PESAFLOW

INTERGRATION SPECIFICATION

Document History:

Author	Release	Date
Evid Sibi	1.0.0.0 Review By - James Ayugi	9th December 2014
Evid Sibi	1.1.0.0 Review By - Evid Sibi	15th December 2014
Evid Sibi	1.2.0.0	10th January 2015
Evid Sibi	1.3.0.0	12th February 2016
Evid Sibi	1.4.0.0	9th December 2016

Charles Sewe	1.5.0.0	4th August 2023

Approval:

Project Manager	**Release**	**Date**
James Ayugi	1.3.0.0	15th February 2016
Charles Sewe	1.5.0.0	4th August 2023

1. Purpose

Purpose of this document is to describe the integration between third-party systems and pesaflow. The third party in this place is any system that wishes to provide online payments via the e-citizen payment platform, Pesaflow.

2. Preamble

This integration allows Pesaflow to collect payments on behalf of a third-party via different payment options, and for Pesaflow to notify the third party all payments made via the different payment channels.

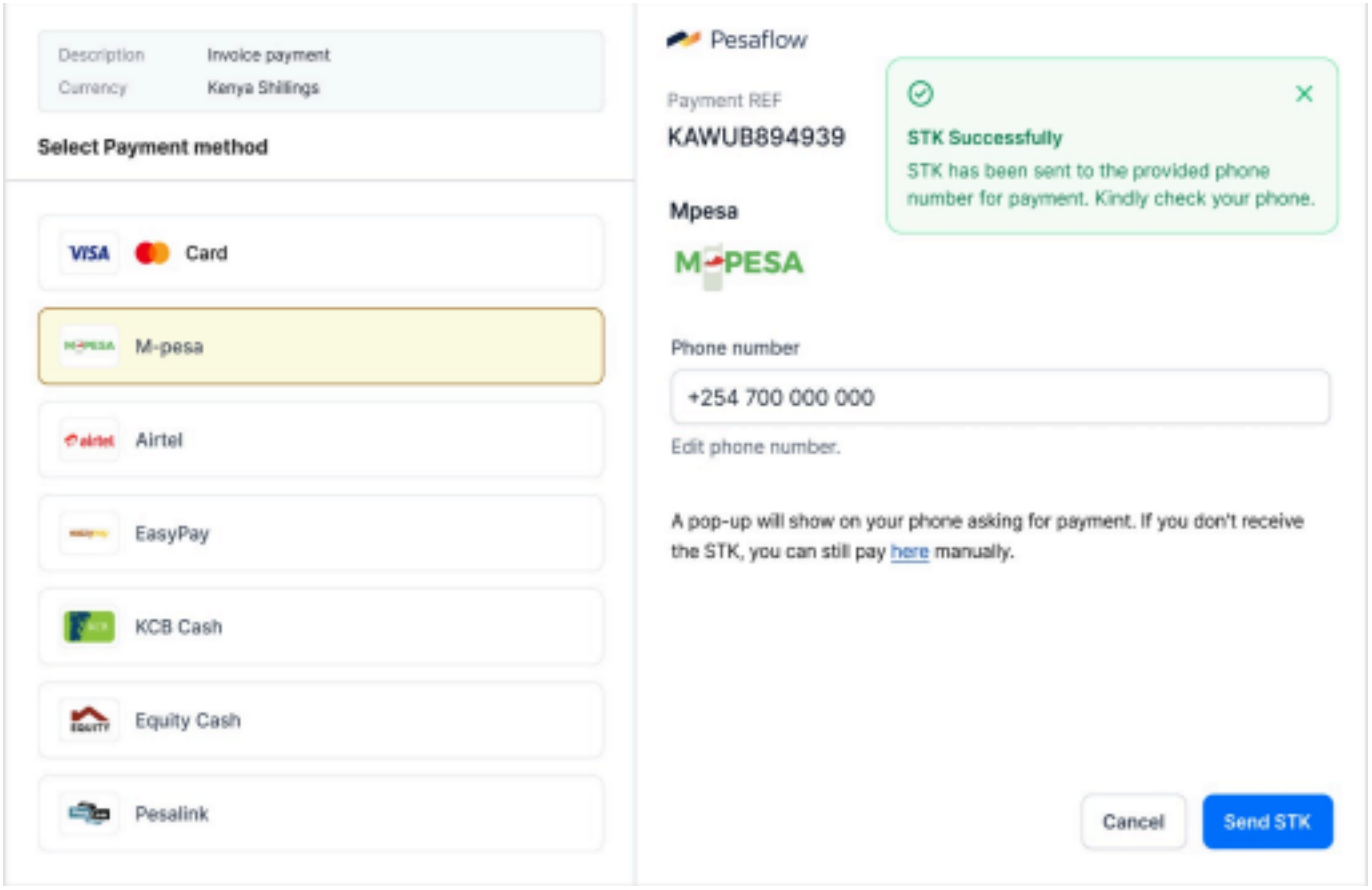
The main points of integration are:

1. **Online Checkout via IFRAME.**
2. **Instant Payment Notification (Push API)**

3. Query Payment Status API

3. Online Checkout

A third party system is expected to consume an I-frame via http-post and pass in the valid parameters to the Pesaflow URL in order to have a display as in the figure below.



PARAMETER	DESCRIPTION	TYPE
apiClientID	API client ID, assigned on activation	(M)String (Provided)
serviceID	Customer service ID, assigned on activation	(M) Int (Provided)
billDesc	Description of the Bill	(M)String

currency	Currency (KES or USD)	(M)Char(3)
billRefNumber	Reference or Invoice Number of The Bill	(M)String(Unique)
clientMSISDN	Mobile Phone Number of the transacting client	(M) BigInt (eg.25472222222)
clientName	Name of the transacting client	(M) String
clientIDNumber	Identification Number of the transacting client	(M)String
clientEmail	Email address of the Transacting client	(M)String
callBackURLONSuccess	Url the client is to be redirected to after successful payments	(M) URL
amountExpected	Full amount expected from client	(M)Float + Convenience FEE
notificationURL	Url instant payment notifications are sent after a successful transaction	(M) URL
secureHash	This is a SHA256 hmac_hash that signs the payload. Give a security. secure_hash = hmac_hash(sha256, 'apiClientID+amount+serviceID+clientIDNum ber+currency+billRefNumber+billDesc+client Name +secret) Base64.encode (secure_hash)	(M)String (Derived)
format	Format of the checkout (JSON or iframe)	String
sendSTK	Push STK on invoice creation (true or false)	String
PictureURL	Clients Image to be shown on the Iframe	URL

URL - https://test.pesaflow.com/PaymentAPI/iframev2.1.php

Protocol - HTTPS

4. IPN (Instant Payment Notification)

The third-party systems implement the following API in order to receive payment notifications from pesaflow.

URL: - Provided by third-party as notificationURL in section 3 abovePROTOCAL: - HTTPS POST

PARAMETER	DESCRIPTION	TYPE
payment_channel	Mode of payment	(M) String
client_invoice_ref	Third party transaction identifier will be the same as the billRefNumber pushed to the iframe	
payment_reference	Pesaflow unique payment ID	
currency	Currrency of the transaction	
amount_paid	Total amounts paid to date for the invoice (case for partial payments)	(M) Float + convenience fee
invoice_amount	Total amount raised for the invoice as pushed to the iframe	
status	Transaction status	
invoice_number	Unique pesaflow invoice reference	
payment_date	Date the payment was made	
token_hash	Pesaflow transaction authentication	
last_payment_amount	Current amount paid	(M) Float

RESPONSES

Format: JSON ENCODED STRING Pesaflow expects the following responses.

·If the wrong API key is used, you will give the following response.
HTTP STATUS CODE: 500

{"query_status":"unauthorized"}

·If the wrong invoice number is used, you will give the following response.
HTTP STATUS CODE: 500
{“query_status":"invalid invoice”}

·If no transaction id is specified by the payment provider
HTTP STATUS CODE: 500
{“query_status":"invalid transaction id"}

·If the amount paid is less than what is need for the invoice
HTTP STATUS CODE: 500
{“query_status":"invalid transaction amount"}

If both the API key and the invoice number is correct, you will give the following

response. HTTP STATUS CODE: 200

5. Query Payment Status API

In the event that push no-fica-on is not available, or Network Timeouts, clients can use this endpoint to query the status of bill payments.

This endpoint confirms a payment for an invoice. To generate a secure hash follow the below guidelines. Please note:

For confirma-on, different api_client_id, secret and key will be provided for different currency opera-ons.

data_string = "$api_client_id”. Ref_no"

hash = base64_encode(hash_hmac('sha256', $data_string, $key));

HTTP Request

GET hTps://test.pesaflow.com/api/invoice/payment/status

URL Parameters

Parameter	Mandatory	Description
api_client_id	Yes	Provided api client id
ref_no	Yes	Unique invoice number

		from Pesaflow
secure_hash	Yes	A sha256 encoded hash.

6. Create Invoice API

URL: https://test.pesaflow.com/api/invoice/create

Parameter	Mandatory	Type	Description
account_id	Yes	String	Provided service id
amount_expected	Yes	String	Total invoice amount
amount_net	Yes	String	Total invoice amount
amount_settled_offline	Yes	String	Any amount that is collected in cash for the invoice
callback_url	Yes	String	Success page/url to which clients will be redirected to upon payment
client_invoice_ref	Yes	String	Merchant System Generated Unique invoice reference
commission	Yes	String	Total commission amount charged on an invoice
currency	Yes	String	Invoice currency
email	Yes	String	Customer email address
format	Yes	String	The expected response format upon invoice creation. Can be json or html
id_number	Yes	String	Customer ID or Passport number
items	Yes	list	Invoice particulars / services billed on the invoices. See items table below

msisdn	Yes	String	Customer phone number
name	Yes	String	Customer Name
notification_url	Yes	String	Payment confirmation url

The Item fields is made up of the following fields

Parameter	**Mandatory**	**Type**	**Description**
account_id	Yes	String	Provided service id
desc	Yes	String	Service description
item_ref	Yes	String	Unique reference
price	Yes	String	Service cost per customer
quantity	Yes	String	No of Customers
require_settlement	Yes	String	Possible values true or false
currency	Yes	String	Item Currency

NB: This API requires a Bearer token include in the header for authentication to succeed.
Use the provided API Consumer Key and Consumer Secret to generate the token.

7. TOKEN Generation

URL: https://test.pesaflow.com/api/oauth/generate/token

Payload

key - Provided consumer key
secret - Provided consumer secret

APPENDIX I

SECURE HASH –
- SAMPLE PHP CODE

```php
<?php
//create your data_string
$data_string =
"apiClientID"."amount"."serviceID"."clientIDNumber"."currency"."billRefNumber"."billDesc".
"clientName"."secret"
//hash the values
$hash = hash_hmac('sha256', $data_string, $key);
//append the has values
$secureHash = base64_encode($hash);

?>
```

- SAMPLE C# CODE

```csharp
using System;
using System.Security.Cryptography;
using System.Text;

class Program
{
    static void Main()
    {
        // Values in the same order as Elixir
        string apiClientId = "223";
        string ref_no = "ERWEGTFD";
        string amount = "100";

        // secret key (same as Elixir :secret)
        string secret = "your_secret_here";
        string key = "";

        // Concatenate values into one string
        string dataString = apiClientId +
                            ref_no +
                            amount +
                            secret ;
```

```
        // Step 1: HMAC-SHA256
        byte[] keyBytes = Encoding.UTF8.GetBytes(key);
        byte[] messageBytes = Encoding.UTF8.GetBytes(dataString);

        using (var hmac = new HMACSHA256(keyBytes))
        {
            byte[] hashBytes = hmac.ComputeHash(messageBytes);

            // Step 2: Convert hash bytes to lowercase hex
            StringBuilder sb = new StringBuilder();
            foreach (byte b in hashBytes)
                sb.Append(b.ToString("x2"));

            string hexHash = sb.ToString();

            // Step 3: Base64 encode hex string (to match Elixir)
            string finalHash = Convert.ToBase64String(Encoding.UTF8.GetBytes(hexHash));

            Console.WriteLine("Secure Hash: " + finalHash);
        }
    }
}
```

- SAMPLE PYTHON CODE

```
import hmac
import hashlib
import base64

api_client_id = "1"
amount_expected = "100"
account_id = "1"
id_number = "1"
currency = "KES"
client_invoice_ref = "1"
desc = "Description"
name = "John Doe"
secret = "1f8THxVsZeilWfUiayDg2DNg4bYCP+HM"
```

```
key = "QO9lwAzI7cqbp74s".encode("utf-8")

data_string = (
    api_client_id
    + amount_expected
    + account_id
    + id_number
    + currency
    + client_invoice_ref
    + desc
    + name
    + secret
)

hmac_hash = hmac.new(key, data_string.encode("utf-8"), hashlib.sha256).digest()

hex_string = hmac_hash.hex()

secure_hash = base64.b64encode(hex_string.encode("utf-8")).decode("utf-8")

print(secure_hash)
```

- SAMPLE JAVA CODE

```
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.*;
import java.lang.*;
import java.io.*;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.net.ssl.HttpsURLConnection;
public class Main {
private final String USER_AGENT = "Mozilla/5.0";
  public static void main(String[] args) throws Exception {
      Main m = new Main();
      String hash = m.genSig();

      System.out.println("\nHash - "+ hash);
  }
  // HTTP POST request
```

```
  private String genSig() {

      /**
       * Validate Payment
       * This endpoint validates a invoices or topups exist before payment is accepted.
       * To generate your secure hash, use format below,
       *
       */


      String provided_hmac_secret = "your-key";
      String api_client_id= "your-api-client-id";
      String data_string = "[YOUR-DATA-STRING]";

      return  generate_signature(data_string, provided_hmac_secret);
  }

  static String generate_signature(String data_string, String secret){
     try{

           SecretKeySpec keySpec = new SecretKeySpec(secret.getBytes(),"HmacSHA256");

           Mac mac = Mac.getInstance("HmacSHA256");
           mac.init(keySpec);
           byte[] result = mac.doFinal(data_string.getBytes());

          String encode_result = Base16Encoder.encode(result);


          return
Base64.getEncoder().encodeToString(encode_result.toLowerCase().getBytes());

       } catch (Exception e) {

            return "Cannot Process Signature";
      }
  }
}
class Base16Encoder {
    private final static char[] HEX = new char[]{
        '0', '1', '2', '3', '4', '5', '6', '7',
        '8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };
    /**
     * Convert bytes to a base16 string.
     */
    public static String encode(byte[] byteArray) {
        StringBuffer hexBuffer = new StringBuffer(byteArray.length * 2);
        for (int i = 0; i < byteArray.length; i++)
```

```
            for (int j = 1; j >= 0; j--)
                hexBuffer.append(HEX[(byteArray[i] >> (j * 4)) & 0xF]);
        return hexBuffer.toString();
    }

    /**
     * Convert a base16 string into a byte array.
     */
    public static byte[] decode(String s) {
        int len = s.length();
        byte[] r = new byte[len / 2];
        for (int i = 0; i < r.length; i++) {
            int digit1 = s.charAt(i * 2), digit2 = s.charAt(i * 2 + 1);
            if (digit1 >= '0' && digit1 <= '9')
                digit1 -= '0';
            else if (digit1 >= 'A' && digit1 <= 'F')
                digit1 -= 'A' - 10;
            if (digit2 >= '0' && digit2 <= '9')
                digit2 -= '0';
            else if (digit2 >= 'A' && digit2 <= 'F')
                digit2 -= 'A' - 10;

            r[i] = (byte) ((digit1 << 4) + digit2);
        }
        return r;
    }
}
```

- SAMPLE ELIXIR CODE

```
def generate_secure_hash(secret, hash_list) when is_list(hash_list) do
   :crypto.hmac(:sha256, secret, hash_list |> Enum.map(&to_string(&1)))
    |> Base.encode16()
    |> String.downcase()
    |> Base.encode64()
end
# call the function
generate_secure_hash("my-key",["1","100","124","12345678","KES","REF-01-2024-XVBC","Some
invoice","John Doe","my-secret"])
```