E-Billing Integration Guide

WEB & MOBILE APPLICATIONS AND WOOCOMMERCE

Digitech Africa SARL
ANCIENNE SOBRAGA (EN FACE DE L'INSTITUT IRTEC) | CONTACT@DIGITECH-AFRICA.COM

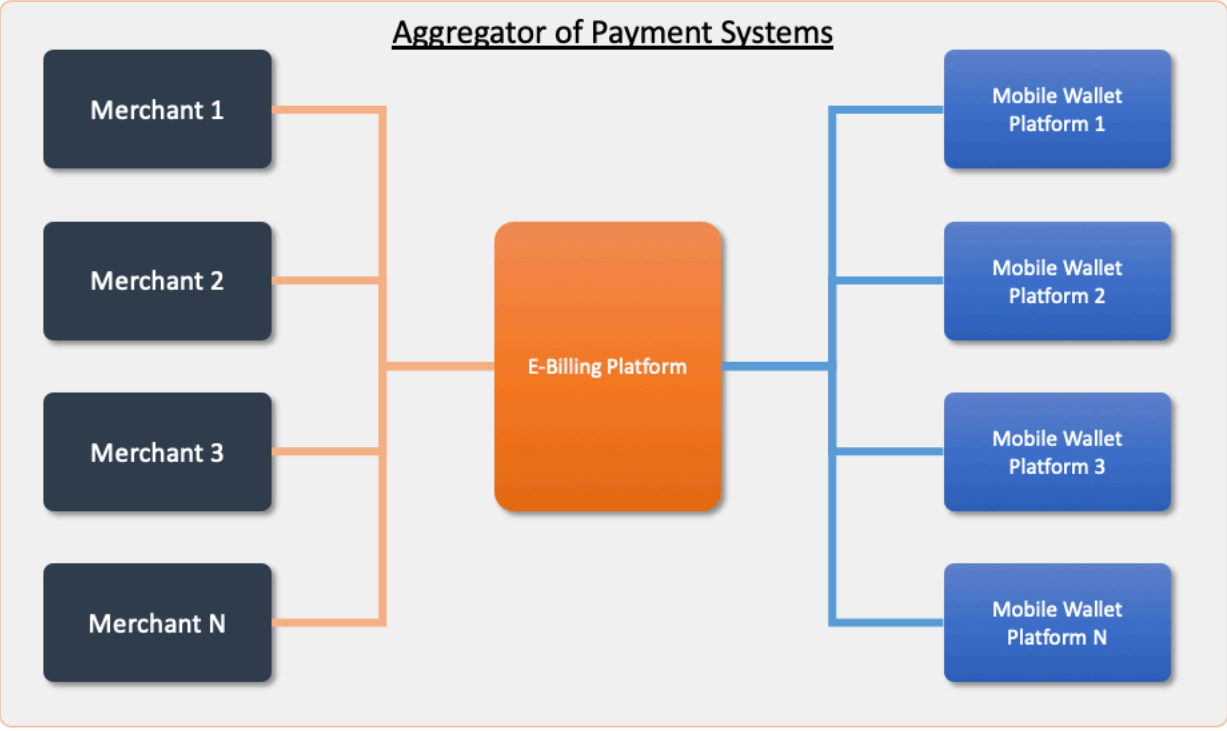
Integration Guide v5

Table of Contents

1 Introduction to payment with E-Billing

1.1 E-Billing, what is it?

E-Billing is a payment gateway that aggregates payment systems and provides to merchants a unique API to integrate payments in their applications.



E-Billing is **not** a mobile wallet.

E-Billing does **not** manage any transaction flow.

E-Billing does **not** collect payment on behalf of merchant. Payment is always **validated** by target payment system.

E-Billing has put in place an automated system that ensures traceability of transaction payment between customer, target payment system and merchant.

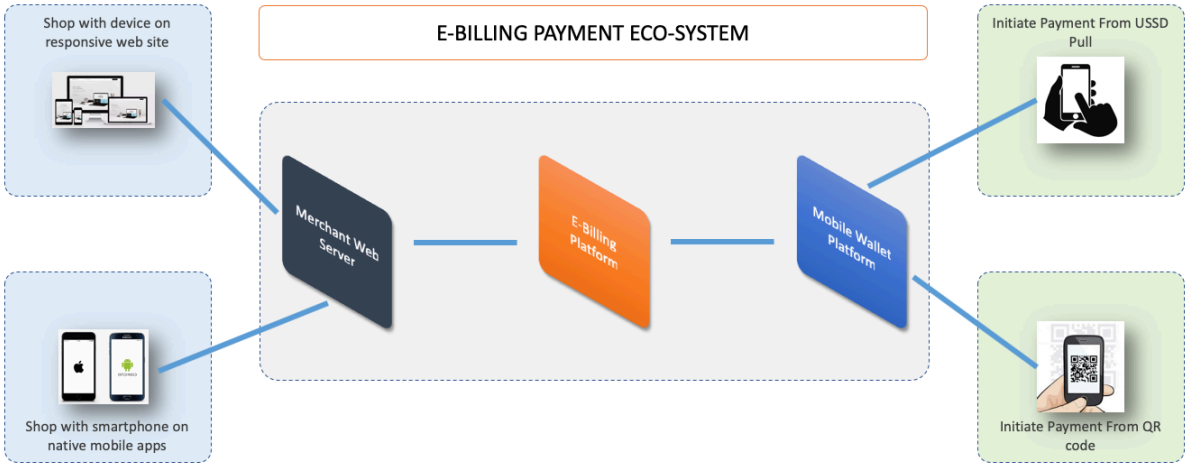
E-Billing has successfully aggregated two (2) payment systems in Gabon: **Airtel Money** and **MobiCash** of Gabon Telecom.

1.2 E-Billing, how it works?

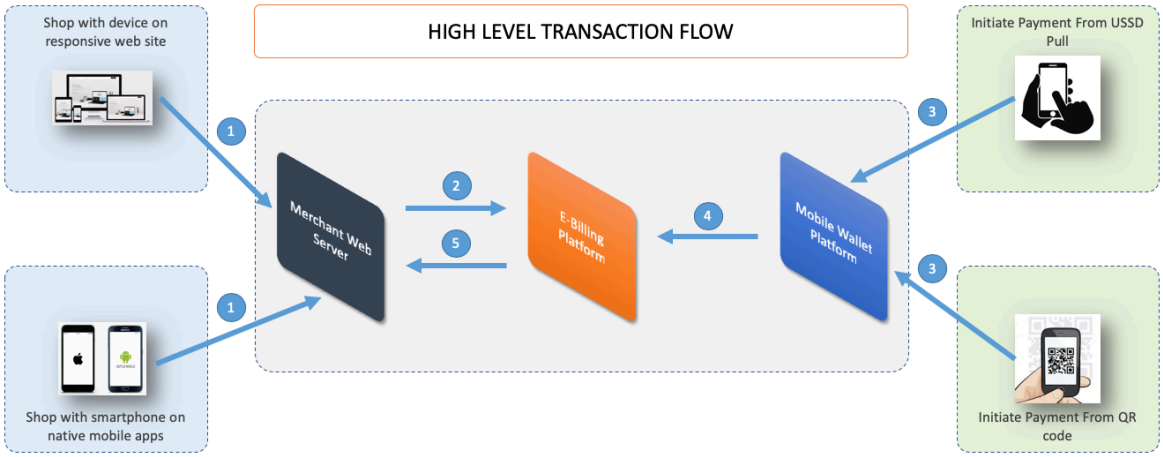
E-Billing connects to the backend systems of **Mobile Wallet Platform** and that of **Merchant** (Web Server typically).

Customer connects to merchant backend system using their browser or mobile application to shop and checkout.

Customer connects to mobile wallet provider to pay transaction using their mobile phone.



To initiate a transaction, customer clicks on **Checkout** (1) button from browser or from mobile application. This action triggers merchant web server to call E-Billing to generate a **unique ID** for the transaction (2). Customer uses transaction ID to initiate payment with mobile wallet (3). Once payment is processed successfully by mobile wallet, it notifies E-Billing (4). E-Billing in turn notifies merchant web server of successful payment of the transaction (5) as seen below:



1) Customer clicks on checkout to initiate transaction. Anything that happens before is application specific (shopping, delivery method, pricing etc.)
2) Merchant Web Server calls E-Billing Platform to generate a unique **Transaction ID**
3) Customer initiates payment with **Mobile Wallet Platform** through USSD Pull or QR code reader using his phone. If shopping is done with smartphone (responsive website or native mobile application), customer will just need to enter his mobile wallet PIN code to validate payment
4) Mobile **Wallet Platform** notifies E-Billing Platform when payment is successful
5) E-Billing notifies **Merchant Web Server** when payment is successful

2 High level integration tasks

In this section, we provide high level description of the tasks you have to perform for the integration of payment with E-Billing. You will go through similar tasks if you integrate other solution like **PayPal**.

We are considering two (2) use-cases: integration of payment in web-based application and integration in native mobile application (iOS, Android).

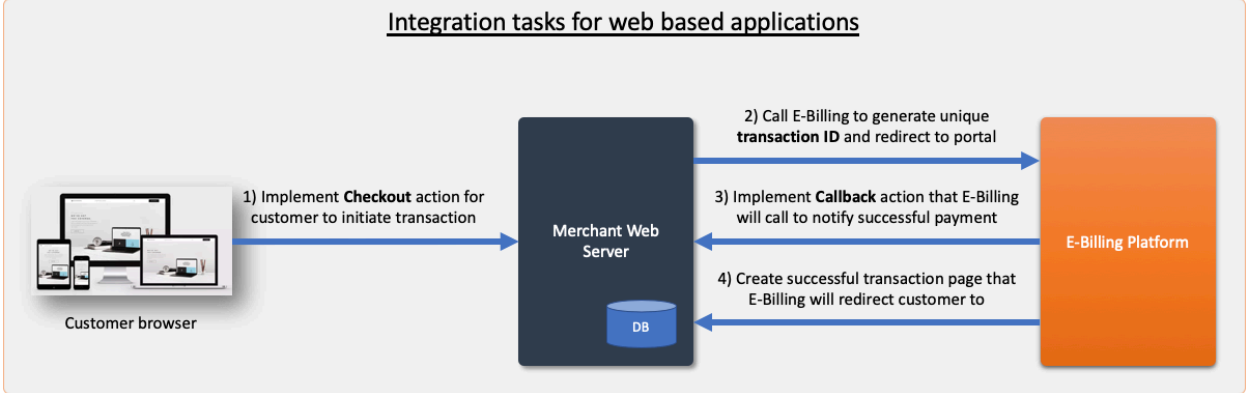
The core tasks remain the same for both use-cases. For integration with **native** mobile application, the main difference is that at some point, customer must leave the application to go payment page and after successful payment, he must leave browser return to the application.

The core tasks are performed in the merchant backend system, typically a web server.

If you are using WooCommerce for your online store, there is plugin available for E-Billing that automates all integration steps. You will find the link at the bottom of this document.

2.1 Integration tasks for web-based applications

If your application is 100% web, responsive, meaning it can be accessed from multiple devices size (computer, tablet, smartphone), you will need to perform the tasks below to integrate payment with E-Billing.



2.1.1 Step1: Implement checkout action

Checkout action is the starting point of the integration with E-Billing. Anything that happens before is application specific.

At this point, application should have collected all information to process payment including but not limited to:

- **Customer information**: full name, email, phones, billing address, shipping address;
- **Product information**: name, description;
- **Pricing**: product price, shipping price, total price;

By clicking on checkout button, customer is initiating transaction and he is agreeing to merchant Terms & Conditions.

This task should create an entry in database in the **Transaction** table or whatever it is named.

2.1.2 Task 2: Implement call to E-Billing

This task implements a call to E-Billing API to create a unique ID for the transaction. If this action is successful, this task should also redirect customer to E-Billing payment portal.

The below minimum information must be provided to E-Billing API to create transaction ID:

- **Amount**: total amount to be paid by customer;
- **Reference**: a unique reference id for the transaction in merchant database;
- **Short description**: a short description of the transaction;
- **Customer information**: name, phone, email, address;
- **Success page URL**: the URL of the page to redirect customer to after a successful payment;

2.1.3 Task 3: Implement Callback action

This task implements a **callback** action that E-Billing will call to notify merchant of successful payment. The URL must be registered under merchant profile in E-Billing portal.

The bellow minimum information will be provided when E-Billing calls the **callback** action:

- **Reference**: the unique reference id of the transaction in merchant database that was provided when transaction id was created;
- **Mobile wallet name**: the name of mobile wallet that performed payment;
- **Transaction id**: the unique transaction from mobile wallet system;

This is probably the most important development activity in the integration process.

2.1.4 Task 4: Create successful transaction page

This task creates a page where customer will be redirected to by E-Billing portal after successful payment is confirmed by the mobile wallet system. The URL of this page is sent to E-Billing on task 2.

2.2 Integration tasks for native mobile applications

If you have a native mobile application, iOS and Android, E-Billing offers three (3) ways to integrate online payment.

2.2.1 Use E-Billing portal turnkey solution

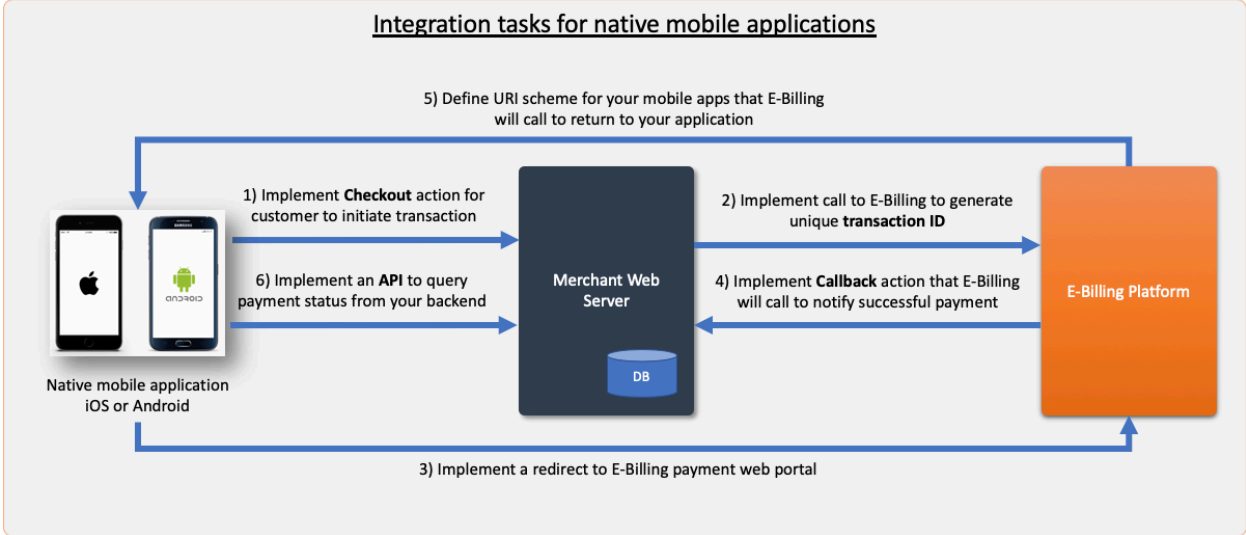
You can rely on E-Billing payment portal to allow customer selecting the operator he wants to use to pay. With this approach, your application does not need to manage which operator the end user is going to use to pay, he/she will make determination from E-Billing portal.

This approach simply requires that user temporary leaves the application and goes to the portal and after payment is completed he will come back.

App URI scheme: the URI scheme of the application. After successful payment, this URI will be used to return to the application from E-Billing portal, please follow these guidelines:

https://blog.branch.io/how-to-open-an-android-app-from-the-browser/

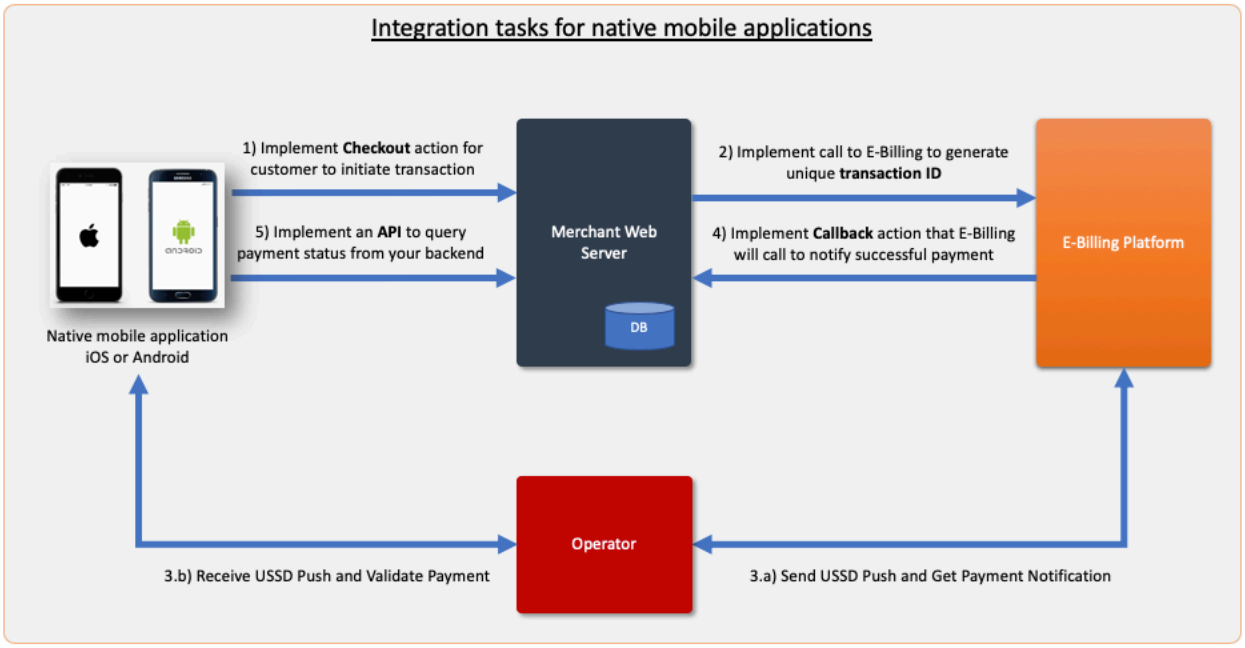
https://stackoverflow.com/questions/25883113/open-ios-app-from-browser



2.2.2 E-Billing USSD Push

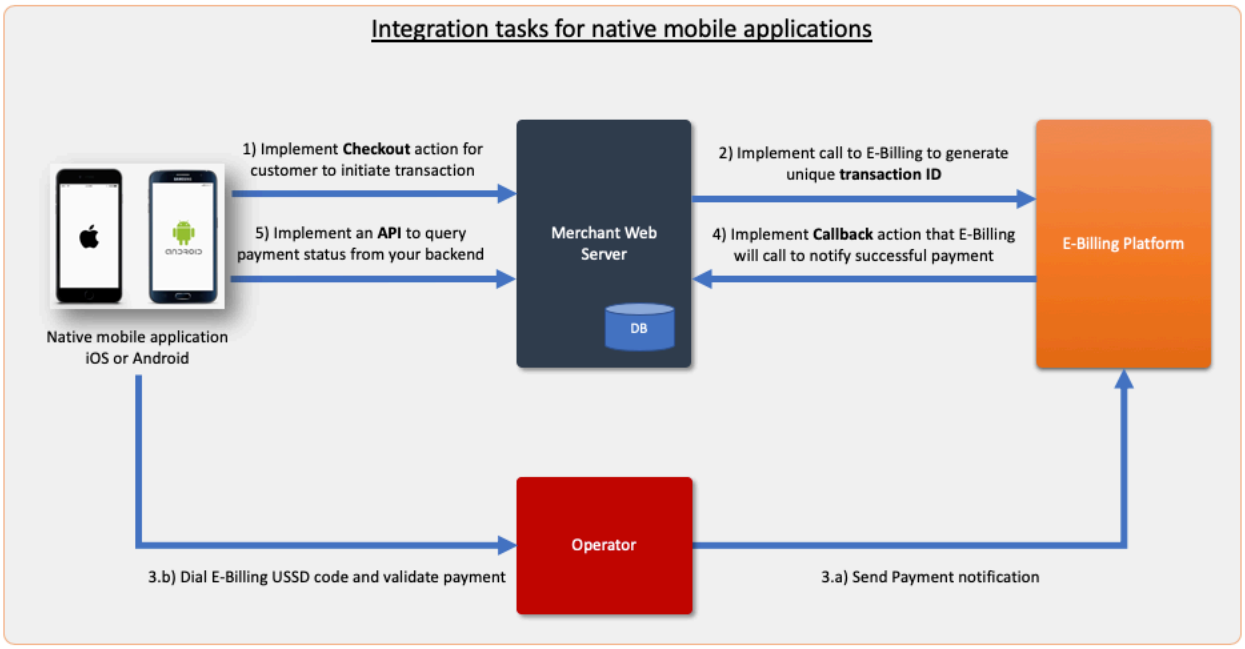
The 2nd approach is to use USSD Push API provided by E-Billing. This approach requires that you let customer select the operator he/she wants to use to pay.

Basically, when user selects the operator and provides the phone number to use to receive USSD Push, E-Billing will send USSD Push to the end user via the operator chosen.



2.2.3 Dial E-Billing USSD Code

The 3rd approach is to dial E-Billing USSD code directly from your application (Android, iOS), for instance: *150*8*5551234567*1# - when dialing this USSD code, user will be prompted to enter his/her PIN to validate payment.



2.2.4 Task 1: Implement checkout action

Checkout action is the starting point of the integration with E-Billing. Anything that happens before is application specific.

At this point, application should have collected all information to process payment including but not limited to:

- **Customer information**: full name, email, phones, billing address, shipping address;
- **Product information**: name, description;
- **Pricing**: product price, shipping price, total price;

By clicking on checkout button, customer is initiating transaction and he is agreeing to merchant Terms & Conditions.

This task should create an entry in database in the **Transaction** table or whatever it is named.

2.2.5 Task 4: Implement callback action

This task implements a **callback** action that E-Billing will call to notify merchant of successful payment. The URL must be registered under merchant profile in E-Billing portal.

The bellow minimum information will be provided when E-Billing calls the **callback** action:

- **Reference**: the unique reference id of the transaction in merchant database that was provided when transaction id was created;
- **Mobile wallet name**: the name of mobile wallet that performed payment;
- **Transaction id**: the unique transaction from mobile wallet system;

This is probably the most important development activity in the integration process.

2.2.6 Task 6: Implement an API to query payment status

When customer returns to mobile application, merchant backend has already updated. Mobile application should be able to query backend to get an update of the payment status.

IMPORTANT:

When payment is validated by the operator, E-Billing will be notified automatically. Upon reception of this notification, E-Billing will notify your backend application of the successful payment.

The backend application shall provide an API for the mobile application to query status of the payment. The mobile application should not query status of payment directly to E-Billing application.

3 Integration environment

The integration environment includes two (2) essential tools: E-Billing Lab Platform and Integration Simulator.

3.1 E-Billing Lab Platform

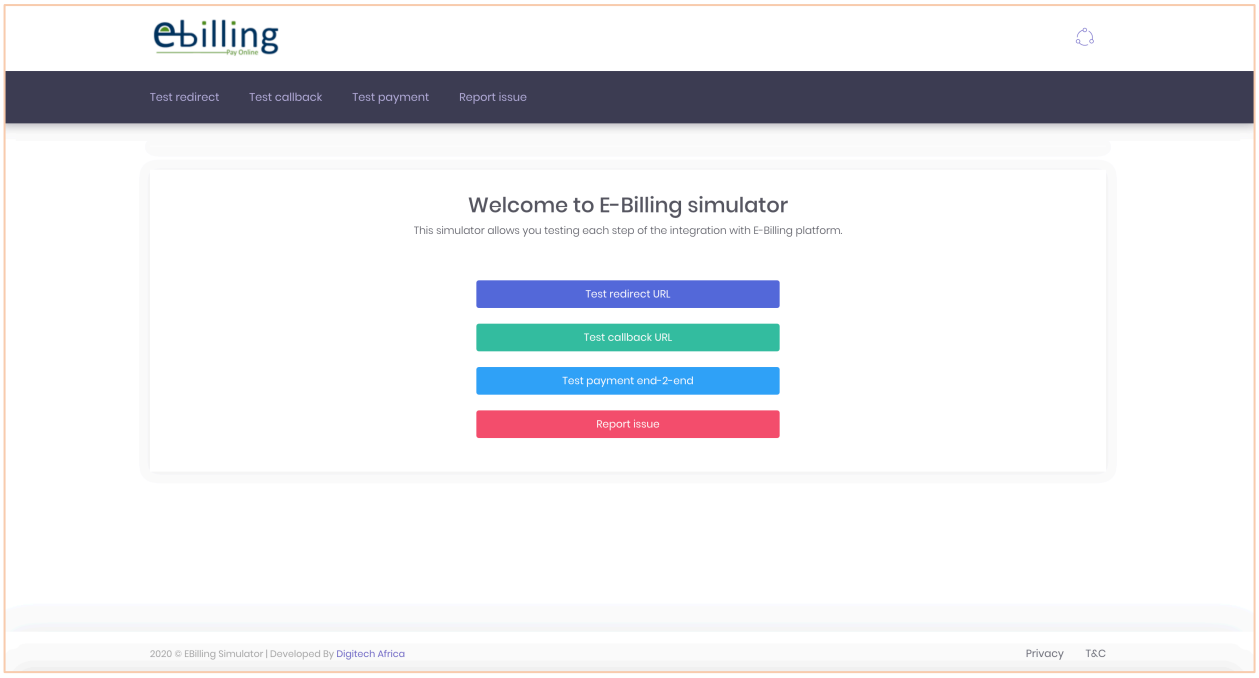
Lab platform is functionally identical to production. This is to make sure that when you can complete integration and testing in Lab, everything should work just fine in production.

Here is the URL of Lab: https://lab.billing-easy.net

3.2 Integration Simulator

A simulator is available to provide you with tool to test most of integration tasks.

Here is the URL of the Integration Simulator: https://simu.billing-easy.net



With the simulator, you will be able to test separately the following features:

(1) **Redirection to the payment success page**: this is very basic test, you just need to specify the URL and ensure that you are being redirected to the page;
(2) **Callback action**: this test is extremely important. You provide URL of your callback and you select parameters that you want to receive;
(3) **End-to-end test**: this is a mandatory test that must be successful in order to validate your integration with E-Billing;

4 Detailed integration activities

In this section, we are going deeper in the integration tasks that you have to perform to add payment with E-Billing to your application, web-based or mobile native.

4.1 Prerequisites

Let first list what you need to have in hand for the integration of payment with E-Billing.

4.1.1 Backend system

Whether or not you are integrating payment on your web-based application or **native** mobile application, you need to have a backend system, typically a web application server.

In your backend system, you need to have a database with a table to track payment status. Let name that table: **TRANSACTION** with at least the following columns:

Column	Type	Null	Key	Default	Comment
id	bigint(20)	NO	PRI	NULL	The primary key
customer_id	bigint(20)	NO	MUL	NULL	This column references customer initiating transaction
product_id	bigint(20)	NO	MUL	NULL	This column references product (good or service) being purchased by customer
amount	float	NO		0.0	This column is the total amount of the transaction
description	varchar(20)	NO		NULL	This column is a short description of the transaction
reference_id	bigint(20)	NO		NULL	This column is the unique reference number of the transaction in your system
status	int(1)	NO		0	This column contains a flag that indicates the payment status of this transaction
timeout	int(10)	NO		NULL	You may wish to have timeout for this transaction to be cancelled if not paid
created_at	timestamp	NO		NULL	This column contains the date and time this transaction was initiated
paid_at	timestamp	NO		NULL	This column contains the date and time the payment notification was received
expired_at	timestamp	NO		NULL	This column defines date et time this transaction is cancelled is not paid
ebilling_id	varchar(20)	YES		NULL	This column contains the unique ID of this transaction in E-Billing system
operator	varchar(50)	YES		NULL	This column contains the name of wallet provider who validated payment
transaction_id	varchar(50)	YES		NULL	This column contains the unique ID of this transaction in the operator system

Of course, you can adjust column type/name to better suit your needs/environment.

Payment with E-Billing is done in **asynchronous** way:

(1) You submit payment;
(2) You wait to be notified by external system (E-Billing) when payment is validated by operator or bank;

This TRANSACTION table is a way for you to:

(1) Keep track of all transactions initiated;
(2) Identify which transaction was paid when you get notification from E-Billing;
(3) Identify transactions not paid or expired;
(4) Provide payment information to customer;
(5) Etc.

The column **status** could be used to track transaction state, you can define these values for instance:

- **Created**: status just after create the new entry and before getting unique ID of the transaction from E-Billing;
- **Pending**: this could be the status after receiving the unique ID from E-Billing;
- **Paid**: this could be the status after payment notification was received from E-Billing;
- **Expired**: this could be the status if transaction was not paid after timeout;

4.1.2 E-Billing merchant account

To begin integration, you need to have your E-Billing merchant account information in LAB environment.

Here is URL for LAB: https://lab.billing-easy.net

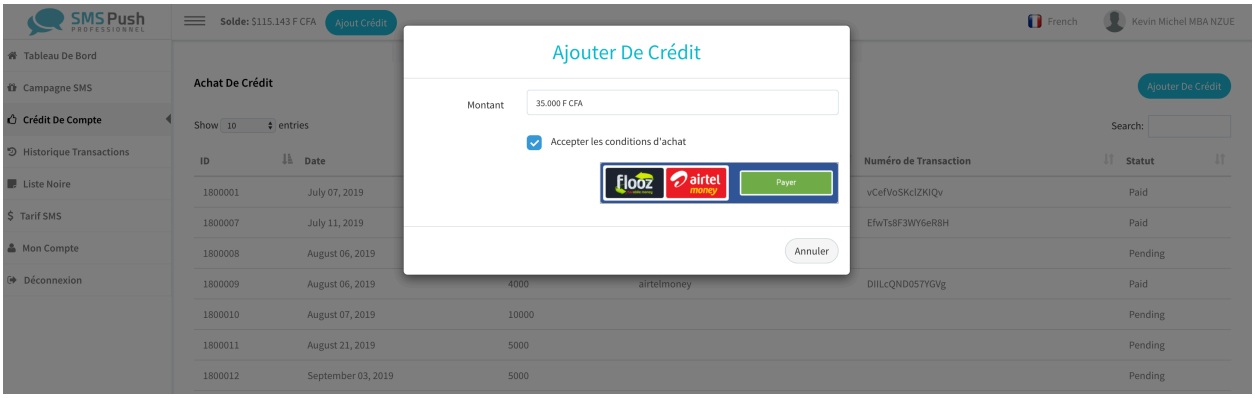
Here is URL to create your merchant account if you don't have one already: https://lab.billing-easy.net/merchant/registrations/new

You need the following information from your E-Billing account:

(1) Username
(2) Password
(3) Shared key

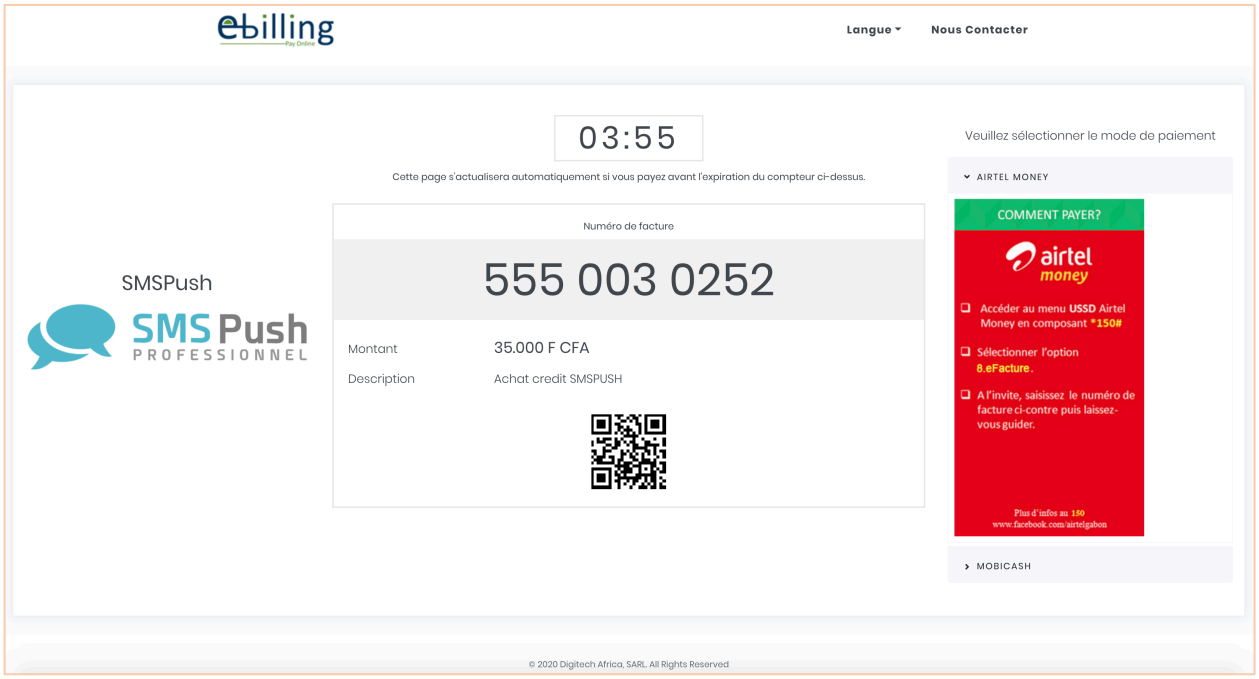
4.2 Initiate payment with E-Billing

The first task of integration is to initiate payment with **checkout** button. Here is an example from www.smspushpro.com:



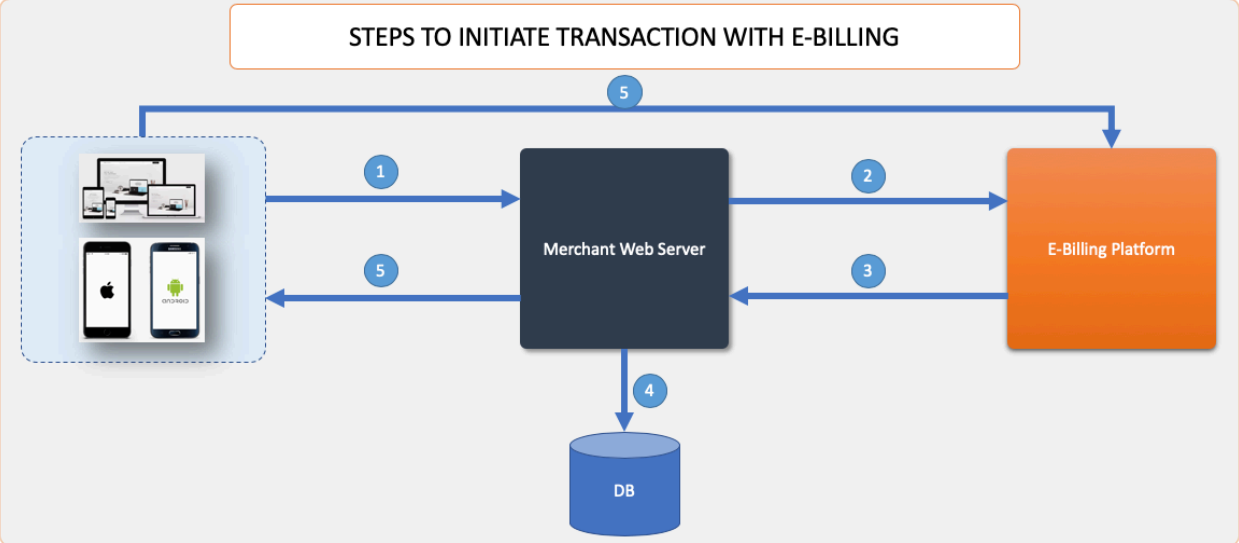
At this point, all information to process payment have already been collected.

When customer clicks on "**Payer**" the equivalent of "**Checkout**", he will be directed to this below page:



This is the standard payment page of E-Billing platform. It provides customer with procedures to pay with his preferred mobile wallet or bank application. Payment steps are completely transparent to merchant.

The initiation of transaction with E-Billing is done in five (5) steps as detected below:



1) Initiate transaction by adding an entry in table TRANSACTION
2) Call E-Billing API create transaction (**this call is synchronous – REST API**)
3) Get unique ID of transaction in E-Billing
4) Update TRANSACTION table with **ebilling_id**, the unique ID of the transaction in E-Billing
5) Redirect customer to E-Billing payment page

4.2.1 Step 1: customer clicks on checkout

After collecting transaction information, you should basically create an entry in table TRANSACTION which should look like this:

id	customer_id	product_id	amount	description	reference_id	status	timeout	created_at	paid_at	expired_at	ebilling_id	operator	transaction_id
100001	178545	6532145	35000.0	Alfiery Edrise MOUTINDI	990015	created	60	23-04-2020 15:20		23-04-2020 16:20			

4.2.2 Step 2: call E-Billing API

In this step, you need to call E-Billing REST API to create a transaction.

API Path: **/api/v1/merchant/e_bills**

- LAB: https://lab.billing-easy.net/api/v1/merchant/e_bills
- PROD: https://www.billing-easy.com/api/v1/merchant/e_bills

API Verb: **POST**

Here is list of parameters for the API:

Param	Type	Required	Default value	Description
amount	double	YES	NONE	This is the amount to be paid
short_description	string	YES	NONE	This is a short description of the transaction
payer_email	string	YES	NONE	This is the email of the customer
payer_msisdn	string	YES	NONE	This is the phone number of customer
payer_name	string	YES	NONE	This is the full name of the customer
external_reference	string	YES	NONE	This is the unique ID of the transaction in merchant DB
expiry_period	integer	NO	0	This param indicates if a timeout (in minutes) should be associated to transaction

You will find sample code in PHP and Java to achieve same at the end of the document.

4.2.3 Step 3: Update TRANSACTION table

That table should look like this after this step:

id	customer_id	product_id	amount	description	reference_id	status	timeout	created_at	paid_at	expired_at	ebilling_id	operator	transaction_id
100001	178545	6532145	35000.0	Alfiery Edrise MOUTINDI	990015	pending	60	23-04-2020 15:20		23-04-2020 16:20	5551234567		

- **status** column has changed from **created** to **pending**;
- **ebilling_id** column was updated with value returned by E-Billing;

4.2.4 Step 4: redirect customer to E-Billing portal

If customer is shopping from browser or native mobile application, you need to redirect him to E-Billing portal. You need to pass 2 parameters in POST method:

- **invoice_number**: this is the unique transaction ID generated by E-Billing;
- **merchant_redirect_url**: this is the URL that E-Billing will redirect customer to if payment is made;

Here is sample PHP code:

```
// LAB payment port
$POST_URL = 'https://test.billing-easy.net';
// Merchant success page
$invoice_number = 'URL TBD';
// Get response in JSON format
$response = json_decode($json_response, true);
// Get transaction from E-Billing response
$invoice_number = $response['e_bill']['bill_id'];
// Redirect to E-Billing portal
echo "<form action='" . $POST_URL . "' method='post' name='frm'>";
echo "<input type='hidden' name='invoice_number' value='".$invoice_number."'>";
echo "<input type='hidden' name='merchant_redirect_url' value='".$merchant_redirect_url."'>";
echo "</form>";
echo "<script language='JavaScript'>";
echo "document.frm.submit();";
echo "</script>";
```

4.3 Implement callback

This activity is all happening on your side. You need to create a **callback** action and register his URL in the merchant profile in E-Billing portal.

What is perform during callback is totally transparent to customer. Callback action is used by **E-Billing** to notify your backend that payment was successful. The callback has no interaction with customer at all. There is no visual page.

At the bar minimum, callback should update TRANSACTION table as follow:

id	customer_id	product_id	amount	description	reference_id	status	timeout	created_at	paid_at	expired_at	ebilling_id	operator	transaction_id
100001	178545	6532145	35000.0	Alfiery Edrise MOUTINDI	990015	paid	60	23-04-2020 15:20	23-04-2020 15:25	23-04-2020 16:20	5551234567	Airtel Money	MP200423.2209.C26009

- **Status** column was update from **pending** to **paid**;
- **Paid_at** was update with date and time callback was received;
- **Operator** column was update with name of the operator;
- **Transaction_id** column was updated with value from the operator;

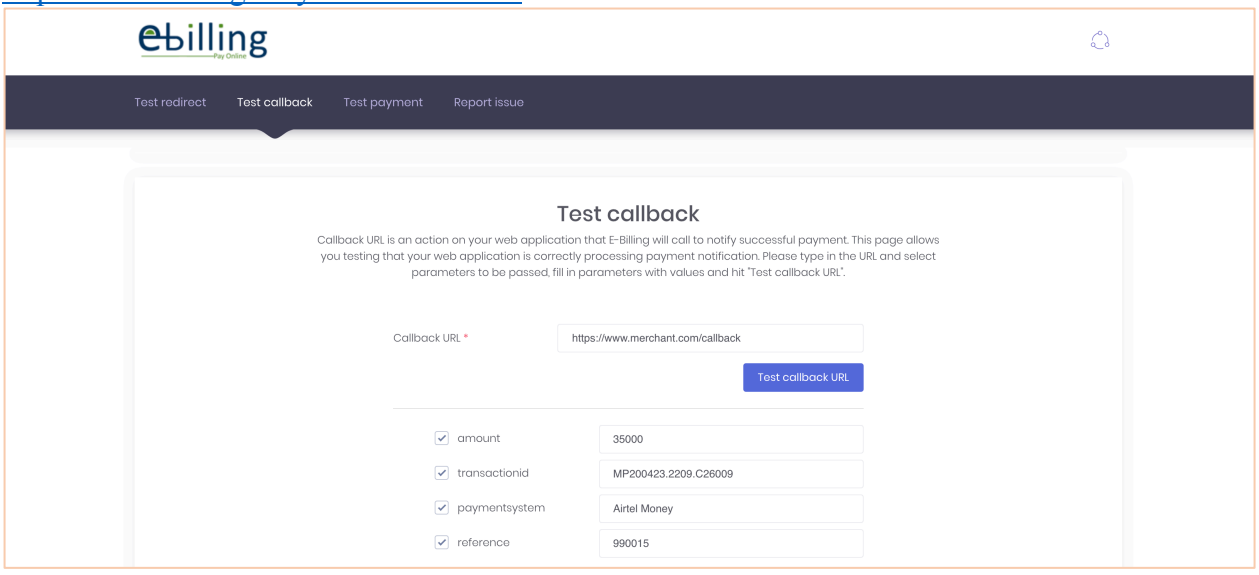
4.3.1 Callback http response code

When invoking the callback action, E-Billing will check the HTTP code returned to determine whether or not payment notification was processed successfully by merchant.

It is important to ensure you return **2xx** if processing was successful and **4xx** if there was an error.

4.3.2 Test callback

You can use the integration simulator to test your callback action. Here is the URL: https://simu.billing-easy.net/callback-url

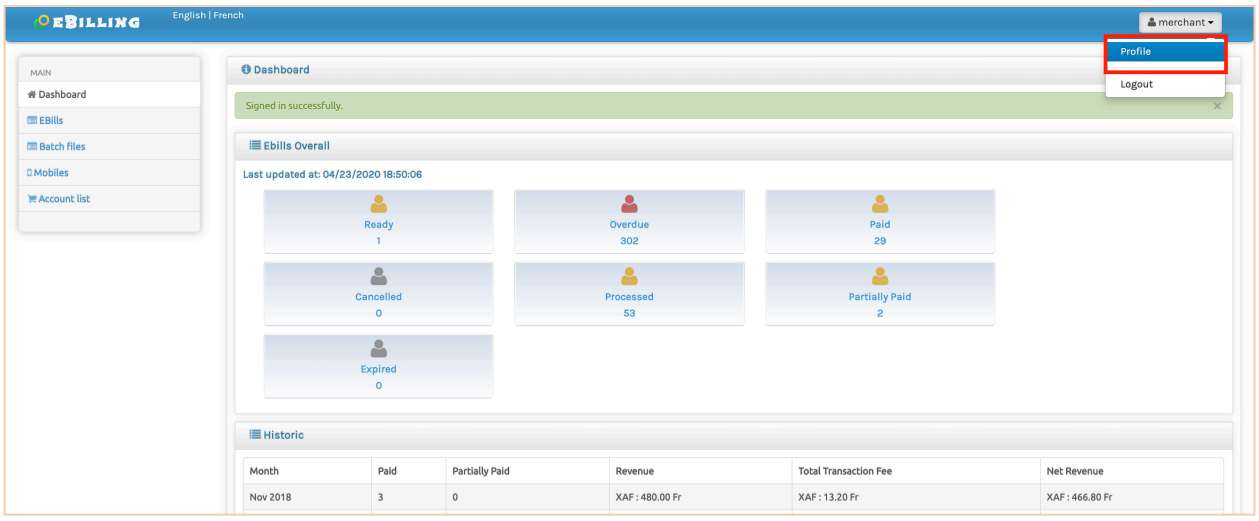


The simulator will call the callback action with parameters selected. Param names are exactly as displayed.

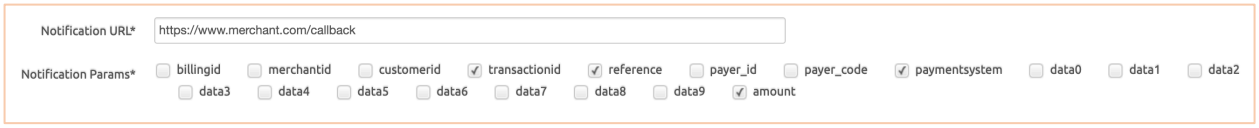
4.3.3 Register callback URL

If callback test is successful, you can register URL in E-Billing portal under merchant profile.

After login to merchant, go to "**Profile**" on the top right corner:



On the page that follow, please click on the "**Edit**" button then go to "**Notification URL**" section:

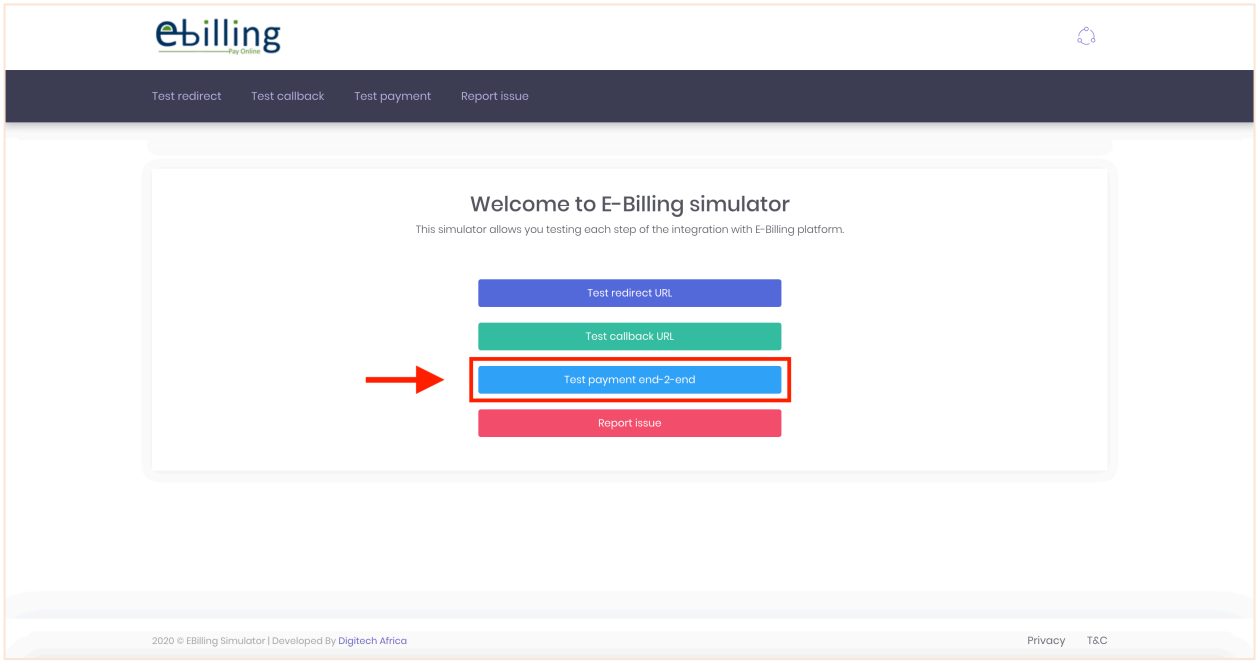


What you need to do is:

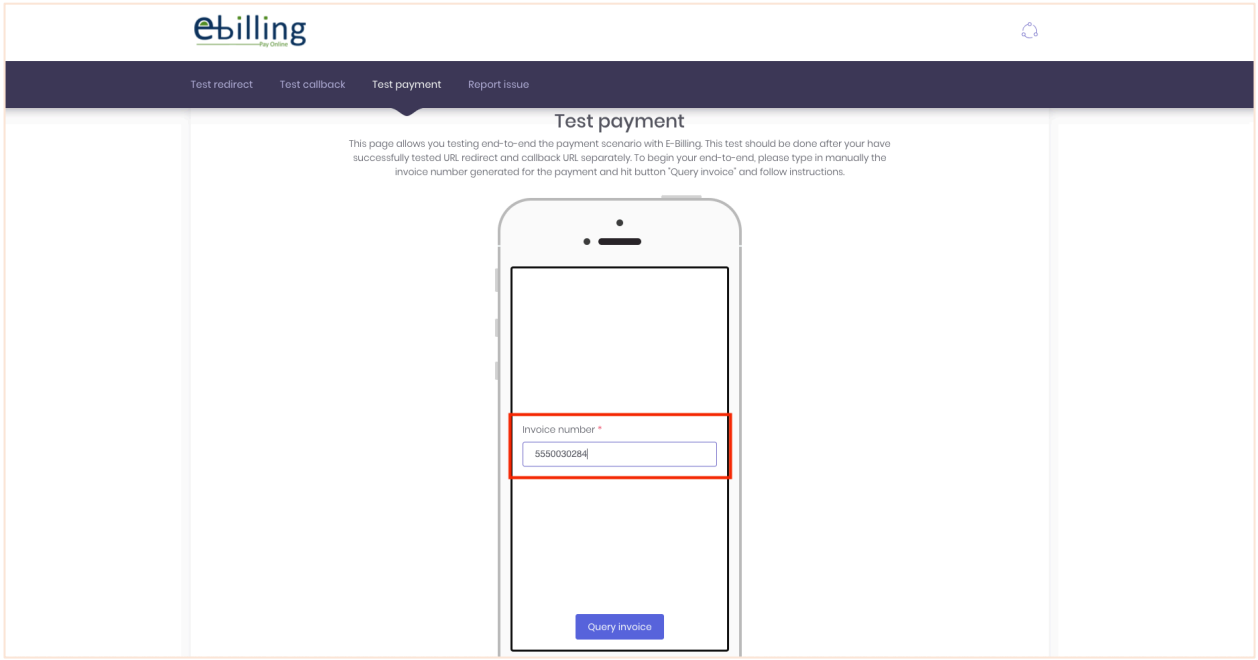
- Enter your callback URL;
- Select parameters you want to receive, at the minimum: **transactionid, reference, paymentsystem, amount**;

4.3.4 Callback end-to-end testing

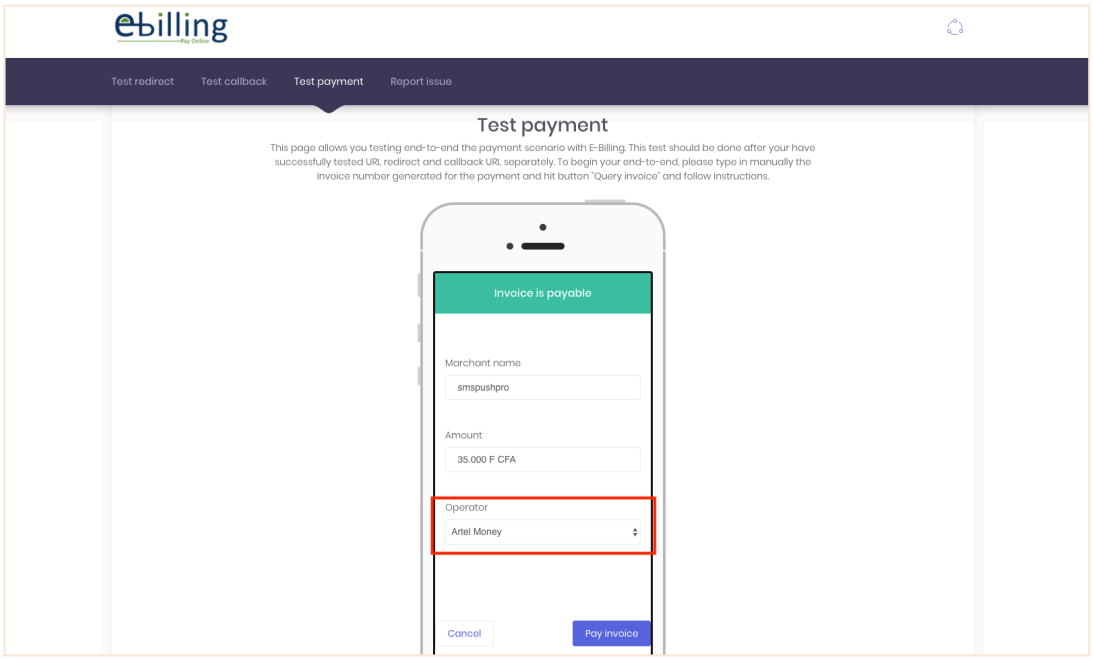
After **callback** was registered in merchant profile, you can proceed to end-to-end testing. For that you will use "**Test payment end-2-end**" button:



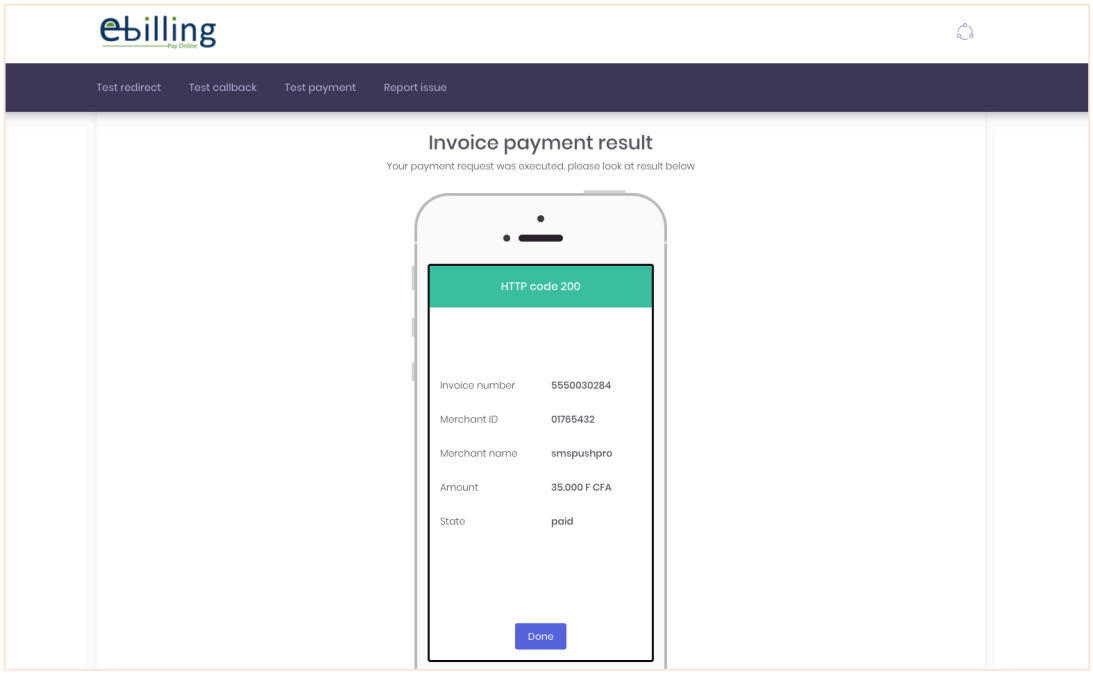
On the page that follows, please enter invoice number (ID of transaction) as shown below and click "**Query invoice**"



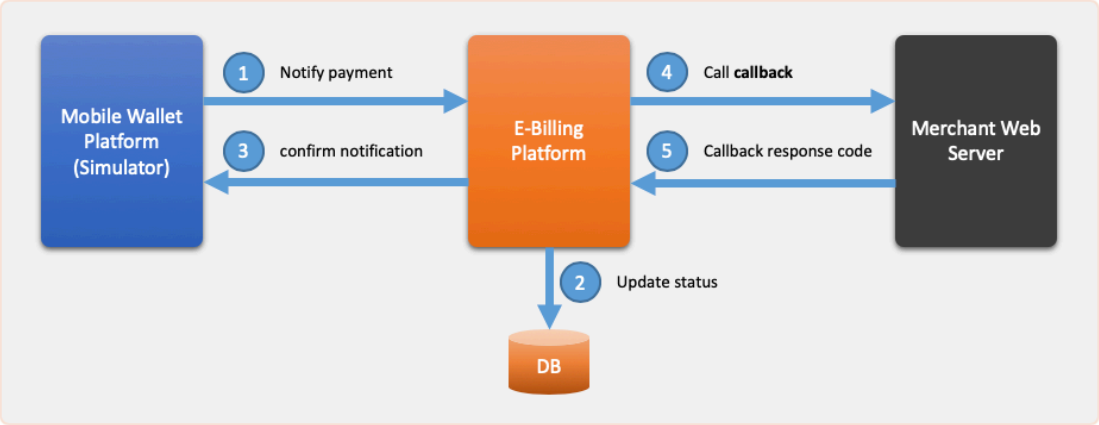
On the screen that follows, please select whatever operator you want and click on "**Pay invoice**"



When you click on "**Pay invoice**", you are basically simulating payment validation by wallet provider. The simulator is connected to LAB platform. When LAB receives payment confirmation, it will immediately notify merchant by calling **callback** registered in its profile.



Above screen indicates LAB platform has successfully receive payment notification from payment system (simulator).



When payment platform (simulator) notifies E-Billing (1), E-Billing updates status of transaction in the system (2) and confirms sends confirmation back to payment system (3) then E-Billing sends notification to merchant using registered **callback** (4) and E-Billing expects merchant's callback to return HTTP 200 (or 2xx) to flag transaction as processed (5).

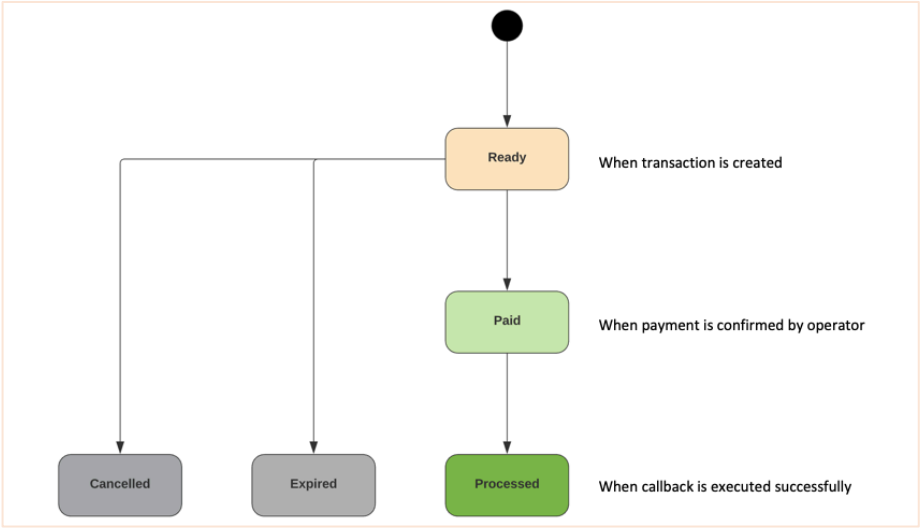
At this point, you should at your backend system to ensure that callback was successfully executed. Normally you tested successfully your callback individually with simulator, there should be no surprise with the end-2-end test.

It is very important to ensure that the HTTP code returned by callback matches the real executed state:

- HTTP 200 if execution was successful;
- HTTP 4xx if execution failed;

This will ensure that E-Billing flags state of transaction correctly.

Below is life cycle of transaction in E-Billing system:

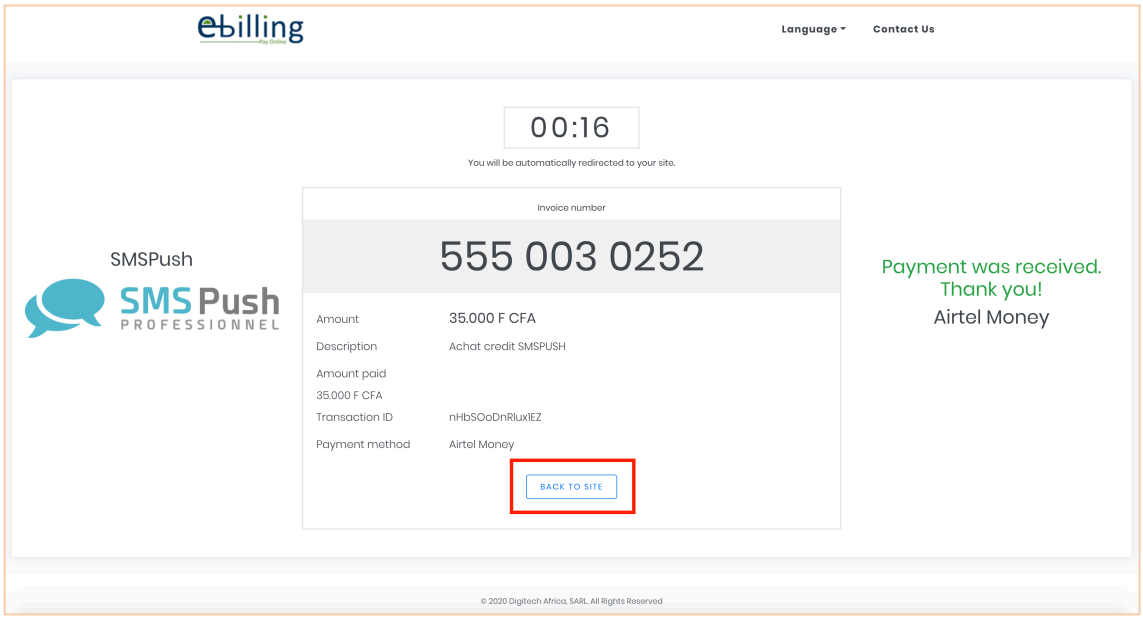


If there is failure with **callback** execution, the transaction will remain in **Paid** state

4.4 Create success page

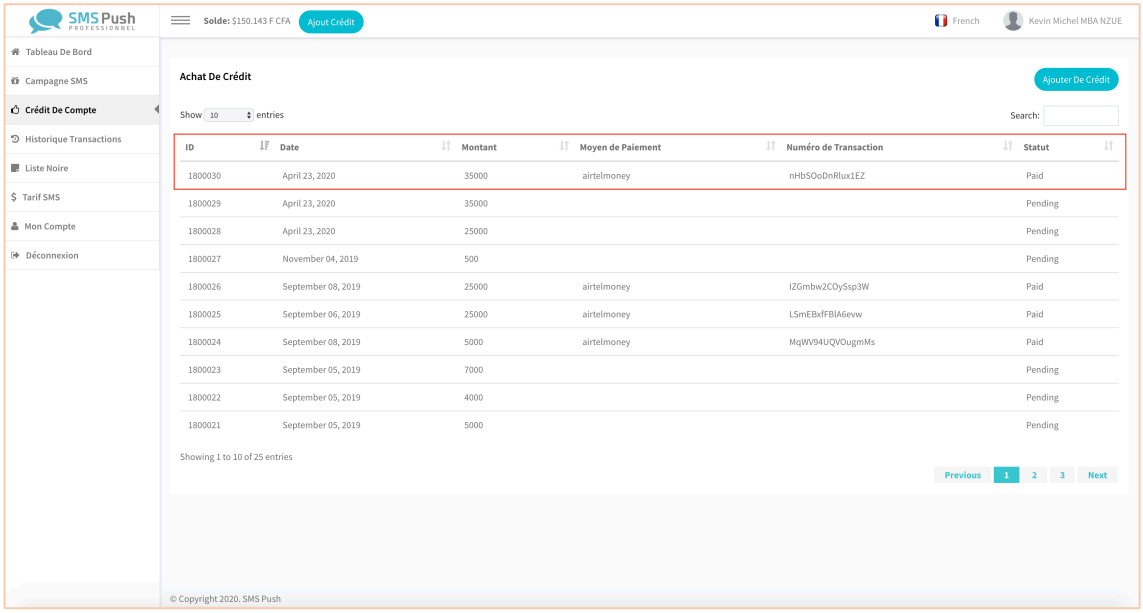
Similar to callback, success page is the physical URL customer where customer will be redirected after payment was validated by 3rd party payment system.

Here is E-Billing portal when payment is validated:



You can notice "**BACK TO SITE**" which provides a link to go back to merchant web site and more precisely to the success payment page.

Here is sample success page for SMSPUSH website:

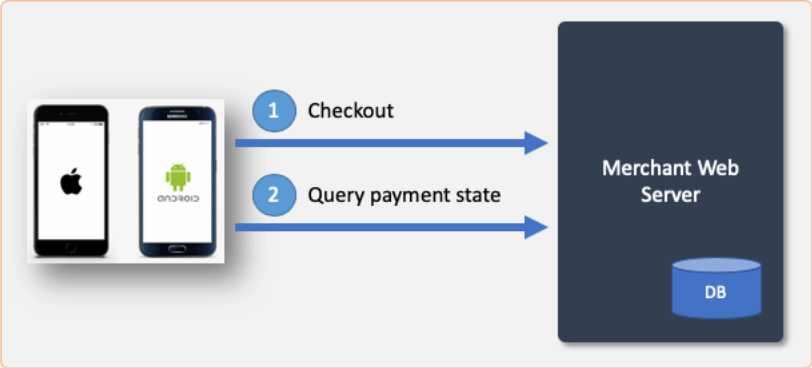


4.5 Additional activities for native mobile application

If you are integrating payment into **native** mobile application (iOS, Android), all previous activities are applicable but on top of that, there are extra tasks in relation to E-Billing integration, we are pretty sure these are same for any other payment gateway like PayPal.

As mentioned in introduction, mobile apps are not interacting directly with E-Billing. Instead, they are interacting with their backend server, usually a web server.

4.5.1 Provide API



There are 100 ways you can achieve that of course. Our recommendation would be to provide at least 2 APIs:

(1) **Checkout**: this API would be used by your mobile apps to initiate payment transaction;
(2) **Query**: this API would be used to query status of payment;

4.5.2 URI scheme or deep linking

Same as you do for PayPal, customer has to leave the native mobile apps and be directed to payment page of E-Billing. After payment, we need a way to bring user back to the application.

You can achieve that by creating URI scheme, for instance **myappli-success://** and use deep linking.

You will basically provide the URI scheme to E-Billing portal as a redirect URL when you call E-Billing to initiate transaction.

https://blog.branch.io/how-to-open-an-android-app-from-the-browser/

https://stackoverflow.com/questions/25883113/open-ios-app-from-browser

5 Evidence of successful integration in LAB

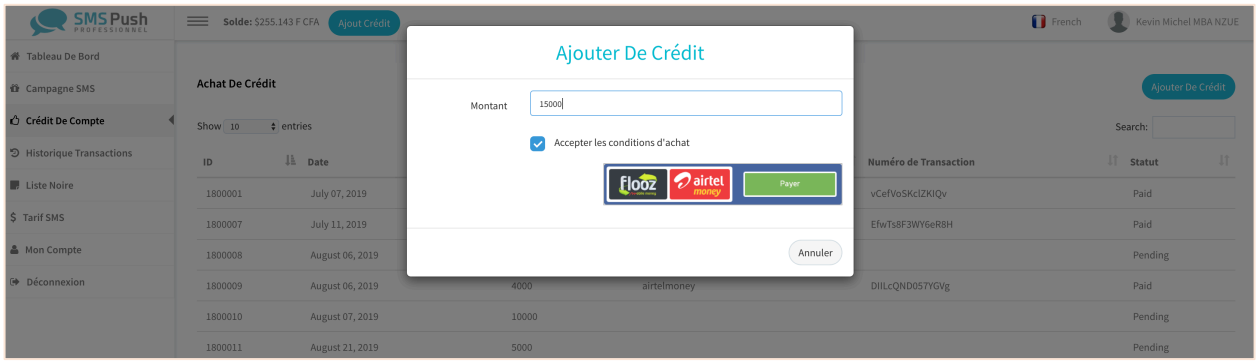
Before you can move in E-Billing production environment, you have to show evidence that your integration was successful in LAB environment.

As a reminder, LAB is functionally identical to production. If your integration is done successfully in LAB, you have the warranty it will work in production. Moving to production just consist changing URLs and credentials.

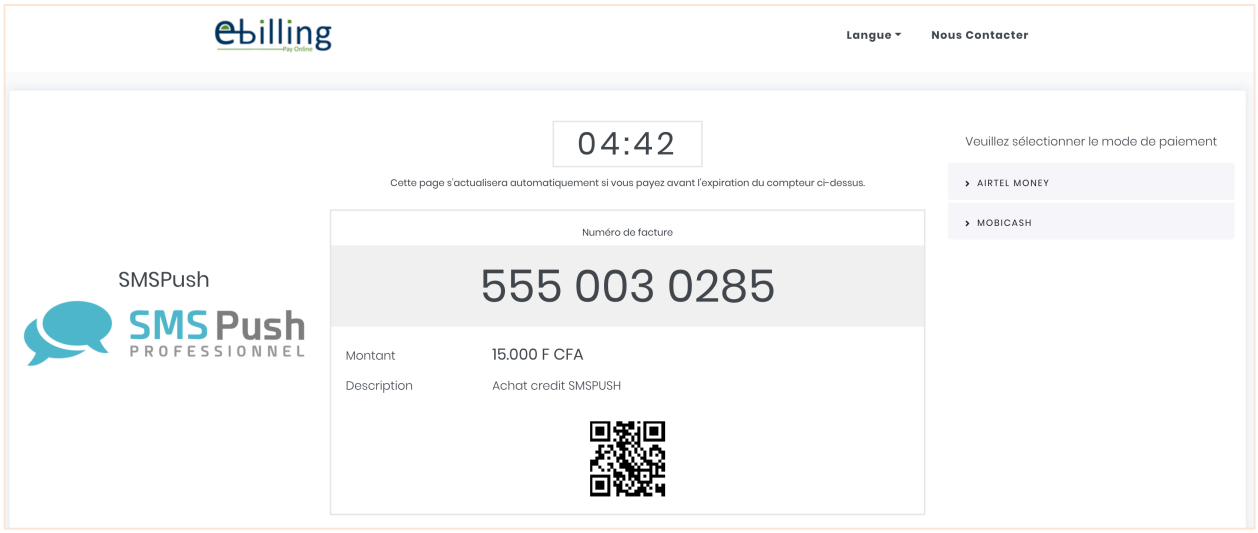
Therefore, please take your time to run all tests in LAB and do not cut corners, you can do as many tests as needed to ensure everything works well.

Below are minimum evidences to be provided that integration was successful in LAB with SMSPUSH application:

(1) Copy of screen of checkout page: customer initiates transaction

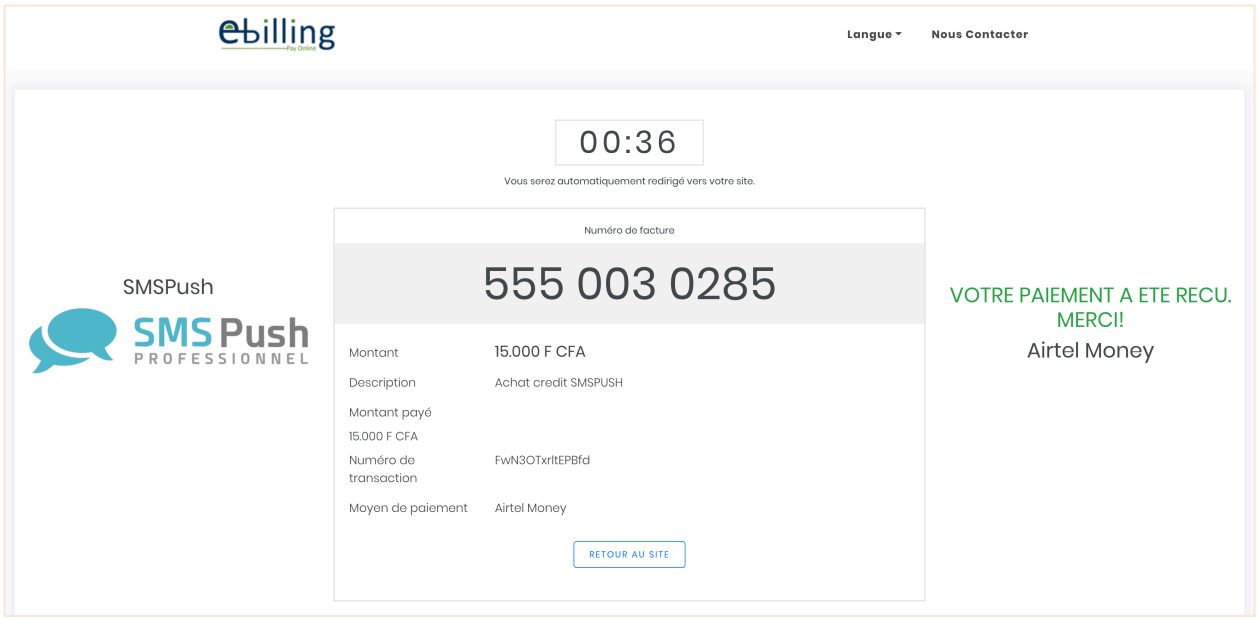


(2) Copy of screen of E-Billing portal: to show customer is redirected to payment page



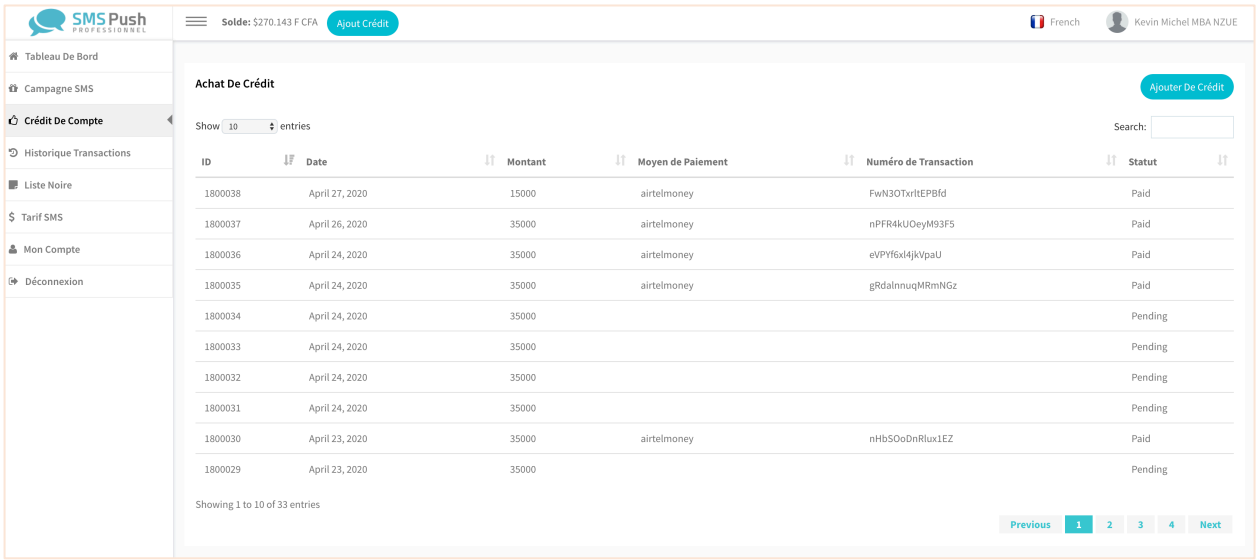
From here, use simulator to pay transaction.

(3) Copy of screen showing payment was received by E-Billing



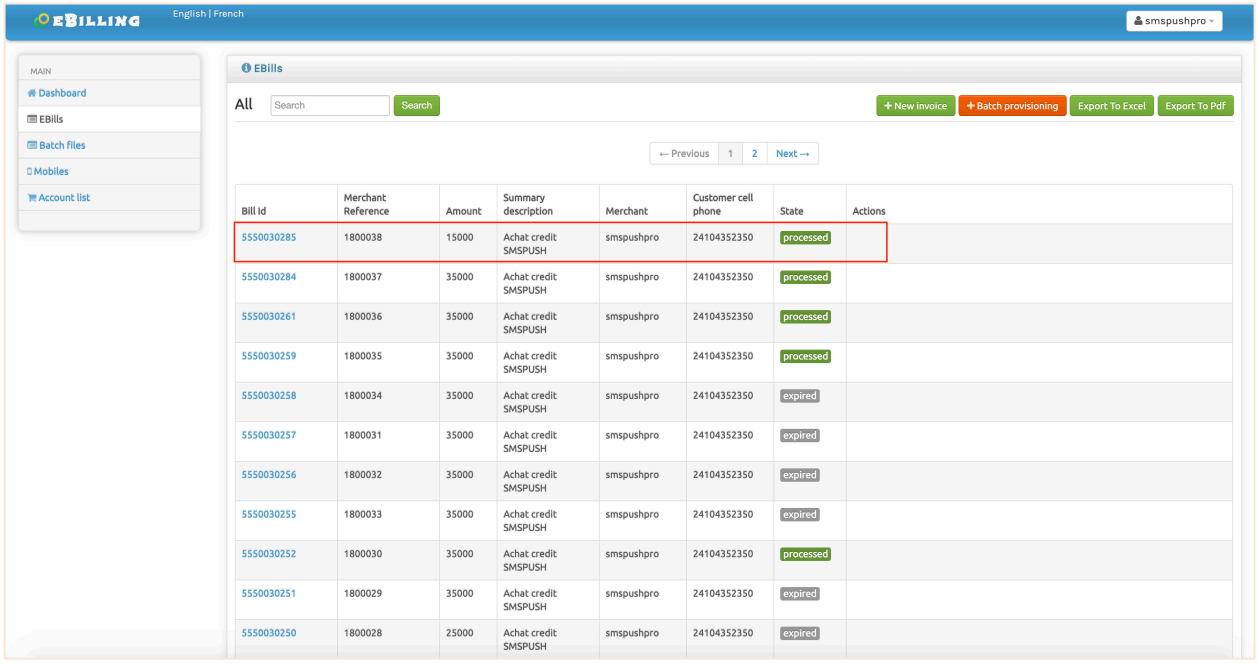
After counter expiration, customer should be redirected to merchant success page.

(4) Copy of success merchant page



Here customer can see on merchant web site that his payment was validated.

(5) Copy of screen of transaction status in E-Billing LAB to show its state is **Processed**



6 WooCommerce Plugin

For those using WooCommerce for their online store, the official E-Billing plugin is now available. The plugin automates all integration steps from **Checkout** button to redirecting customer to payment success page.

Here is the link to E-Billing Gateway Payment plugin for WooCommerce:

https://wordpress.org/plugins/e-billing-payment-gateway/

You will find all details to install and configure the plugin.