

Bancstac's Iframe option is highly customizable, you can deploy your own CSS and even HTML to fully customize the page.

Example page with embedded Iframe



The screenshot displays a web interface for paying a bill. At the top, there is a navigation bar with links for 'Personal', 'Business', and 'About Vodafone'. Below this is a red header with the Vodafone logo and a search bar. The main heading is 'Pay bill'. A text input field is labeled 'Mobile service number or account number' with a placeholder 'e.g. 0474 123123'. Below this, the 'Payment details' section contains several input fields: 'Name on Card', 'Card Number', 'CVV', and 'Card Expiry'. The 'Amount' is displayed as '12.00'. A blue 'Submit' button is located below the payment details. At the bottom, there is a disclaimer: 'Your payment is safe with Vodafone CyberShield. We accept VISA, MasterCard, Diners Club and American Express cards. Please be aware there's a processing surcharge of 1% plus GST on VISA and MasterCard, 2% plus GST on Diners Club and 2.15% plus GST on American Express cards. These charges will appear on your next bill.'

4.2. Prepopulated Hosted Redirect Page (Prepopulated Hosted Payment Page)

Why Use a Prepopulated Hosted Redirect?

The Prepopulated Hosted Redirect brings the user off the merchant website and onto a discrete secure payment page hosted by Bancstac. The Prepopulated Hosted Redirect has many of the same advantages of the Iframe, being highly secure, fully PCI compliant and customizable using CSS and HTML. This can be a good option for small merchants wishing to leverage the trust and brand visibility of the bank and Bancstac.

4.3. Open Field Hosted Redirect Page (OpenField Hosted Payment Page)

Why use an Open field Hosted Redirect?

The Difference between the Open Field Hosted Redirect page and the prepopulated one is that the metadata fields remain open to be entered by the cardholder. This can be a simpler implementation for small merchants. This is also useful where you need the cardholder to enter information about their payment themselves.

4.4. Form Post

Why use a Form post?

This option can only be used by merchants that have obtained PCI DSS compliant certification. The Form Post gives the merchant the most control over the customer experience when making their payment. However, this comes at the significant cost of bringing the entire web server, and its associated devices and servers on the same network into PCI DSS scope.

5. Add Ons and Additional Features

In addition to the fundamental payment process, the Advanced Payment page also allows for a number of separate features.

5.1. Tokenization

In addition to processing payments directly, Bancstac Paycenter Web includes a tokenization service which enables you to securely keep credit card details in Bancstac's encrypted token vault ("Secure CardVault Service"). Here the card is stored in Bancstac's encrypted PCI compliant servers, we issue the merchant a code called a token which represents the stored card.

Bancstac Paycenter Web can be used for storing cards with Bancstac's Secure CardVault Service and obtaining tokens. However in order to process tokens you must use Real Time payments .

The Bancstac tokens are numeric non-mod10 values which retain the same first 6 and last 4 digits of the credit card they replace. There is a 1 to 1 relationship between the token and the card. This enables the merchant to use the token in the same fields and locations as a credit card, displaying a masked token will look the same as a card and they will fit in the same database fields as a real card.

We will discuss tokenization and making payments using tokens in detail in section 08.

5.2. 3D Secure

Use with: Any payment method

3D Secure is an additional security layer which allows the issuer or the card holder to check that the transaction about to take place isn't fraudulent before the actual payment takes place. The other major advantage of 3D Secure is that the chargeback liability can be shifted away from the merchant and back to the issuer. This is a great tool for high-risk payments or where security is paramount. 3D Secure is enabled by default to ensure merchant safety. In some instances 3D Secure introduces an additional layer of friction to complete a transaction during the checkout process.

Please speak with your bank representative or your Bancstac implementation agent to obtain more information about how enabling 3D Secure can help your business

5.3. Hosted HTML & CSS

Use with: Iframes or Hosted Pages

One of the key advantages of Bancstac PCI compliant payment solutions is that they remain PCI compliant whilst being highly customizable and integrate seamlessly into your own website.

CSS: The Bancstac page can reference an externally available CSS file hosted in the merchant environment. This way there is no need to contact Bancstac for any minor CSS updates.

HTML: Bancstac can also host a HTML template file to further customize the Iframe or Hosted page, ensuring the page fits within your style guides and branding. For security purposes, this is hosted by Bancstac and can be updated easily by merchants.

6. Transaction Processing

Bancstac advanced payment pages offer three technical methods of processing transactions for merchants.

6.1. Iframe Page

Merchants will embed a Bancstac hosted payment page as an Iframe to their website. Merchants can host CSS for Iframe, so they have full control over the look and feel of the page. A merchant may also supply Bancstac with a HTML template to improve page seamlessness.

6.2. Hosted Page

Merchants will redirect the payer to a Bancstac hosted page. Bancstac URL will be visible on the page. Like the Iframe page, this method also enables merchants to have full control over the look and feel of the page.

6.3. Form post

Merchants who are PCI DSS certified can manage their own HTML and post payer's card information to Bancstac.

Note: We recommend the Iframe or Hosted solution for most merchants

6.4. Payment Workflow

The page is a highly customizable PCI compliant Payment that uses JSON based web service calls and one HTTPS call in order to complete the Payment Request.

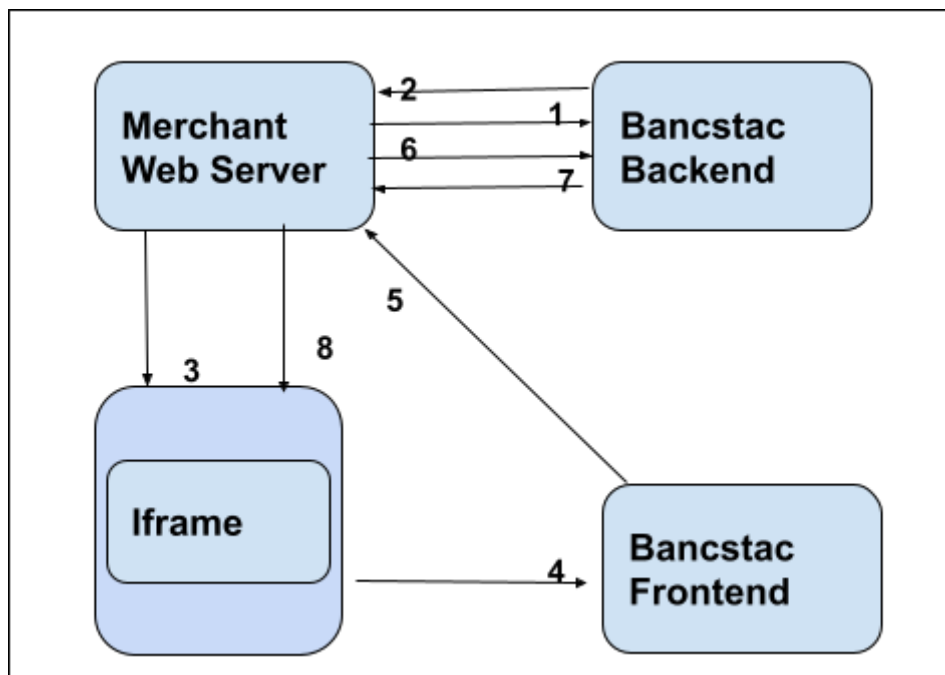
The **First** call is the ***Payment Initialization***.

The response to this message is the **Paymentinit Response**. This response contains the **paymentPageUrl**. This is the URL you would embed in the Iframe or redirect to in the case of a redirect method (Hosted Payment Pages). For the Iframe option you will embed the URL as follows,

```
<iframe src="value of the paymentPageUrl goes here "></iframe>
```

After the card data is entered by the customer in the capture page and submitted, the merchant will receive a response to the receipt URL specified in the payment initialization request '**Redirect**' parameter. This response will be via HTTP GET with a parameter called ReqID.

The **Second Call** is the **Payment Complete call**. This call is triggered to submit the full bundle comprising the first two calls to the merchant's acquiring Bank. The merchant submits the ReqID and is returned with a full transaction response, including any/all metadata sent in the first Call, the token, if it was requested, sanitized version of the cardholder data (truncated card, expiry date, no CVC). At this point the payment is complete.



1. Merchant submit Payment_Init call
2. Bancstac responds back with ReqID & Payment URL.
3. The Merchant's web server delivers the parent page to the browser, including the Payment URL embedded as an Iframe.
4. Cardholder completes payment, cardholder data captured by Bancstac.
5. Response returned to receipt URL specified in Call one.
6. Merchant submits Payment_Complete call.
7. Bancstac confirms payment processed and returned full transaction bundle, including token if any.
8. Merchant's web server serves browsers with a receipt page payment Initialization message structure.

6.5. Payment initialization message structure

This method initializes a payment and returns a unique request ID, Operation: PAYMENT_INIT

6.6.Payment init request fields

Property	Type	Description
clientID	Integer	ClientId provided by Bancstac required.
type	String	Options: PURCHASE, AUTHORISATION, TOKEN Default: PURCHASE
tokenize	Boolean	Flag to tokenize cards with PURCHASE and AUTHORISATION. If you only want to tokenize use type as TOKEN.
tokenReference	String	Reference used in conjunction with a tokenization request..
amount	Object	Transaction amount and currency. Payment Amount: Type decimal currency: 3 char ISO code Required. Amount / Currency code - ISO LKR USD
redirect	Object	Set of redirect URLs you provide for Bancstac to redirect payers to the merchant website. returnUrl: The payer is redirected to this URL after submitting the payment. ReturnMethod:
clientRef	String (50 Chars Max)	Merchant specified field
comment	String (100 Chars Max)	Merchant specified field
extraData	Map(100 Chars Max)	Merchant specified key value pair metadata.

Sample Payment_INIT Request generated by the API for you

```

POST /paycorp-webservice/InterfaceServlet HTTP/1.1
Host:https://paycorp-dfcc.prod.aws.paycorp.lk/rest/service/proxy
AUTHTOKEN: 2167847f-0b16-4a53-a3da-ee3118fded42
Content-Type: application/json
Cache-Control: no-cache
{
  "version": "1.5",
  "msgId": "AD32B8FC-0D72-41D3-8F6B-51FB2107835E",
  "operation": "PAYMENT_INIT",
  "requestDate": "2023-11-10T07:50:43.920+0530",
  "validateOnly": false,
  "requestData": {
    "clientId": "10000001", "clientIdHash":
    "", "transactionType": "PURCHASE",
    "transactionAmount": {
      "totalAmount": 0,
      "paymentAmount": 200,
      "serviceFeeAmount": 0,
      "currency": "LKR"
    },
    "redirect": {
      "returnUrl": "cancelUrl": "",
      "returnMethod": "GET"
    },
    "clientRef": "",
    "comment": "",
    "tokenize": false,
    "cssLocation1": "",
    "cssLocation2": "",
    "useReliability": true, "extraData ":
    "{1,2,3}"
  }
}

```

Note: To get the successfull response in the test environment you need to put at least 200(2.00) (your currency) as PaymentAmount. Decimal values are not supported as payment amounts in the testing environment.

For example: Use amount values such as: 200, 300, 5000, etc.

6.7. Payment_INIT Response Fields

Property	Type	Description
reqid	Integer	A unique payment id provided by Bancstac.
expireAt	String	Time at which this request will expire. Currently valid for 30 mins.
paymentPageUrl	Object	Url to get Iframe and hosted payment pages, not used for form post solutions.

Sample Payment_INIT Response raw JSON message

```
{
  "msgId": "AD32B8FC-0D72-41D3-8F6B-51FB2107835E",
  "responseData": {
    "reqid": "NGt15m5NTrORp58XS4NW",
    "expireAt": "2023-11-10T09:14:18.839+0000",
    "paymentPageUrl":
"https://paycorp-dfcc.prod.aws.paycorp.lk/webinterface/app/payment?reqid=NGt15m5NTrORp58XS4NW"
  }
}
```

6.8. Form Post Data Fields

The form post method is only for merchants who are PCI DSS certified compliant. Fields for form post request.

After submitting, Bancstac will store payment details and redirect payer to return URL using return method as provided in payment init request. Only parameters which will pass in the return URL will be "reqid".

Example: `http://some_merchant.com/confirm?reqid= 5bDqiptxShatUDb5kzIY`

6.9.Payment Complete Message Structure

Completes payment and returns final transaction response. Operation: PAYMENT_COMPLETE

Payment Complete Request Fields

Property	Type	Description
clientId	Integer	ClientId provided by Bancstac Required.
reqid	String	Request Id provided payment_init request.

6.9.1 Sample Payment_Complete request generated by the API for you

```
{
  "version": "1.5",
  "operation": "PAYMENT_COMPLETE",
  "msgId": "AD32B8FC-0D72-41D3-8F6B-51FB2107835E",
  "requestDate": "2023-11-10T13:55:45.920+0530",
  "validateOnly": false,
  "requestData": {
    "clientId": "10000003",
    "reqid": "fu7Qj0gaSYG2MHObq7WE"
  }
}
```

6.10.PAYMENT_COMPLETE Response

Property	Type	Description
clientId	Integer	ClientId provided by Bancstac.
type	String	Transaction type as provided in payment init request
tokenize	Boolean	Flag to tokenize card with PURCHASE and AUTHORISATION transaction type
tokenReference	String	
creditCard	object	Credit Card info submitted by payer type holderName number expiry
amount	object	Transaction amount in CENTS and currency type as submitted by merchant in payment init request. E.g: initRequest.transactionAmount = new TransactionAmount(1010,"AUD"); Please see currency codes given below.
clientRef	String	Merchant specified field
comment	String	Merchant specified field
extraData	String	Merchant specified key value pair metadata.
txnReference	Integer	Response Field - Bancstac transaction reference
responseCode	String	Response Field - Bank Response Code
responseText	String	Response Field - Bank Response Test

6.10.1 Sample payment_Complete Response

```
{
  "msgId": "AD32B8FC-0D72-41D3-8F6B-51FB2107835E",
  "responseData": {
    "clientId": 10000003,
    "transactionType": "PURCHASE",
    "creditCard": {
      "type": "MASTERCARD",
      "holderName": "Kavindya",
      "number": "535927****1606",
      "expiry": "0828"
    },
    "transactionAmount": {
      "totalAmount": 0,
      "paymentAmount": 200,
      "serviceFeeAmount": 0,
      "withholdingAmount": 0,
      "currency": "LKR"
    },
    "clientRef": "Test_321",
    "extraData": {},
    "txnReference": "2023200000001395",
    "responseCode": "00",
    "responseText": "Transaction Approved",
    "tokenized": false,
    "cvcResponse": ""
  }
}
```

7. Tokenization

7.1. Storing a card and obtaining a token

7.1.1. Tokenization Request

- JAVA

```
PaymentInitRequest initRequest = new PaymentInitRequest();
initRequest.setClientId(10000000);
initRequest.setTransactionType(TransactionType.TOKEN);
initRequest.setTransactionAmount(new TransactionAmount(0));
initRequest.setClientRef("merchant_token_reference");
initRequest.setRedirect(new Redirect("https://merchant.com"));
PaymentInitResponse paymentInitResponse = client.payment().init(initRequest);
```

- .NET

```
PaymentInitRequest initRequest = new PaymentInitRequest();
initRequest.clientId = 10000000;
initRequest.transactionType = Enums.TransactionType.TOKEN.ToString();
// $10.10 dollars
TransactionAmount transactionAmount = new TransactionAmount();
initRequest.transactionAmount = transactionAmount;
initRequest.clientRef = "merchant_token_reference";
Redirect redirect = new Redirect("https://merchant.com");
initRequest.redirect = redirect;
PaymentInitResponse paymentInitResponse = client.payment.init(initRequest);
```

- PHP

```
initRequest = new PaymentInitRequest();
initRequest->setClientId(10000000);
initRequest->setTransactionType(TransactionType::$TOKEN);
// $10.10 dollars
$transactionAmount = new TransactionAmount(0);
initRequest->setTransactionAmount($transactionAmount);
initRequest->setClientRef("merchant_token_reference");
$redirect = new Redirect("https://merchant.com");
initRequest->setRedirect($redirect);
$initResponse = $client->getPayment()->init(initRequest);
```


7.1.2. Performing a purchase and obtaining token simultaneously

PURCHASE with tokenize Request

- JAVA

```
PaymentInitRequest initRequest = new PaymentInitRequest();
initRequest.setClientId(10000000);
initRequest.setTransactionType(TransactionType.PURCHASE);
// $10.10 dollars as the amount must be entered in cents
// "AUD" is the currency type . If this is not set then the default currency based
// on the country of the acquiring bank will apply
initRequest.setTransactionAmount(new TransactionAmount(1010,"AUD"));
initRequest.setClientRef("merchant_payment_reference");
initRequest.setRedirect(new Redirect("https://merchant.com"));
initRequest.setTokenize(true);
initRequest.setTokenReference("merchant_token_reference");
PaymentInitResponse paymentInitResponse = client.payment.init(initRequest);
```

- .NET

```
PaymentInitRequest initRequest = new PaymentInitRequest();
initRequest.clientId = 10000000;
initRequest.transactionType = Enums.TransactionType.PURCHASE.ToString();
// $10.10 dollars as the amount must be entered in cents
// "AUD" is the currency type . If this is not set then the default currency based
// on the country of the acquiring bank will apply
initRequest.transactionAmount = new TransactionAmount(1010,"AUD");
initRequest.clientRef = "merchant_reference";
initRequest.comment = "merchant_additional_data";
initRequest.redirect = new Redirect("https://merchant.com");
initRequest.tokenize = true;
initRequest.tokenReference = "merchant_token_reference";
PaymentInitResponse initResponse = client.payment.init(initRequest);
```

- PHP

```
initRequest = new PaymentInitRequest();
initRequest->setClientId(10000000);
initRequest->setTransactionType(TransactionType::$PURCHASE);
// $10.10 dollars as the amount must be entered in cents
// "AUD" is the currency type . If this is not set then the default currency based
// on the country of the acquiring bank will apply
$transactionAmount = new TransactionAmount(1010,"AUD");
initRequest->setTransactionAmount($transactionAmount);
initRequest->setClientRef("merchant_payment_reference");
$redirect = new Redirect("https://merchant.com");
initRequest->setRedirect($redirect);
initRequest->setTokenize(TRUE);
initRequest->setTokenReference("merchant_token_reference");
initRequest->$client->getPayment()->init($paymentInitRequest);
```

7.1.3. Tokenization Responses

Successful Responses

The following responses can be returned upon successful token generation.

Response Code	Response Text
00	SUCCESS
01	TOKEN ALREADY EXISTS IN DATABASE - CARD EXPIRY UPDATED.
02	PREAUTH TRANSACTION FAILED, HOWEVER TOKEN WAS STILL GENERATED.

Unsuccessful Responses

The following responses indicate a non-successful result.

Response Code	Response Text
20	INSUFFICIENT PARAMETERS SUPPLIED TO PERFORM FORM SUBMISSION
21	INTERNAL ERROR; PLEASE CONTACT SUPPORT.
22	INVALID CLIENTID USED WITH PROVIDED SSL CLIENT CERTIFICATE.
23	UNABLE TO FIND AND DELETE STORED CARD WITH SUPPLIED TOKEN.
24	UNABLE TO UPDATE STORED CARD - INVALID EXPIRY OR FAILED PREAUTH.
25	UNABLE TO FIND AND UPDATE STORED CARD WITH SUPPLIED TOKEN
26	UNABLE TO FIND STORED CARD WITH SUPPLIED TOKEN
27	PREAUTH TRANSACTION FAILED, NO TOKEN GENERATED
28	ACTION NOT PERMITTED FOR SUPPLIED CLIENTID.
29	TOO MANY RETRIEVAL ATTEMPTS - PLEASE CONTACT SUPPORT
30	INVALID CARD DATA SUPPLIED.

8. Making Payments Using Tokens

In order to make payments using tokens you need to use the Real Time Payments option provided by the same API. Please note that you need to set up your credentials in the same manner as described before.

- JAVA

```
PaymentRealTimeRequest realTimeRequest = new PaymentRealTimeRequest();
CreditCard creditCard = new CreditCard();

// Enter TOKEN instead of actual card number
creditCard.setNumber("4564456687594564");
creditCard.setExpiry("1225"); *
realTimeRequest.setClientId(10000000);
realTimeRequest.setTransactionType(TransactionType.PURCHASE);
realTimeRequest.setCreditCard(creditCard);
// $10.10 dollars as the amount has to entered in cents
// "AUD" is the currency type . If this is not set then the default currency based
// on the country of the acquiring bank will apply
realTimeRequest.setTransactionAmount(new TransactionAmount(1010,"AUD"));
realTimeRequest.setClientRef("merchant_reference");
realTimeRequest.setComment("merchant_additional_data");
PaymentRealTimeResponse realTimeResponse = client.payment().realTime(realTimeRequest);
```

- NET

```
PaymentRealTimeRequest realTimeRequest = new PaymentRealTimeRequest();
CreditCard creditCard = new CreditCard();
// TOKEN
creditCard.number = "4564456687594564";
creditCard.expiry = "1225";
realTimeRequest.clientId = 10000000;
realTimeRequest.transactionType = Enums.TransactionType.PURCHASE.ToString();
realTimeRequest.creditCard= creditCard;
// $10.10 dollars
// "AUD" is the currency type . If this is not set then the default currency based
// on the country of the acquiring bank will apply
realTimeRequest.transactionAmount = new TransactionAmount(1010,"AUD");
realTimeRequest.clientRef = "merchant_reference";
realTimeRequest.comment = "merchant_additional_data";
PaymentRealTimeResponse realTimeResponse = client.payment.realTime(realTimeRequest);
```

- PHP

```
$realTimeRequest = new PaymentRealTimeRequest();
$creditCard = new CreditCard();
// TOKEN
$creditCard->setNumber("4564456687594564");
$creditCard->setExpiry("1225");
$realTimeRequest->setClientId(10000000);
$realTimeRequest->setTransactionType(TransactionType::$PURCHASE);
$realTimeRequest->setCreditCard($creditCard);
// $10.10 dollars
// "AUD" is the currency type . If this is not set then the default currency based
// on the country of the acquiring bank will apply
$transactionAmount = new TransactionAmount(1010,"AUD");
$realTimeRequest->setTransactionAmount($transactionAmount);
$realTimeRequest->setClientRef("merchant_reference");
$realTimeRequest->setComment("merchant_additional_data");
$realTimeResponse = $client->getPayment()->realTime($realTimeRequest);
```

Support and Troubleshooting

If you need assistance or encounter any issues during integration, please contact our support team at support@bancstac.com or call us on +94 115 143 143.