Safaricom Daraja API Tutorial

Learn how to easily integrate into the new REST API with step by step tutorials.

The feature packed, simpler and more intuitive Proxy API is now out and available to all! **Find out more** (https://proxyapi.co.ke/).

Introduction

Welcome to the Safaricom API Tutorials. The main intention of this tutorial is try to explain the ins and outs of the API in more detail than the site's documentation (https://developer.safaricom.co.ke/docs).

Ok, first, you need to open an account with them. That's pretty straight forward. They also have the option of Company management whereby you can be a member of a team. Haven't tried that out, but you can check it out (https://developer.safaricom.co.ke/api_company/companies/list). The API is designed to work exclusively over the Internet, so no need for VPN setups or IP whitelisting compared to the previous Soap API.

Once you have created an account and accessed your account, you will be given access to five main menu items. Allow me to expound on some of them:

Docs (https://developer.safaricom.co.ke/docs)

This is where you will find the Official site documentation.

APIs (https://developer.safaricom.co.ke/apis-explorer)

This is where you will find the list of all available APIs provided.

My Apps (https://developer.safaricom.co.ke/user/me/apps)

This holds a list of all applications you have created. Apps are a way of grouping APIs into a single collection for easier classification on the user's side. You shall need at least one app on the portal to use the APIs.

Go Live (https://developer.safaricom.co.ke/production_profile/form_production_profile)

This is the link that will enable you to begin the process to take your application live. It has quite some hustle, but it’s the only way available under your control.

Blog (https://developer.safaricom.co.ke/blog)

Not sure what this does or is supposed to do, since it is blank. But am guessing a Blog is supposed to be here...?

UPDATE: They just added their first post (May 07, 2019), hehehe.

UPDATE 2: This is no longer available.

Forums (https://developer.safaricom.co.ke/forum)

This is where you get support from the API team when you are stuck. You can ask questions and they can give you the answers you need.

UPDATE: This is no longer available too.

Below are some prerequisites I believe are needed for those planning to use the API:

- Strong knowledge of a general programming language e.g. Java, Python, PHP for making the API requests. You should be comfortable enough to make HTTP requests in that language.
- Knowledge of REST-based (Representational State Transfer) web services, and the JSON interchange format.
- Basic Knowledge of troubleshooting via TCP dumps or Log dumps. This is very useful.
- Working knowledge of Asynchronous HTTP Requests
- Access to a publicly available web server for receiving HTTP requests from M-Pesa.
- If you are planning to go live using the API, an **existing** pay bill number with all KYC details (especially Phone Number and Email) properly filled in and available on the M-Pesa Org (https://org.ke.m-pesa.com/) portal.

API Apps

Now, assuming you have access to the portal, first thing you do is create an app. The app you create will determine the APIs you use e.g. if you create an app and assign it Lipa Na M-Pesa Online product only, it means that app will only apply to Lipa na M-Pesa Online API, and you will not be able to use it's keys with the M-Pesa Sandbox APIs, and vice versa. Also, beware, you need to have planned out your APIs before going live. You cannot use the same product for **both** B2C APIs and C2B APIs when going live. If you applied for a C2B shortcode/paybill, it will not be allowed to use the B2C API, and if you applied for a B2C shortcode, it will not be allowed to go live with a C2B API. But the C2B API and Lipa Na M-Pesa API can be used by the same App. Also beware the name of the app will be used to identify your app in the System and will be the main identifier for all enquiries about APIs from the team. The API also has a little known analytics page, where you can view the performance of your app. This is under the **Your App** section on the **My Apps** page.

For the APIs, they are divided amongst the products as shared below:

Lipa Na M-Pesa Online

- Lipa Na MPESA Online

M-Pesa Sandbox

- Reversal
- Transaction Status
- Account Balance
- B2C
- B2B
- C2B

The OAuth API is shared among all APIs.

For each App you create, there will be a set of two parameters that will tie a request to the app as mentioned before: the Consumer Key and the Consumer Secret. They do not have an expiry on testbed, but for prod, not so sure (Can someone confirm this for me?). These will be used in the API specifically applied for when creating the Application. Beware the app credentials are in no way tied to the M-Pesa credentials, which are a separate set of credentials used in the API requests. Will explain more of that as I go along.

Now, for the setup. We are going to go through each API, and I am going to try and extrapolate the details as much as I can. We are going to use the below details in the API requests, hopefully they are active by the time you read this. You can get the M-Pesa Sandbox credentials from the Credentials page located here (https://developer.safaricom.co.ke/test_credentials). I think they change after some time when you are inactive, and they also have an expiry date, so in case the below do not work, just use your own test credentials. If you change they will be saved locally on your browser for easy access next time you visit.

- Sandbox Consumer Key: **SPLwLd2uA3onoPRXCJF6bWqWGxNvA8BZ**
- Sandbox Consumer Secret: **6WOghMAGTuEYKjX3**
- Sandbox Shortcode 1: **601426**

Sandbox Initiator Name (Shortcode 1): **apitest361**

Sandbox Security Credential (Shortcode 1): **361reset**

Sandbox Shortcode 2: **600000**

Sandbox Test MSISDN: **254708374149**

Lipa Na MPESA Online Consumer Key: **A6U3ryasDir1GCozVKKefdtVGu53xz0J**

Lipa Na MPESA Online Consumer Secret: **ZPWGWrJzsbW9ef6Y**

Lipa Na MPESA Online Shortcode: **174379**

Lipa Na MPESA Online Passkey:
bfb279f9aa9bdbcf158e97dd71a467cd2e0c893059b10f78e6b72ada1ed2c919

Change credentials

For those not familiar with M-Pesa APIs, a short intro is needed:

All M-Pesa requests (except Register URL API) are of an asynchronous nature. This means that you will not receive a result showing completion of the transaction immediately after making the request. Instead, you make the request, receive an acknowledgement, and wait for feedback via a **Listener / Callback URL / Webhook** (henceforth identified as the Callback URL). Thus, you first make a request, then, if your request passes all checks, you get an **Acknowledgement** (which is **not** the result of the transaction) that the request has been received for processing asynchronously. Then, after processing has completed, you get feedback via the callback URL which you need to have specified beforehand via two means: registration of the URLs on the system (used by C2B API only), or preset in your initial request (used by the other APIs). Depending on the transaction type and the outcome of the transaction, the result can either be a success or failure. So don't mistake the acknowledgement (which is the response received after you make the request) for a successful transaction.

Let me try to use a real-world scenario to make the above more clearer. You may skip this section if you already understand the above.

Imagine you are a student currently sitting for an examination. You are required to fill in an exam paper, and hand it over for marking when you are done. Now, when you are done with the paper, you want to know how much you have scored, and the teacher needs to mark the paper before they can tell you your score. The teacher will most probably not mark the paper immediately, but will instead tell you that you shall hand in the paper and he/she shall give you your results after marking them later (after their tea break, of course). The teacher will then first confirm that all required questions have been answered, and that at least you have put your own name on the paper. After accepting and marking the paper, the teacher notifies you by either calling you to the office, sending someone to fetch you, sending a messenger to deliver the marked paper to you, or any other method applicable, with your result on it. Then you shall know whether you passed or failed.

Applying the above to our situation, the paper you are sitting for represents the request you are sending to Safaricom. The teacher is the API/M-Pesa, and the communication between you and the teacher represents the HTTP requests between your system and the API. When you send a request to the API, it sends back an Acknowledgement, telling you that your request is received. This is analogous to the teacher telling you "OK, I have received your paper, let me mark it and send the results back to you later". The action of the teacher checking that you have answered all questions represents the API checking your request before accepting it. If there are missing parameters, or other errors in your request structure, the API immediately informs you of the error, and you are left to correct the errors and trying again. Until all parameters are correctly filled, or the structure fits the required format, the API will not accept your request. The teacher sending the results back to you using whichever method represents the API calling your system back via a normal HTTP POST Request using the pre-defined URL and delivering the results to you. The results tell you the outcome of your request (failed or succeeded) and any extra information necessary if required.

Hope that helps.

Generate Token

This API generates the tokens for authenticating your API calls. This is the first API you need to engage with in the set of APIs available as all the other APIs require the information from this API to work.

The API works as below:

1. Get the Base-64 encoding of **Consumer Key + ":" + Consumer Secret** (note the full colon in the encoding, and without the quotes)

2. Create a `GET` request and set an Authentication header with the value as Basic + encoded value from above step e.g. using the Consumer Credentials above, the header will be `Authorization: Basic`

3. Send the request to the endpoint `https://sandbox.safaricom.co.ke/oauth/v1/generate?grant_type=client_credentials`. The raw request will look similar to the following :

```
GET /oauth/v1/generate?grant_type=client_credentials HTTP/1.1
Host: sandbox.safaricom.co.ke
Authorization: Basic U1BMd0xkMnVBM29ub1BSWENKRjZiV3FXR3hOdkE4Qlo6NldPZ2hNQUdUdUVZS2pYMw==
Content-Type: application/json
```

4. You will get a response in the sample format shown below. This shows your token and how long it will take before it expires.

```
{
    "access_token": "[access_token]",
    "expires_in": "3599"
}
```

Once generated, you somehow need to keep track of the timeout period so it does not expire unknowingly. This means you need to either generate a new token for every request you make, or keep a timer variable somewhere and keep checking it. That's up to you. But know that when you get the "Invalid Access Token" error, your Auth token has expired or is not set. Get a new one.

The panel on the right, I call it a playground, is designed to help you test out the APIs as we go through them. It shall consist of multiple tabs. At the minimum, it shall have a "Try it" tab for testing the specific API which you are reading on, and a quick access "Refresh Credentials" tab for when the token expires in the middle of a session to quickly generate a new token. If the panel is not completely visible, it means you may be viewing on a 4:3 ratio monitor. Try zooming out to around 80-90%, to be able to view the scrollbars. There is also a panel showing any callbacks that may be received from using the testers, accessible by clicking on the ✎ icon at the top-right corner of the screen. The latest received callbacks will be displayed there whenever available **if you have not changed the default callback preset in the requests**, ordered by latest callbacks first. It will record all callbacks received, unfiltered for now, so you may need to dig around for your callback from the incoming requests. Use a unique Identifier to try and capture yours on the log.

Generate Sandbox Token

Try It

Base-64 Encoded Credential

Basic U1BMd0xkMnVBM29ub1BSWENKRjZiV3FXR3h(

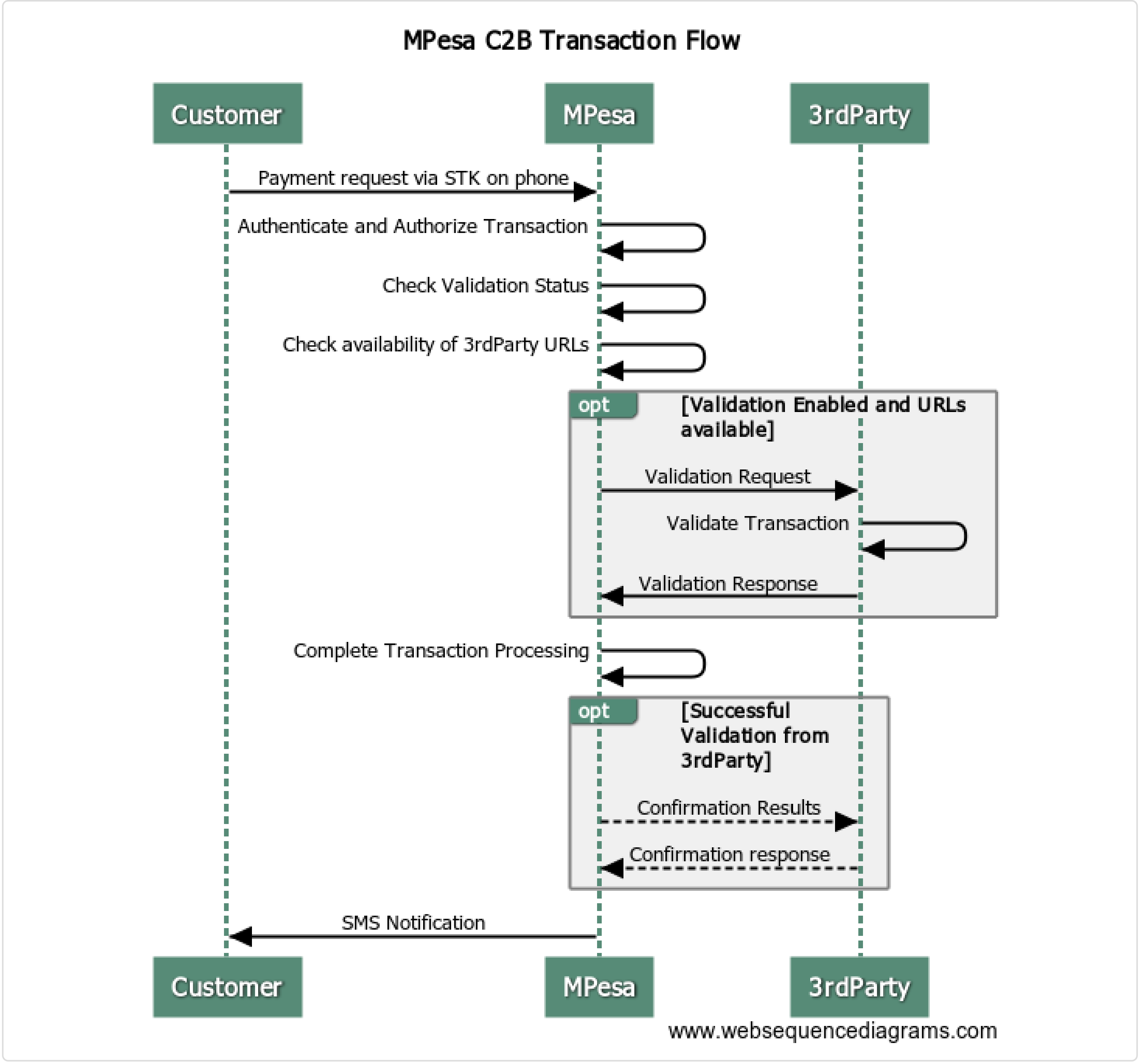
Send Request

Response

Register URL

The first API we shall look at is the Register URL API under Consumer to Business (C2B) APIs. It is the first half of the C2B API for receiving payment notifications to your paybill. This API enables you to register the callback URLs via which you shall receive payment notifications for payments to your paybill/till number. The URLs are used by the C2B payment simulation API for sending the transaction details to you.

There are two URLs required for RegisterURL API: **Validation URL** and **Confirmation URL**. The reason for two of them is due to the nature of C2B API calls. Below is a flowchart of the C2B process:


MPesa C2B Transaction Flow
Customer
MPesa
3rdParty
Payment request via STK on phone
Authenticate and Authorize Transaction
Check Validation Status
Check availability of 3rdParty URLs
opt
[Validation Enabled and URLs available]
Validation Request
Validate Transaction
Validation Response
Complete Transaction Processing
opt
[Successful Validation from 3rdParty]
Confirmation Results
Confirmation response
SMS Notification
Customer
MPesa
3rdParty
www.websequencediagrams.com

C2B transaction flow

The process is explained as below:

1. A customer sends a payment request to your paybill from their phone
2. M-Pesa receives the request and validates it internally first
3. M-Pesa then checks if you have enabled **External Validation** for the paybill receiving the request
4. If External Validation is **enabled**:
   a. M-Pesa first sends a **Validation request** to the Validation URL registered in the system (3rd party) with the payment details.
   b. The 3rd Party validates the request and sends an appropriate response to M-Pesa. This response must be received within a given time period or M-Pesa marks the endpoint system as unreachable. The response informs M-Pesa to either complete or cancel the payment:
   c. M-Pesa receives the response and processes the transaction accordingly:
   d. If you had chosen to **complete** the transaction, M-Pesa sends a **Confirmation request** to your Confirmation URL with the details of the completed transaction. The transaction is then complete. Thus you shall receive **two** API calls on your system.
   e. If you had chosen to **cancel** the payment, M-Pesa simply cancels the transaction and no other request is sent. The transaction is then complete
5. If External Validation is **disabled**, M-Pesa automatically completes the transaction, and if the transaction is a success, M-Pesa sends a **Confirmation** request to the Confirmation URL registered in the system. This is the

only API call you shall receive on your end.

6. If External Validation is **enabled**, but for some reason M-Pesa could **not** reach your endpoint to validate the transaction within the stipulated time period (usually < 8 seconds from the moment the request leaves -Pesa), or no response was received by the time M-Pesa terminates the request, it checks on the default action value saved during registration of the URLs. If the default action was set to **Completed**, M-Pesa automatically completes the transaction and also tries to send a Confirmation request to your other endpoint. If the default action was set to **Cancelled**, M-Pesa simply cancels the transaction and no Confirmation callbacks are sent. The transaction is then complete.
7. If no URLs are registered in the system, M-Pesa automatically completes the request.
8. M-Pesa then sends an SMS notification to both the customer and paybill owner with the results of the transaction as usual.
9. When the external notifications fail to be sent, you can check on the M-Pesa Org portal and cross-check against received callbacks. The portal has all the payments ever made available, whether you received the callback or not. Manual, but necessary once in a while.

For the two URLs, below are some pointers. These will also apply to the Callback URLs we shall use later on in other APIs:

- Use publicly available (Internet-accessible) IP addresses or domain names.

- **Do not** use the words **MPesa, M-Pesa, Safaricom** or any of their variants in either upper or lower cases in your URLs, the system filters these URLs out and blocks them. Of course any Localhost URL will be refused.

- **Do not** use public URL testers e.g. ngrok (https://ngrok.com/), mockbin (http://mockbin.org/) or requestbin (https://requestb.in/) **especially** on production, they are also usually blocked by the API.

NB: C2B Transaction Validation is an optional feature that needs to be activated on M-Pesa, the owner of the shortcode needs to make this request for activation to the M-Pesa Support or API Support team if they need their transactions validated before execution.

The request structure will be as shown below:

```
// URL
[POST] https://sandbox.safaricom.co.ke/mpesa/c2b/v1/registerurl

// HEADERS
Host: sandbox.safaricom.co.ke
Authorization: Bearer [access token]
Content-Type: application/json

// BODY
{
    "ShortCode": "601426",
    "ResponseType": "[Cancelled/Completed]",
    "ConfirmationURL": "[confirmation URL]",
    "ValidationURL": "[validation URL]"
}
```

The other parameters are:

Shortcode
This is your C2B-enabled paybill number/till number, which you expect to receive payments notifications about.

Response Type
This is the default action value that determines what M-Pesa will do in the scenario that your Validation endpoint is unreachable or is unable to respond on time. Only two values are allowed: **Completed** or **Cancelled**. Completed

means M-Pesa will automatically complete your transaction, whereas Cancelled means M-Pesa will automatically cancel the transaction.

Then a success response will look like the one below:

```
{
    "ConversationID": "",
    "OriginatorCoversationID": "",
    "ResponseDescription": "success"
}
```

Anything else other than the above means there is an error, check error description and fix it if it's your error. I will cover some common errors encountered on the portal later on in a different section as I compile them one by one from the APIs.

Unfortunately, there is no way to check which URLs are currently registered on the system, or whether the URLs were actually registered in the first place. You can only confirm by performing C2B requests and checking if the registered endpoints are hit or not. Otherwise you can get in touch with the support team and enquire which URLs are currently under your paybill on the system

Also, it is good to know the relationship between short codes and the URLs. One paybill can only have one pair of URLs under it, but the same pair of URLs can be used by multiple paybill numbers for transaction notifications. That means it will be up to you to differentiate requests for different paybills.

NB: For this API,and the C2B API below, please test using your own test paybill assigned to you. Most probably, by the time you register your URLs with my paybill above, then get down to testing on the C2B section, someone else will have overwritten your URLs with their own URLs. Then you will be left scratching your head as to why "M-Pesa is not calling my endpoints".

And that's pretty much it for Register URL API.

Register URL

Try It | Refresh Token

Access Token

Bearer [access token]

Shortcode

601426

Response Type

Cancelled

Validation URL

https://peternjeru.co.ke/safdaraja/api/validation.php

Confirmation URL

C2B API

This API is used to simulate payment requests from clients and to your API. It basically simulates a payment made from the client phone's STK/SIM Toolkit menu, and enables you to receive the payment requests in real time. It is the second half of the Register URL API covered above, and requires the URLs registered by the previous API to work. Hopefully you have gone through the previous section to understand how the C2B payment process works, as both are tied to each other. If not, please do, I can wait...

Welcome back. Moving on, the basic C2B request looks like the sample below:

```
// URL
[POST] https://sandbox.safaricom.co.ke/mpesa/c2b/v1/simulate

// HEADERS
Host: sandbox.safaricom.co.ke
Authorization: Bearer [access token]
Content-Type: application/json

// BODY
{
    "ShortCode": "601426",
    "CommandID": "CustomerPayBillOnline",
    "Amount": "100",
    "Msisdn": "254708374149",
    "BillRefNumber": "account"
}
```

These are explained below:

Shortcode
This is your paybill number/till number, which you expect to receive payments notifications about.

CommandID
This is a transaction type identifier, which Identifies the type of C2B transaction being made. There are two options: **CustomerPayBillOnline** and **CustomerBuyGoodsOnline**. CustomerPayBillOnline is used when simulating Pay Bill requests, which the customer does by going to the `M-PESA -> Lipa na M-PESA -> Pay Bill` option on mobile, and uses a Paybill/Store number. Thus it requires an Account number/Bill reference number for it to be valid. CustomerBuyGoodsOnline is used for Buy Goods simulation, which is when a customer goes to `M-PESA -> Lipa na M-PESA -> Buy Goods and Services` on their mobile phones and uses a Till number. This does not require a paybill number, and putting one actually causes the request to fail. Beware that you cannot use a Till number with CustomerPayBillOnline command ID and you cannot use a Pay Bill/Store number with CustomerBuyGoodsOnline Command ID. As of now, I can only see the paybill number provided on the portal, there is no Till number, so the CustomerBuyGoodsOnline command ID might not work as expected, and you might not receive callbacks if you use it since the payment might automatically fail on the backend. But am still going to provision for it here.

Msisdn
This is the test phone number of the virtual customer paying to your paybill. Use the one given in your test credentials (https://developer.safaricom.co.ke/test_credentials) section (Test MSISDN) to test. Testing using your own phone numbers will not work with this API since the numbers need to be registered on the testbed first. The number should either begin with 07XX or 2547XX (for now). International numbers are not supported.

BillRefNumber
This simulates the account number that a user would have entered when making a Pay Bill request. This parameter is only required for CustomerPayBillOnline transaction type. It is sent as part of the validation and confirmation requests to you (3rd party) to validate and confirm. It has a maximum of 20 characters.

After sending the request, and assuming you have all the correct details, the success response should be the acknowledgement below:

```
{
    "ConversationID": "AG_20180324_000066530b914eee3f85",
    "OriginatorCoversationID": "25344-885903-1",
    "ResponseDescription": "Accept the service request successfully."
}
```

The above response shows that your request has been accepted, and will be processed. The two 'ConversationID's are unique identifiers of your transaction's journey on M-Pesa. M-Pesa itself processes transaction in <~8 seconds, thus, adding 1-2 seconds of network delays and communication, it should take no more than 10 seconds for the validation/confirmation request to hit your endpoint. If it is consistently above that, please check for network or processing delays around your system.

For those who have enabled External Validation, you will first receive a validation request on your Validation URL to validate the request. That request will have the structure below:

NB: For those unfamiliar with callbacks, all API callbacks from transactional requests are `POST` requests, do not expect `GET` requests for callbacks. Also, the data is not formatted into `application/x-www-form-urlencoded` format, it is `application/json`, so do not expect the data in the usual `POST` fields/variables of your language, read the results directly from the incoming input stream.

```
[POST] https://yourdomain.co.ke/your/validation/url

// HEADERS
Content-Type: application/json

// BODY
{
    "TransactionType": "",
    "TransID": "LHG31AA5TX",
    "TransTime": "20170816190243",
    "TransAmount": "200.00",
    "BusinessShortCode": "601426",
    "BillRefNumber": "account",
    "InvoiceNumber": "",
    "OrgAccountBalance": "",
    "ThirdPartyTransID": "",
    "MSISDN": "254708374149",
    "FirstName": "John",
    "MiddleName": "",
    "LastName": "Doe"
}
```

TransID
This is M-Pesa's unique transaction identifier for your transaction. This can be used for searching for the transaction later on using the Transaction Query API.

TransTime
Simply the time the transaction was completed on M-Pesa in the format **YYYYMMddHHmmss** (https://en.wikipedia.org/wiki/Date_format_by_country) (https://en.wikipedia.org/wiki/Date_format_by_country).

TransAmount
The amount transacted by the customer when paying to your paybill/till.

BusinessShortCode
The shortcode to which the customer paid to. This can be used to differentiate payments to different paybills via the same notification URLs.

BillRefNumber
The account number the customer entered on their phone when making the payment. Applicable to PayBill requests.

MSISDN
The phone number from which the payment was made.

FirstName, MiddleName, LastName
The names of the customer under whom the MSISDN above is registered. The First Name and Last Name are usually mandatory. The Middle Name is optional.

After receiving the request, you are supposed to process it and respond to the API call with either an **accept** or **reject** response. To accept, you send the below JSON making sure the value of **ResultCode** is 0 (zero, **must** be a literal Integer, not a String like "0"), but the value of ResultDesc can be any alphanumeric value.

```
{
    "ResultCode": 0,
    "ResultDesc": "Accepted"
}
```

To reject a transaction, you send the same JSON above, but with the value of ResultCode being any integer **EXCEPT** 0, as shown below

```
{
    "ResultCode": 1,
    "ResultDesc": "Rejected"
}
```

So, basically, sending a ResultCode value of 0 means you accept the transaction, and sending anything else rejects the transaction. Values below 0 are not accepted, and also constitute a rejection.

If transaction has been accepted, M-Pesa will complete the transaction and send a Confirmation request to you. This will have the same structure and values as the Validation request JSON above. This also applies for those who have disabled External Validation. You may respond to the Confirmation request with the JSON below. If you cancel the transaction, no Confirmation will be sent to you. Beware that you cannot cancel a transaction after the Confirmation request has been sent to you. Confirmation marks the completion of the transaction on M-Pesa.

```
    {
        "C2BPaymentConfirmationResult": "Success"
    }
```

That marks the end of the C2B transaction. If you find out that M-Pesa is not getting to you, or that your systems are not receiving payment notifications, you can always fall back to the M-Pesa Org Portal to confirm the transactions were received and processed as required.

Caveat: please disconnect your shortcode from the old Soap API before using it on the new API if you are planning to migrate the shortcode. The settings from the old API could really mess up your integration in the new API. Make sure to delete the registered C2B URLS from the old System by placing a request with the Support team to have them deleted. This will also apply to the Lipa Na M-Pesa API.

C2B API

Try It   Refresh Token

Access Token

Bearer [access token]

Shortcode

601426

Command ID

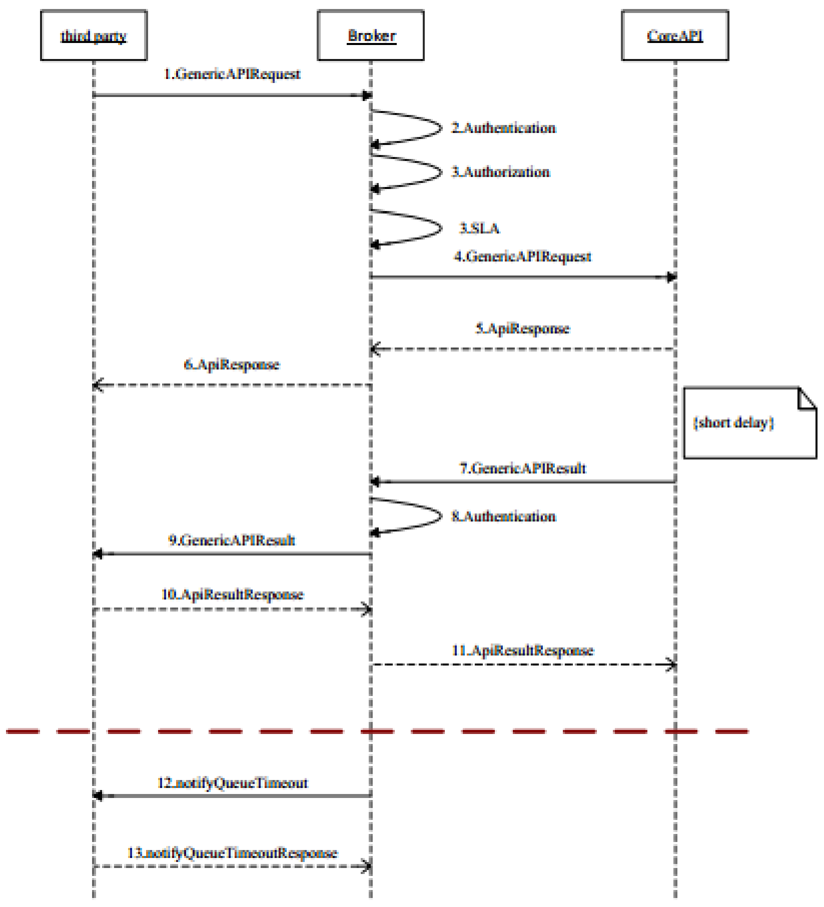
CustomerPayBillOnline ▼

Amount

500

B2C API

Also known as the Bulk Payment API, the Business to Consumer (B2C) API enables a Business or Organization to pay their customers for a variety of reasons. The most common reasons a business can pay their customer include salary payments, promotion payments (e.g. betting winnings payouts) or normal business payments (e.g. transfers from bank to mobile accounts). Each of these scenarios has their own unique characteristics, but all lie under the B2C API category. Below is the flow of a B2C transaction.



B2C transaction flow

1. The Business sets the data in the request and sends it
2. M-Pesa received the request and validates it internally first, then sends you an acknowledgement response.

3. M-Pesa processes the request, then sends the response back to you via the callback URL specified in your initial request.
4. As of now, there are no repeat callbacks for failed callbacks, thus if M-Pesa is unable to send the callback to you, you will need to confirm from the portal as to the status of the request or use the Transaction Status API to get the result.

The B2C API has quite some hustle to understanding how to make it work, thus I shall explain it from the point of perspective of a client who has access to the M-Pesa Organization Portal, since we shall touch on that portal quite frequently from here on. But I shall make sure to expound properly for those with no access to the portal.

For one to be able to perform operations on the M-Pesa system, one must have access to it. Access to the system is mainly based on your identity, i.e. you are either a **Customer**, an **Organization** or **Service Provider (SP)**. The SP is Safaricom, and they have full control over the other two entities. The customer is obviously the client registered on M-Pesa, who can perform the transactions allowed to them via phone or other means, and are identified by their phone number/MSISDN (https://en.wikipedia.org/wiki/MSISDN). The Organization is the Business, Company e.t.c. who are identified by the shortcode they applied for, and want to perform operations or transactions on the system.

3rd Party access to M-Pesa is via three channels: **Web**, **API** and **Handset**. Customers mainly use the handset channel, Organizations take the Web and API channel. For Organizations (Orgs from here on), the access channels give rise to two classes of users: **Web Operators** and **API operators**. Web operators can access M-Pesa via the M-Pesa Portal, commonly known as the M-Pesa Org. Portal, to perform their transactions and operations there. They can also perform operations or transactions via API given the right roles. API operators, on the other hand, access the M-Pesa system via API calls. They can perform the operations/transactions assigned to them only via API calls, and cannot log into the system via the web portal. Also, Web operators have control over API operators.

Both types of operators are limited to their actions on the portal via roles assigned to them by the SP (RBAC (https://en.wikipedia.org/wiki/Role-based_access_control)). On a typical 3rd Party M-Pesa Org portal, there are at least three roles: **Business Administrator**, **Business Manager**, and **Org API operator**. The Business Administrator is the overall admin of the account, and controls other users on the portal and can perform bulk operations for their organization. The Business Manager mainly controls the financial aspects of the system, including transactions overview, authorization, perform transactions and reverse them too via portal e.t.c. The Business Admin and Business Manager are both Web operators i.e. they can only access the system via the web portal. The Org API Operator is the user allowed to perform transactions/actions over API calls. This is the user who is used in the B2C calls on the API portal. The actions they can perform are limited to the permissions assigned to them by their Business Administrator. The operator is identified as the **Initiator** in API calls. At the minimum, the below are the permissions assignable to them and the actions/transactions they perform (note: the case might be different):

ORG B2C API initiator
This enables the API operator to perform B2C and B2B transactions via API.

Transaction Status Query Org API
This enables the API operator to query for the status of transactions performed previously by the same operator via API. An operator can only query for their own transactions which they perfomed.

Balance Query Org API
This enables the API operator to query the balance of their Organization's M-Pesa accounts via API.

Note, the Org's Business Admin requires the **Set Restricted Org API Password** permission for them to set passwords for other users. Without that permission, you cannot set a password for other operators. This is assigned by the Support Team from M-Pesa after taking your paybill live.

This (files/org_operator_manual.pdf) document shows how to create an API operator, give them their respective permissions and assign them a password for use in API calls. The same user can also be used by C2B paybill owners to perform transaction query and balance query API calls.

So, for you to be able to perform B2C API calls on the API, you need the below at a minimum:

Initiator username: this is the API operator's username as set on the portal when the user was created. For Sandbox users, the username is already created and assigned to them and is available on the test credentials (https://developer.safaricom.co.ke/test_credentials) page as **Initiator Name (Shortcode 1)**.

Initiator Password: this is the password assigned to the API operator after being created by the Business Administrator. For Sandbox users, this is available as **Security Credential (Shortcode 1)** on the test credentials page. **Note:** the password should be limited to specific special characters such as '#', '&', '%' and '$'. Other characters might cause issues, and the password may refuse to be accepted e.g. using a '(' or ')' character will be refused. Also, '@' is not a special character on M-Pesa, it's treated as a normal character.

Public Key Certificate: this is the certificate used to encrypt the Initiator's plaintext password for use in the API calls. This is provided for both Sandbox (https://developer.safaricom.co.ke/sites/default/files/cert/cert_sandbox/cert.cer) and Production (https://developer.safaricom.co.ke/sites/default/files/cert/cert_prod/cert.cer) clients on the portal. You need to learn how to encrypt using your API language to be able to make API calls, or find a way to encrypt beforehand and set the password as a static variable on the API call. The playground offers the capability to encrypt your test password for you on the right.

These are the credentials which I mentioned are not tied to the Dev Portal Consumer credentials at all. If your API calls are accepted, but you get errors such as "Initiator Information is invalid", it means its the above M-Pesa credentials which have issues, not the API credentials. If the API credentials have any issue, you will definitely not be able to reach M-Pesa itself. But if you are able to reach M-Pesa (requests are being accepted), it means your API Consumer credentials are OK.

Back to the API. Currently the B2C API allows the org to perform around 3 types of transactions: **Salary Payments**, **Business Payments** or **Promotion payments**. Salary payments are used by organizations paying their employees via M-Pesa, Business Payments are normal business transactions to customers e.g. bank transfers to mobile, Promotion payments are payments made by organization carrying out promotional services e.g. betting companies paying out winnings to clients. Business Payments are the most common, and can be used for any of the above mentioned scenarios, but they all carry their own tariffs and configurations, so be sure which you are using beforehand.

The B2C request structure is as shown below:

```
// URL
[POST] https://sandbox.safaricom.co.ke/mpesa/b2c/v1/paymentrequest

// HEADERS
Host: sandbox.safaricom.co.ke
Authorization: Bearer [access token]
Content-Type: application/json

// BODY
{
    "InitiatorName": "apitest361",
    "SecurityCredential": "nq095ATk1fEmQBZBuOI9cWo5/w8eMkzu3nu4JWdgPa2/s7WfH0Q1gecizQLBagL+nHN3rZ9ymJ
    "CommandID": "[CommandID]",
    "Amount": "1000",
    "PartyA": "601426",
    "PartyB": "254708374149",
    "Remarks": "",
    "QueueTimeOutURL": "https://peternjeru.co.ke/safdaraja/api/callback.php" ,
    "ResultURL": "https://peternjeru.co.ke/safdaraja/api/callback.php",
    "Occassion":  ""
}
```

The parameters are explained below:

InitiatorName
The username of the API operator as assigned on the M-Pesa Org Portal.

SecurityCredential
The password of the API operator **encrypted** using the public key certificate provided.

CommandID
This specifies the type of transaction being performed. There are three allowed values on the API: **SalaryPayment**, **BusinessPayment** or **PromotionPayment**.

PartyA
This is the identifier of the Debit party of the transaction, in this case the debit party being the Organization thus the identifier being the Shortcode of the organization.

PartyB
This is the identifier of the Credit party of the transaction, here the credit party being the customer thus the identifier being the customer's phone number (beginning with 07XX or 2547XX)

Remarks, Occassion
A very short description of the transaction from your end. Occassion can be left blank, Remarks cannot be blank.

ResultURL
This is the callback URL where the results of the transaction will be sent. Please visit the API Apps section to understand how this is used if you are not familiar with it.

QueueTimeOutURL
This is the callback URL used to send an error callback when the transaction was not able to be processed by M-Pesa within a stipulated time period.

Once sent, you shall expect a success acknowledgement response from the API informing you that your request was accepted. The response format is as below:

```
{
  "ConversationID": "AG_20180326_00005ca7f7c21d608166",
  "OriginatorConversationID": "12363-1328499-6",
  "ResponseCode": "0",
  "ResponseDescription": "Accept the service request successfully."
}
```

Note the value of **ResponseCode**. Any value other than 0 (zero) means the request was unsuccessful, and the error is defined in the **ResponseDescription** element. So you need to fix that first. A value of 0 means the request was accepted by the API.

After M-Pesa completes processing the transaction, it sends back the callback via the ResultURL you specified in the initial request. A callback from M-Pesa can either be a success callback or a failure callback. A sample of a successful transaction callback is as shown below:

```
{
    "Result":
    {
        "ResultType":0,
        "ResultCode":0,
        "ResultDesc":"The service request has been accepted successfully.",
        "OriginatorConversationID":"14593-80515-2",
        "ConversationID":"AG_20170821_000049448b24712383de",
        "TransactionID":"LHL41AHJ6G",
        "ResultParameters":
        {
            "ResultParameter":
            [
                {
                    "Key":"TransactionAmount",
                    "Value":100
                },
                {
                    "Key":"TransactionReceipt",
                    "Value":"LHL41AHJ6G"
                },
                {
                    "Key":"B2CRecipientIsRegisteredCustomer",
                    "Value":"Y"
                },
                {
                    "Key":"B2CChargesPaidAccountAvailableFunds",
                    "Value":0.00
                },
                {
                    "Key":"ReceiverPartyPublicName",
                    "Value":"254708374149 - John Doe"
                },
                {
                    "Key":"TransactionCompletedDateTime",
                    "Value":"21.08.2017 12:01:59"
                },
                {
                    "Key":"B2CUtilityAccountAvailableFunds",
                    "Value":98834.00
                },
                {
                    "Key":"B2CWorkingAccountAvailableFunds",
                    "Value":100000.00
                }
            ]
        },
        "ReferenceData":
        {
            "ReferenceItem":
            {
                "Key":"QueueTimeoutURL",
                "Value":"https:\/\/internalsandbox.safaricom.co.ke\/mpesa\/b2cresults\/v1\/submit"
            }
        }
    }
}
```

Of major importance are:

ResultCode

This is a simple outcome of the transaction. A 0 (zero) standing for successful transaction, and any other code standing for an unsuccessful transaction. If not successful, the description of the error that was raised is in the **ResultDesc** element.

ResultDesc
This is a one line description of the result code received above. It explains, if unsuccessful, why a transaction failed.

TransactionID, TransactionReceipt
This is M-Pesa's unique identifier of the transaction. As long as the request was accepted by M-Pesa, you can expect the Transaction ID parameter in your callback, whether the transaction was successful or not. Note: TransactionReceipt and TransactionID both represent the same transaction, thus are the same.

TransactionAmount
This is the amount that was transferred to the customer in this transaction.

ReceiverPartyPublicName
This is a concatenation of the receiver's phone number + First name + Last name as registered on M-Pesa

B2CUtilityAccountAvailableFunds, B2CWorkingAccountAvailableFunds
Simply the current usable balances of the Organization's Utility Account and Working account respecively as at the time of the current transaction.

For an unsuccessful transaction, the response format will be:

```
{
    "Result":
    {
        "ResultType":0,
        "ResultCode":17,
        "ResultDesc":"System internal error.",
        "OriginatorConversationID":"16940-3815719-3",
        "ConversationID":"AG_20171228_00004fd3a482e7f73145",
        "TransactionID":"LLS81H3W6E",
        "ReferenceData":
        {
            "ReferenceItem":
            {
                "Key":"QueueTimeoutURL","Value":"https:\/\/internalsandbox.safaricom.co.ke\/mpesa\/b2
            }
        }
    }
}
```

Note the value of ResultCode, which gives an instant view of the status of the transaction. The cause of the error is then described in the ResultDesc element. For this specific error (System internal error), get in touch with the support team, something's wrong with the backend. Also note the transaction also has an M-Pesa Transaction ID, which identifies your transaction in the system.

That is pretty much B2C in a nutshell.

B2C API

Try It   Encrypt Test Password   Refresh Token

Access Token

Bearer [access token]

Initiator Name (API Operator's username)

apitest361

Security Credential (Initiator's Encrypted Password)

```
nq095ATk1fEmQBZBuOI9cWo5/w8eMkzu3nu4JWdgP
a2/s7WfH0Q1gecizQLBagL+nHN3rZ9ymJcvUbyGu0iK
B2oCFdpVKpte+kFJ0Val8jhKQO/VtdqR8ZXIcCDYvb3
X4ZdgbhJR+7nqCgUR5WVEzVoLgnuQopi5j+Q9QFV
WWO/KBQ6vhl01j1abm5ZQ1i1MVgI6j8pvCp+LOQyAa
```

B2B API

NB:
As of January 2019, the B2B API has been disabled on the Daraja API. Thus this section is here for historical purposes only. The test API requests will most probably not work if tried out. The main way left to perform B2B transfers is now via the M-Pesa Org (https://org.ke.m-pesa.com/) portal or via the API at proxyapi.co.ke (https://proxyapi.co.ke)

The Business to Business (B2B) API enables a Business or Organization to perform transactions between each other. The transaction flow is the same as the B2C API transaction flow, but this time the Credit Party is another Business/Company/Organization. It requires the same credentials and information as the B2C API.

Currently the B2B API allows an organization to perform 5 types of transfers:

Business Pay Bill: This is a transfer of funds from one Organization's Working Account to another Organization's Utility Account.

Business Buy Goods: A transfer of funds from one Organization's Working Account to another Organization's Merchant Account.

Disburse Funds To Business: A transfer of funds from one Organization's Utility Account to another Organization's Working Account.

Business To Business Transfer: A transfer of funds from one Organization's Working Account to another Organization's Working Account.

Merchant To Merchant Transfer: A transfer of funds from one Organization's Merchant Account to another Organization's Merchant Account.

For any two shortcodes to perform B2B transactions between themselves, they both need to have the B2B product assigned to them, otherwise the transaction request will fail. The B2B request structure is as shown below:

```
// URL
[POST] https://sandbox.safaricom.co.ke/mpesa/b2b/v1/paymentrequest

// HEADERS
Host: sandbox.safaricom.co.ke
Authorization: Bearer [access token]
Content-Type: application/json

// BODY
{
    "Initiator": "apitest361",
    "SecurityCredential": "[encrypted password]",
    "CommandID": "[CommandID]",
    "SenderIdentifierType": "4",
    "RecieverIdentifierType": "4",
    "Amount": "100000",
    "PartyA": "601426",
    "PartyB": "254708374149",
    "AccountReference": "",
    "Remarks": "",
    "QueueTimeOutURL": "https://peternjeru.co.ke/safdaraja/api/callback.php" ,
    "ResultURL": "https://peternjeru.co.ke/safdaraja/api/callback.php"
}
```

These are explained below:

Initiator
The username of the API operator as assigned on the M-Pesa Org Portal.

SecurityCredential
The password of the API operator **encrypted** using the public key certificate provided.

CommandID
This specifies the type of transaction being performed. There are five allowed values on the API: **BusinessPayBill**, **BusinessBuyGoods**, **DisburseFundsToBusiness**, **BusinessToBusinessTransfer** or **MerchantToMerchantTransfer**.

SenderIdentifierType, RecieverIdentifierType
This is the type of Identity performing the transaction. An organization's Identity type for a shortcode (identity being used in the request) is 4, thus for both the above parameters, the value will always be 4 for B2B transfers (NB: you cannot use Till numbers in B2B transfers).

PartyA
This is the identifier of the Debit party of the transaction, in this case the debit party being the Organization and the identifier being the Shortcode of the organization.

PartyB
This is the identifier of the Credit party of the transaction, here the credit party being another organization and the identifier being the other org's shortcode.

AccountReference
This is a custom value that may represent an account or unique value item being paid for. It is only required for the BusinessPayBill Command ID.

Remarks
A very short description of the transaction from your end, or just a minimum of 2 characters.

ResultURL

This is the callback URL where the results of the transaction will be sent. Please visit the API Apps section to understand how this is used if you are not familiar with it.

QueueTimeOutURL
This is the callback URL used to send an error callback when the transaction was not able to be processed by M-Pesa within a stipulated time period.

Once sent, you shall expect a success acknowledgement response from the API informing you that your request was accepted. The response format is as below:

```
{
    "ConversationID": "AG_20180326_00005ca7f7c21d608166",
    "OriginatorConversationID": "12363-1328499-6",
    "ResponseCode": "0",
    "ResponseDescription": "Accept the service request successfully."
}
```

From here, process is the same as the B2C flow. A sample success B2B callback is shown below:

```
{
    "Result":
    {
        "ResultType":0,
        "ResultCode":0,
        "ResultDesc":"The service request has been accepted successfully.",
        "OriginatorConversationID":"8551-61996-3",
        "ConversationID":"AG_20170727_00006baee344f4ce0796",
        "TransactionID":"LGR519G2QV",
        "ResultParameters":
        {
            "ResultParameter":
            [
                {
                    "Key":"InitiatorAccountCurrentBalance",
                    "Value":"{ Amount={BasicAmount=46713.00, MinimumAmount=4671300, CurrencyCode=KES
                    }}"
                },
                {
                    "Key":"DebitAccountCurrentBalance",
                    "Value":"{Amount={BasicAmount=46713.00, MinimumAmount=4671300, CurrencyCode=KES}}
                },
                {
                    "Key":"Amount",
                    "Value":10
                },
                {
                    "Key":"DebitPartyAffectedAccountBalance",
                    "Value":"Working Account|KES|46713.00|46713.00|0.00|0.00"
                },
                {
                    "Key":"TransCompletedTime",
                    "Value":20170727102524
                },
                {
                    "Key":"DebitPartyCharges",
                    "Value":"Business Pay Bill Charge|KES|77.00"
                },
                {
                    "Key":"ReceiverPartyPublicName",
                    "Value":"603094 - Safaricom3117"
                },
                {
                    "Key":"Currency",
                    "Value":"KES"
                }
            ]
        },
        "ReferenceData":
        {
            "ReferenceItem":
            [
                {
                    "Key":"BillReferenceNumber",
                    "Value":"aaa"
                },
                {
                    "Key":"QueueTimeoutURL",
                    "Value":"https://internalsandbox.safaricom.co.ke/mpesa/b2bresults/v1/submit"
                }
            ]
            }
        }
    }
```

Some new elements descriptions:

ReceiverPartyPublicName
The Shortcode + Registered Name of the organization being credited in the transaction.

DebitPartyCharges
How much the debit party is being charged for the transfer.

For an unsuccessful transaction, the response format will be the same as the B2C callback, and follows the same rules.

That is pretty much B2B in a nutshell.

B2B API

Try It | Encrypt Test Password | Refresh Token

Access Token

Bearer [access token]

Initiator Name (API Operator's username)

apitest361

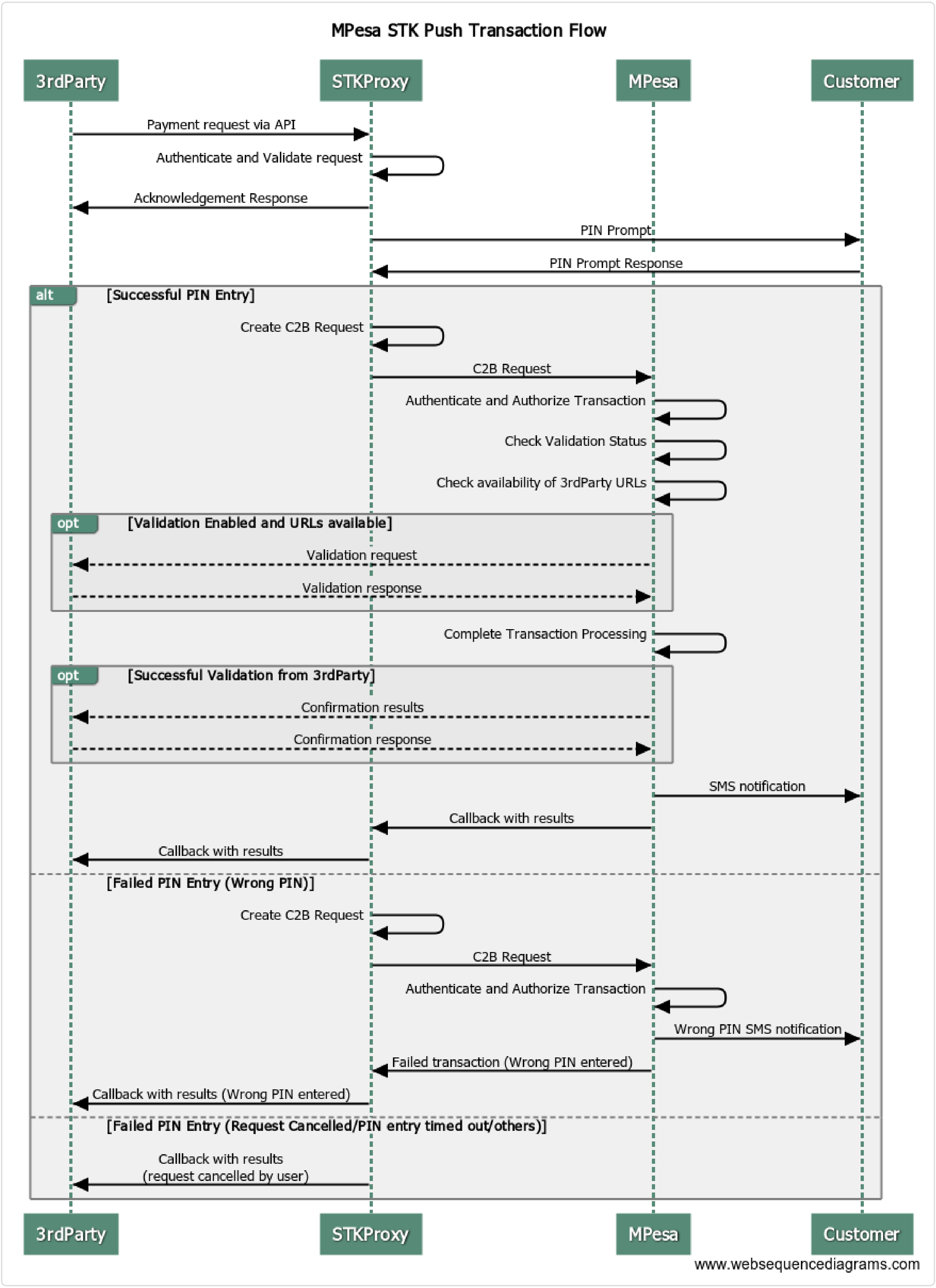
Security Credential (Initiator's Encrypted Password)

```
kEdj/NXOf3+igmuYXYmJUGK86bq7fmTEx0ubGtp7+8i
mSrhInlFrgOTz/36aiZrft/DJgFjh2lk/7/sSzuTNUUc1XaV
Aj0zA6HTQKsCPVJD2TMcxF0t9DbIdhfQ6wRlQpr2JIU
bydIx79NqGFYv2aG5ImhxS5Jv22kdRksVGMjot/coWJ
LnIM0PB6L29OgVg/HPiueEgU4GFLTaXQ3mDAOzfn8
```

Command ID

Lipa Na M-Pesa Online API

The Lipa na M-Pesa (LNM) API is a C2B API designed to utilize the new feature introduced by Safaricom known as STK Push. This feature allows the transaction initiation to be moved from the paying customer's side to the payee Organization's side. This eliminates the hustle of having to remember business paybill numbers and account numbers for customers, and allows them to simply confirm the current transaction by entering their M-Pesa PIN on their mobile phone. This is done via the STK Push prompt which appears on a customer's phone that asks them to enter their PIN. For businesses, this API enables them to preset all the correct info in the payment request and greatly reduce chances of wrong payments being sent to their systems. It is a C2B transaction, but with the initiator being the organization instead of the customer. Here, the organization has the option of presetting all required variables in the request before sending the request, thus this API has no Validation-Confirmation process of *it's own* unlike the previous C2B API (but is still affected by any previous C2B integrations done on the shortcode being used in the request, especially Validation/Confirmation). It's process is explained below together with a flow chart:


MPesa STK Push Transaction Flow
3rdParty
STKProxy
MPesa
Customer
Payment request via API
Authenticate and Validate request
Acknowledgement Response
PIN Prompt
PIN Prompt Response
alt
[Successful PIN Entry]
Create C2B Request
C2B Request
Authenticate and Authorize Transaction
Check Validation Status
Check availability of 3rdParty URLs
opt
[Validation Enabled and URLs available]
Validation request
Validation response
Complete Transaction Processing
opt
[Successful Validation from 3rdParty]
Confirmation results
Confirmation response
SMS notification
Callback with results
Callback with results
[Failed PIN Entry (Wrong PIN)]
Create C2B Request
C2B Request
Authenticate and Authorize Transaction
Wrong PIN SMS notification
Failed transaction (Wrong PIN entered)
Callback with results (Wrong PIN entered)
[Failed PIN Entry (Request Cancelled/PIN entry timed out/others)]
Callback with results
(request cancelled by user)
3rdParty
STKProxy
MPesa
Customer
www.websequencediagrams.com

STK Push transaction flow

1. The Business sets the data in the request and sends it
2. The API receives the request and validates it internally first, then sends you an acknowledgement response.
3. The API then sends an STK Push request to the target customer's mobile phone. The customer's phone has to be online and unlocked to receive the request.
4. The customer confirms the payment amount via the message displayed on-screen, then either enters the PIN or cancels the request accordingly.
5. The API receives the customer's response. If the response is a negative, it cancels the transaction and sends a corresponding callback to the initiating 3rd party via the predefined callback URL in the initial request, with the info on why the transaction was cancelled. The possible negative responses could be due to the following scenarios:
    - An invalid PIN entered by the customer
    - Timeout due to customer not entering the PIN within a given time period (usually 1 min 30 secs)
    - The customer's SIM card not having the STK applet on it
    - A literal request cancellation by the user on their phone
    - Another STK transaction is already underway on the customer's phone (no more than one request can be processed at the same time on the same phone)
6. If the PIN is correct, it means the customer accepted the request. The API forwards the transaction to M-Pesa.
7. M-Pesa automatically processes the request, then sends the response back to the API system which then forwards it to you via the callback URL specified in your initial request. Here, the callback can also either be a success or failure, just like a normal C2B transaction.
8. There are no repeat calls for failed callbacks, thus if the API is unable to send the callback to you, you have the Transaction Status Query API to confirm the status of the request, or also confirm via the M-Pesa Org. portal.

For this API, you shall need your actual production line to test the API. Your line must have the STK applet installed (you can update by dialing ***234*1*6#** on your handset (as of the day of writing of this tutorial. If the update does not work, or your physical SIM card is more than 3 years old, consider replacing it), and you must be registered on M-Pesa. The funds utilized are automatically refunded at midnight. I suggest using values as low as 1 - 10 shillings to perform your transactions to lengthen the number of tries you make. Also beware, the API does not allow one to perform more than 5 consecutive STK requests without completing them. This is marked as phishing, and will cause your line to be blocked from making any more STK requests over 24 hours.

The LNM request takes the format below:

```
// URL
[POST] https://sandbox.safaricom.co.ke/mpesa/stkpush/v1/processrequest

// HEADERS
Host: sandbox.safaricom.co.ke
Authorization: Bearer [access_token]
Content-Type: application/json

// BODY
{
    "BusinessShortCode": "174379",
    "Password": "MTc0Mzc5YmZiMjc5ZjlhYTliZGJjZjE1OGU5N2RkNzFhNDY3Y2QyZTBjODkzMDU5YjEwZjc4ZTZiNzJhZGEx
    "Timestamp": "20201015091344",
    "TransactionType": "[Transaction Type]",
    "Amount": "1000",
    "PartyA": "254708374149",
    "PartyB": "174379",
    "PhoneNumber": "254708374149",
    "CallBackURL": "https://peternjeru.co.ke/safdaraja/api/callback.php"
    "AccountReference": "account",
    "TransactionDesc": "test" ,
}
```

Some definitions

BusinessShortCode

This is the shortcode of the organization initiating the request and expecting the payment.

Password
This is the Base64-encoded value of the concatenation of the **Shortcode + LNM Passkey + Timestamp**, e.g. given the test values above, and using a timestamp of 20201015091344, the encoded password will be

```
MTc0Mzc5YmZiMjc5ZjlhYTliZGJjZjE1OGU5N2RkNzFhNDY3Y2QyZTBjODkzMDU5YjEwZjc4ZTZiNzJhZGExZWQyYzkxOTIwMjAxM
```

You can use this site (http://www.timestampgenerator.com/tools/base64-decode/) to confirm your encoding.

Timestamp
This is the same Timestamp used in the encoding above, in the format **YYYMMDDHHmmss**.

TransactionType
The type of transaction being performed. These are the same values as the C2B command IDs (CustomerPayBillOnline and CustomerBuyGoodsOnline) and the same rules apply here. For now, only **CustomerPayBillOnline** is supported.

Amount
Self explanatory.

PartyA
The Debit party of the transaction/the party **paying out** in the transaction, hereby the phone number of the customer.

PartyB
The credit party of the transaction/the party **being paid** in the transaction, hereby being the shortcode of the organization. This is the same value as the Business Shortcode

PhoneNumber
Same as PartyA.

CallBackURL
This is the endpoint where you want the results of the transaction delivered. Same rules for Register URL API callbacks apply

AccountReference
This is the value the customer would have put as the account number on their phone if they had performed the transaction via phone.

TransactionDesc
Short description of the transaction. Optional, but element must be present.

After sending a successful transaction, you can expect a response in the below format:

```
{
    "MerchantRequestID": "25353-1377561-4",
    "CheckoutRequestID": "ws_CO_26032018185226297",
    "ResponseCode": "0",
    "ResponseDescription": "Success. Request accepted for processing",
    "CustomerMessage": "Success. Request accepted for processing"
}
```

Note the **ResponseCode**. The value 0 (zero) means the request was accepted successfully. Any other value means there was an error validating your request. Confirm the error on the **ResponseDescription** and fix it. The CheckoutRequestID is your unique request ID and can be used later for the LNM Transaction Query API.

After sending the callback, and assuming a customer has accepted your request and responded to it, a successful callback will have the structure below:

```
{
    "Body":
    {
        "stkCallback":
        {
            "MerchantRequestID": "21605-295434-4",
            "CheckoutRequestID": "ws_CO_04112017184930742",
            "ResultCode": 0,
            "ResultDesc": "The service request is processed successfully.",
            "CallbackMetadata":
            {
                "Item":
                [
                    {
                        "Name": "Amount",
                        "Value": 1
                    },
                    {
                        "Name": "MpesaReceiptNumber",
                        "Value": "LK451H35OP"
                    },
                    {
                        "Name": "Balance"
                    },
                    {
                        "Name": "TransactionDate",
                        "Value": 20171104184944
                    },
                    {
                        "Name": "PhoneNumber",
                        "Value": 254727894083
                    }
                ]
            }
        }
    }
}
```

I believe all elements there are already known. I will shift your attention to the **ResultCode**,which shows the status of the request. The 0 (zero) means a success as usual, anything else will be an error whose description is defined in **ResultDesc**. The M-Pesa Receipt Number is your unique identifier of the transaction on M-Pesa, and can be used with the Sandbox Transaction Query API.

LNM API

Try It   Encode Test Passkey   Refresh LNM Token

Access Token

Bearer [access token]

Business ShortCode, Party B

174379

Encoded Password

MTc0Mzc5YmZiMjc5ZjlhYTliZGJjZjE1OGU5N2RkNzFhNDY3Y2QyZTBjODkzMDU5YjEwZjc4ZTZiNzJhZGExZWQyYzkxOTIwMjAxMDE1MDkxMzQ0

Timestamp

Reversal API

According to the documentation, this API enables one to reverse a transaction done. But there are some limitations to this I believe you need to know:

- You will probably only be able to reverse a transaction where you are the credit party. When you are the debit party, you may not be able to initiate a reversal via API. This means it will be done via the Web portal, and may require manual authorization from the SP side. But if you are allowed to reverse a transaction via API, it may also need to be authorized from the SP side. This means you will only get the callback after the SP has reviewed and completed processing the transaction on their side, which can be a couple of hours. Otherwise you are good
- Obviously the reversal is dependent on the funds being available in the originally credited account. If there are no funds in the originally credited account, the reversal fails.
- Charges accrued during the transaction will most probably **not** be reversed.
- You cannot reverse a reversal transaction.
- Any C2B transaction reversal that will cause the customer's account to exceed the maximum allowed account limit (100K) or Daily transaction limits will be declined. If possible notify the customer beforehand.
- Not a limitation but a requirement: an initiator needs the **Org Reversals Initiator** role to be able to perform reversals via API.

The reversal request format is as below:

```
// URL
[POST] https://sandbox.safaricom.co.ke/mpesa/reversal/v1/request

// HEADERS
Host: sandbox.safaricom.co.ke
Authorization: Bearer [access token]
Content-Type: application/json

// BODY
{
    "Initiator":"apitest361",
    "SecurityCredential":"[encrypted password]",
    "CommandID":"TransactionReversal",
    "TransactionID":"[original trans_id]",
    "Amount":"[trans amount]",
    "ReceiverParty":"601426",
    "RecieverIdentifierType":"4",
    "ResultURL":"https://peternjeru.co.ke/safdaraja/api/callback.php",
    "QueueTimeOutURL":"https://peternjeru.co.ke/safdaraja/api/callback.php",
    "Remarks":"please",
    "Occasion":"work"
}
```

Important parameters:

TransactionID
This is the M-Pesa Transaction ID of the transaction which you wish to reverse.

Amount
The amount transacted in that transaction to be reversed, down to the cent.

ReceiverParty
Your Org's shortcode here.

A successful callback will be as shown below:

```
{
    "Result":
    {
        "ResultType":0,
        "ResultCode":0,
        "ResultDesc":"The service request has been accepted successfully.",
        "OriginatorConversationID":"10819-695089-1",
        "ConversationID":"AG_20170727_00004efadacd98a01d15",
        "TransactionID":"LGR019G3J2",
        "ReferenceData":
        {
            "ReferenceItem":
            {
                "Key":"QueueTimeoutURL",
                "Value":"https://internalsandbox.safaricom.co.ke/mpesa/reversalresults/v1/submit"
            }
        }
    }
}
```

Note the TransactionID. This is the transaction ID of the reversal request itself, not the original transaction which was being reversed. The reversal request itself gets its own transaction ID.

Reversal API

Try It | Encrypt Test Password | Refresh Token

Access Token

Bearer [access token]

Initiator Name (API Operator's username)

apitest361

Security Credential (Initiator's Encrypted Password)

```
OVHNJxw8z0LzvxrHOsmydcaT3ImUxM3rAIyXJcGXD
BNTlnuKvLVc4UIw3WC2YHys0/jUsXBu7eNZjxt+0maF
FNQ4IRk/IPAiKtP0NiaP4UIBSCoemdkXvFEIwB3Jy+yr
+1eDbousu6eqVsdyFoid1pvtYkkEtJoCd0Lff0/e5AF5xJI
3vVSIw+0Njg+bEIarM1F+RUraAAd3vVD2zc6K7WZ4z
```

Transaction ID

Go Live

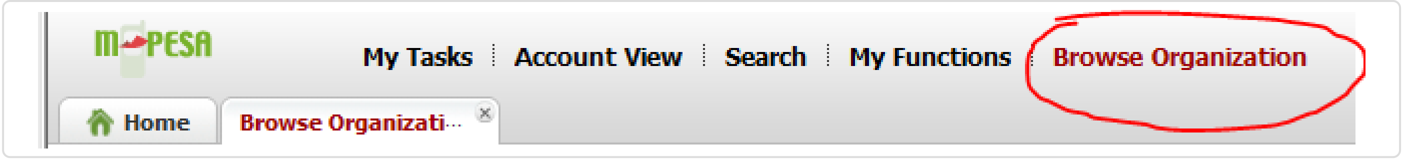
This is one process which has had several people scratching their heads on the portal. It is the process which one uses to enable their application to carry out requests on the live API and begin processing client payments. For this process, you require at least the following two items already existing:

Paybill Number
This is the shortcode which you received after you registered to use M-Pesa services.

Business Administrator/Business Manager
These are admin users on your M-Pesa Organization portal who have the roles of either the Business Administrator or the Business Manager assigned to them. To confirm who this is, you can try to follow the steps below on the M-Pesa Org Portal:

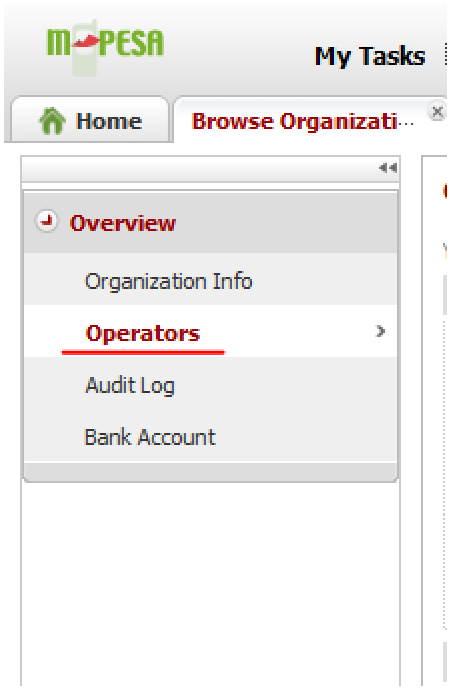
- Log into the Org portal (https://org.ke.m-pesa.com/login.action) using the shortcode which you shall use in the Go Live process using any available web operator. Note that you must have a client certificate installed on your machine for this to work, otherwise the connection is simply terminated due to the browser not being able to get a certificate to verify you with. The Org. Portal Certificate currently works on Chrome, Mozilla and IE. To apply for a certificate, send a blank email to m-pesacertpassword@safaricom.co.ke (mailto:m-pesacertpassword@safaricom.co.ke), you shall receive a temporary username, password and some more instructions after a while. The summary of the instructions is: go to vmtke.ca.vodafone.com/certsrv/ (https://vmtke.ca.vodafone.com/certsrv/) using IE on Windows (unfortunately this site does not accept any other browser or anything else, its based on VB :-( ), enter the username and password provided via email, enter your details then submit. After that, join the cat and mouse game and wait for it to be processed :)

- Check if you have the **Browse Organization** menu at the top of the screen. If not, you are not an admin, sorry. If you have it, click on it.



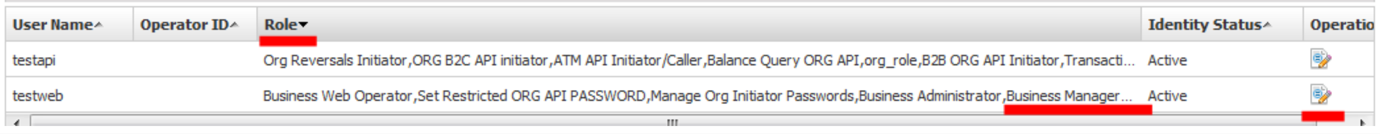
Browse Org Menu Item

- Click on the **Operators** sub-menu on the bar that appears on the left of the screen



Operators Menu Item

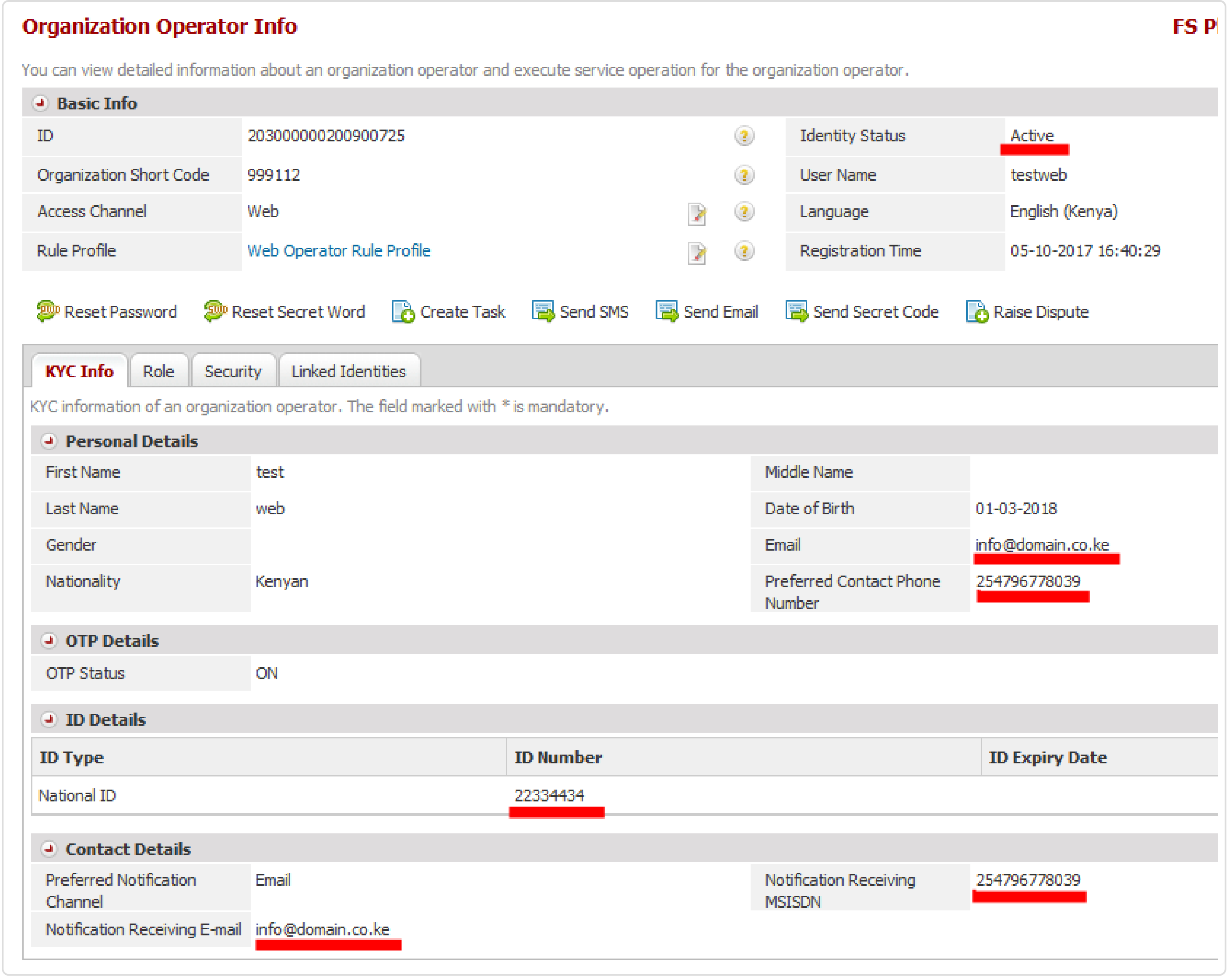
- A list will appear showing all the users on your portal, and their roles. From this list, check on the **Role** column and see if anyone has the **Business Administrator/Manager** role on them. You can also click on the Operation icon at their far right on the list to see their profile and the complete list of roles assignable to them. That user with the Business Administrator/Manager role is the required one for the Go Live process.



User Name	Operator ID	Role	Identity Status	Operatio
testapi		Org Reversals Initiator,ORG B2C API initiator,ATM API Initiator/Caller,Balance Query ORG API,org_role,B2B ORG API Initiator,Transacti...	Active	
testweb		Business Web Operator,Set Restricted ORG API PASSWORD,Manage Org Initiator Passwords,Business Administrator,Business Manager...	Active	

Roles List

- Open the Business Administrator/Manager user's details by clicking the Operation icon at the far right. Once there, click on the KYC tab and confirm that the following details are there:
    - Identity Status is **Active**
    - Email
    - Preferred Contact Phone Number (starting with 2547XX)
    - Notification Receiving MSISDN (starting with 2547XX)
    - Notification Receiving E-Mail
    - ID type and Number



Organization Operator Info FS P

You can view detailed information about an organization operator and execute service operation for the organization operator.

Basic Info

ID	203000000200900725	Identity Status	Active
Organization Short Code	999112	User Name	testweb
Access Channel	Web	Language	English (Kenya)
Rule Profile	Web Operator Rule Profile	Registration Time	05-10-2017 16:40:29

KYC information of an organization operator. The field marked with * is mandatory.

Personal Details

First Name	test	Middle Name	
Last Name	web	Date of Birth	01-03-2018
Gender		Email	info@domain.co.ke
Nationality	Kenyan	Preferred Contact Phone Number	254796778039

OTP Details

OTP Status	ON

ID Details

ID Type	ID Number	ID Expiry Date
National ID	22334434	

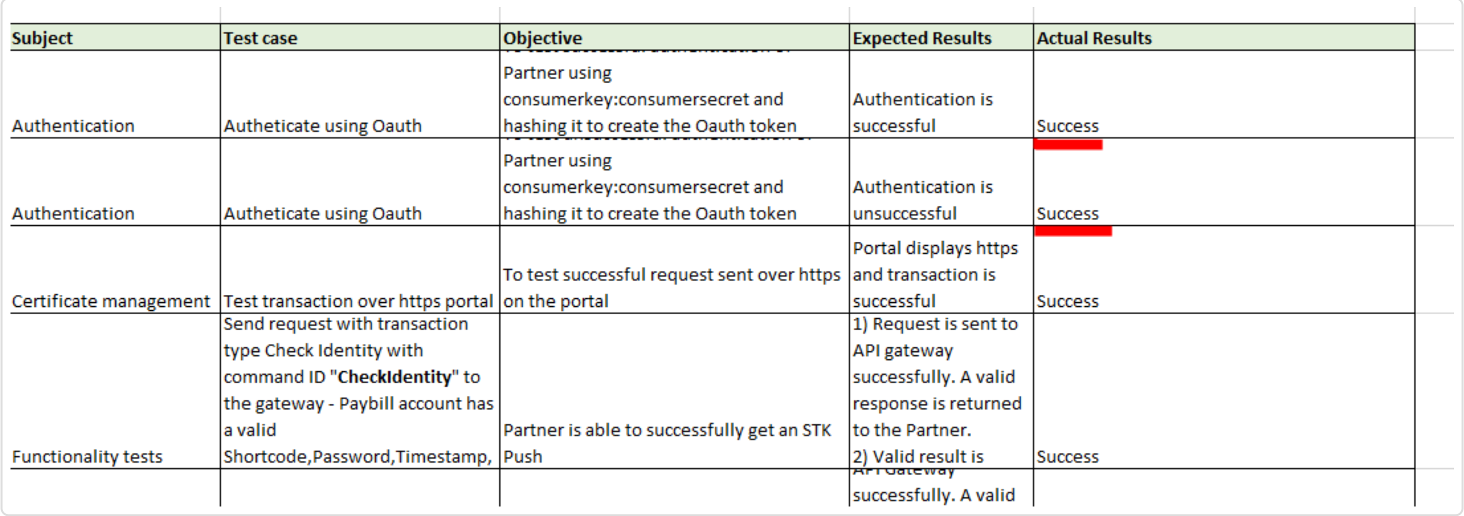
Contact Details

Preferred Notification Channel	Email	Notification Receiving MSISDN	254796778039
Notification Receiving E-mail	info@domain.co.ke		

Operator Details

These details will be used for verification during the Go Live journey.

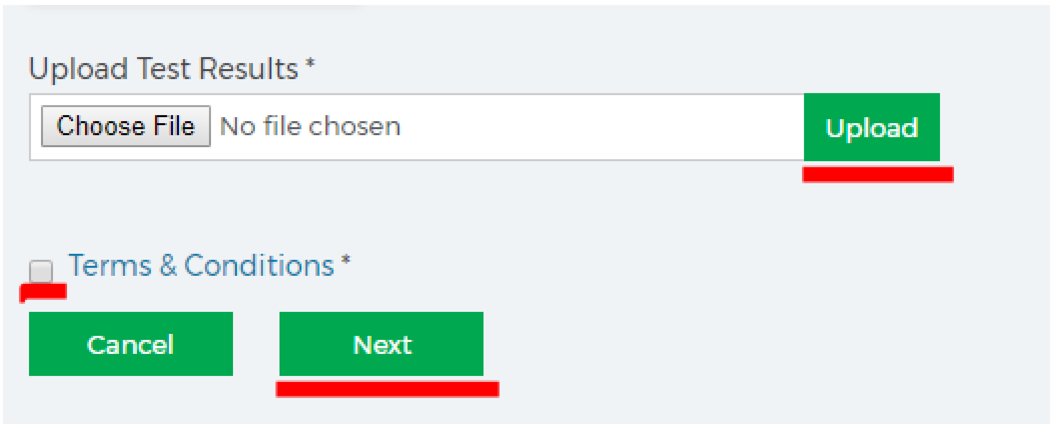
If those details are not available or are not updated, click on the Edit icon on the right side of the panel to add/modify them, then save. After confirming the above details, you may start the process. On the first step (https://developer.safaricom.co.ke/production_profile/form_production_profile) of the Go Live process, you shall be required to download and fill in a form specifying how your tests went along. These cases consist of the possible scenarios which you shall encounter as you test your application. The test cases just need a simple **Success** or **Fail** response on the Actual Results tab. I doubt you will have a Fail on that file anyways :)



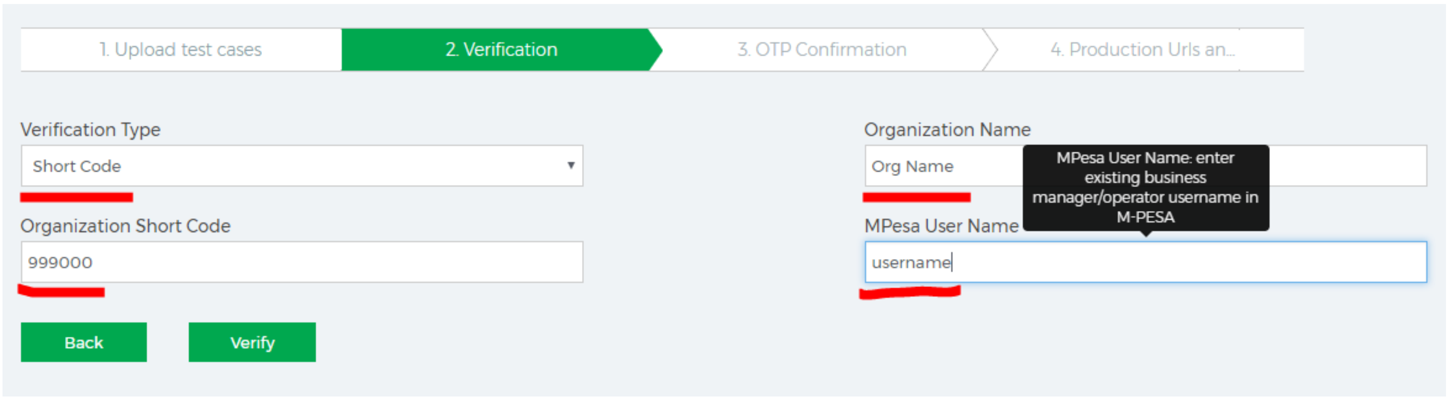
Subject	Test case	Objective	Expected Results	Actual Results
Authentication	Autheticate using Oauth	Partner using consumerkey:consumersecret and hashing it to create the Oauth token	Authentication is successful	Success
Authentication	Autheticate using Oauth	Partner using consumerkey:consumersecret and hashing it to create the Oauth token	Authentication is unsuccessful	Success
Certificate management	Test transaction over https portal	To test successful request sent over https on the portal	Portal displays https and transaction is successful	Success
Functionality tests	Send request with transaction type Check Identity with command ID "**CheckIdentity**" to the gateway - Paybill account has a valid Shortcode,Password,Timestamp,	Partner is able to successfully get an STK Push	1) Request is sent to API gateway successfully. A valid response is returned to the Partner. 2) Valid result is	Success
			successfully. A valid	

How to fill Test Cases form

Once downloaded and filled, go back to the same Page and upload the form via the **Upload Test Results** form, click on **Upload**, then after the form has been picked up, check the **Terms and Conditions** checkbox, then click on Next.



On the **Verification** section, you shall need to confirm your ownership of the paybill you are taking live. This will require you to enter your paybill and your contact details, receive a One Time Pin (OTP) on your registered Mobile Phone, then put that OTP on the portal to confirm your authenticity. On that page, you shall select Verification type as Short Code (only one supported for now), Organization Name as the name of the Organization as registered on M-Pesa, your Organization ShortCode, and the username of either the Business Administrator or Business Manager as filled in on the Org Portal (that's why I had you confirm the details exist first). Without either of these, the process will fail as there will be no contact phone to send you the OTP or the email to send the production URLs.



Verification Type: Short Code
Organization Short Code: 999000
Organization Name: Org Name
MPesa User Name: username
MPesa User Name: enter existing business manager/operator username in M-PESA

Once you click on Submit, and M-Pesa confirms the details you put as correct, it will send an OTP to the Phone Number registered on the Org Portal under the User whose username was filled on the verification form. Once you get the OTP on your phone (you might need a few retries), you shall fill it on the next section, the **OTP Confirmation** section. From the docs, the OTP has an expiry timeout of 3 minutes, thus you need to be fast, or you can just click on Resend OTP. Once the OTP has been confirmed, you shall then be given the chance to select the APIs you are applying for (note the limitations mentioed in the API Apps section). After completing the section, there shall also be created automatically the production apps for your account. These apps need to be approved internally, then after approval, you shall have the production URLs sent to the email registered against the same user who applied for Verification in step 2 of the process. You will then have the new Consumer Keys and Consumer Secrets for your app ready for use in production. More on this process can be found on the official site (https://developer.safaricom.co.ke/docs#going-live). For password creation, you can follow the steps given in the B2C section above

API Errors

These are some of the errors you will meet on the API. More are being added as they are found. If you meet an error not in the list below, feel free to drop it in the comments, will be glad to add it to the list.

Error Message	**Description and Possible Causes**	**Solutions**

(STK_CB) DS Timeout	Applies to Lipa Na M-Pesa API. It means that the STK Push Prompt never got to the user. Causes include: • The user not having an updated SIM Card, thus needs an update • The SIM card being too old (3+ years) to have received the STK Update to allow access to this service. • Mobile phone is offline.	• Update SIM card via ***234*1*6#** or Upgrade SIM card • Make sure target SIM card's mobile phone is online.
SMSC ACK Timeout	Also applies to Lipa Na M-Pesa API. It means that the STK Push Prompt got to the customer but the response by the customer was not sent back on time. This is a backend API issue, not a user issue.	• Simply retry again after receiving the callback. Make sure to notify the user that the request failed.
(STK_CB) Request Cancelled By User	Also applies to Lipa Na M-Pesa API. Means that STK Push Prompt was cancelled from user end. Causes are: • STK Prompt timed out waiting for user input (takes between 1-3 minutes depending on phone model). • User literally cancelled the request on their phones.	• Depending on scenario, either inform the user that they did not respond, or just cancel the transaction, then retry again.
Unable To Lock Subscriber, A Transaction Is Already In Process For The Current Subscriber	Means the user already has another STK Prompt currently active on their mobile phones.	Inform the user, then retry after 2-3 minutes (time taken to automatically cancel an STK Prompt)

The Initiator Information Is Invalid	This error and the rest of the "information is invalid" errors usually applies to Reversal, Transaction Status, Account Balance, B2B and B2C APIs. It means there is a mismatch in the data provided by the user in the request. These include: • Incorrect Initiator username for above API requests • Incorrect Initiator password, or password not encrypted, or wrongly encrypted, for above API requests • Incorrect values for Parties A and B, and Identifier Types where used	• Make sure the Initiator username is correct. This can be checked against what you have on your credentials page or the username that was assigned to you from the M-Pesa Portal • For APIs with Initiator Identifier Types parameter, ensure the Initiator value matches the Initiator Type value e.g. for paybills, Initiator Identifier type is 11. For the Sender/Party A and Receiver/Party B Identifier types, Paybills have identifier type 4, Till Numbers have identifier type 2, and MSISDNs/Phone numbers have Identifier type 1. • For operators, make sure the password is encrypted using the correct public key certificate, and that the encryption algorithm used actually produces the correct result. You can also just encrypt the password using the provided portal tools and use it as a static value in your API calls to make work easier. Also make sure the initiator belongs to the shortcode being used, as the initiator has a direct relation to the shortcode used in the transaction. Also, make you do not copy paste additional spaces during password encryption. This causes alot of issues as the resulting password is completely different from the expected one. • In the API calls, the Sender/PartyA represents the Debit party (party being debited the cash), and the Receiver/PartyB represents the credit party (party receiving the cash value). Make sure you use the correct value for each e.g. for B2C, the sender is the Shortcode, the receiver is the MSISDN/Phone number. In B2B requests, both the sender and receiver are shortcode numbers, but **NOT** the same shortcode. Also, please do not confuse B2B requests (movement of funds between different paybills) with Intra-account transfers (movement of funds between accounts within same paybill) • For reversal calls, make sure the Initiator belongs to the shortcode reversing the transaction, and that the transaction being reversed was not debited from the initiator's paybill. You cannot reverse a transaction debited from your own account.
The Receiver Information Is Invalid	See above	See above

Credit Party Customer Type (Unregistered Or Registered Customer) Can't Be Supported By The Service	This applies to C2B, B2C and Lipa na M-Pesa API. It means the number in the request is not recorded with M-Pesa, whether as a registered or unregistered customer	Ensure the phone number being used in the transaction is registered with M-Pesa.
Invalid Access Token Error	Applies to all APIs. Means the access token used in the API calls (usually preceded by the word **Bearer**) is expired or invalid.	On Sandbox, the token usually lasts for 1 hour, so refresh the token again by sending a new Generate Token API call.
Invalid Amount	Means you have entered a weird value as the amount	Make sure the amount makes sense and is actually valid for the transaction e.g. C2B requests cannot have amounts greater than Ksh. 70K in the request, and all APIs cannot have amounts below Ksh. 0. Also, all APIs cannot have amounts going above Ksh. 999,999,999 in value. I also don't think you got such an amount in your account :)
Transaction Failed, M-Pesa Cannot Complete Payment	M-Pesa was not able to complete the transaction in the back end.This is usually due to an M-Pesa rule not being fulfilled in the back end, thus depends on the API being used.	For each API, you need to make sure all prerequisites are fulfilled before making the request. These include: • For B2C requests, the client has to make sure the transaction does not cause their accounts to exceed the maximum allowed limit of 100K. • For C2B, it means the paybill being used may have external validation enabled, and that it failed, thus confirm external validation is disabled, or that it went through and was accepted. • Otherwise, means there was an error on the backend we are not aware of. So try again later.
Paybill Verification Failed Due To The Following Reason: No Paybill Verification Data Found Service Request For Registration...	Occurs during the Go Live process. Means M-Pesa could not find the required data on the paybill to verify the person taking the shortcode to production.	Follow the instructions in the Go Live process above to update your KYC data on the M-Pesa Portal, then try again.

Merchant Not Allowed To Carry Out Transaction Type CustomerPayBillOnline	Occurs in Lipa na M-Pesa API, mostly production. Means the shortcode (paybill or till number) being used by the merchant is not allowed to perform Lipa na M-Pesa API calls using the CustomerPayBillOnline Command ID. Same goes for CustomerBuyGoodsOnline Command ID. It could also mean you are using the wrong command ID for your shortcode.	You need to request to have the Command ID enabled on your shortcode. Send the request to apifeedback@safaricom.co.ke requesting for your shortcode to have the shortcode enabled for Lipa na M-Pesa API. You will then also receive a passkey to use in your requests. Also make sure you are using the correct command ID for your shortcode i.e. CustomerPayBillOnline for Paybill numbers, CustomerBuyGoodsOnline for Till numbers.
Missing ICCID value in request	Occurs on Lipa na M-Pesa API. Means the number you have entered is not a Safaricom registered number, or it's blocked/inactive/dead.	Ensure the number you are using is actually a Safaricom number, it is registered, and is not blocked/inactive e.t.c.
MerchantValidate - Wrong credentials	Occurs on Lipa na M-Pesa API. Means the `Password` parameter you have provided is not valid due to either of the following: • If you are on production, your Shortcode has not been allowed to perform LnM requests. For sandbox, there is only one allowed shortcode for LnM, **174379** as of 2019, so confirm the Shortcode in your request is the allowed one. • The `Password` parameter you have provided in your request is either invalid or even missing. • The `Password` given (after proper encoding it) does not match or does not belong to the `BusinessShortcode` given in the same request. • Either the `BusinessShortCode` or `Timestamp` used in encoding the `Password` does not match the one used in the body of the request.	• If you are or have gone live, ensure you have actually been allowed to perform the LnM API request by the Safaricom team and assigned a password/passkey. If not, contact them to be allowed and get one. • On all environments, ensure the password is correctly encoded, including the BusinessShortCode, Timestamp and its format, and ensure the BusinessShortCode and Timestamp used in the encoding are the same as in the body of the request. This is shown in the LnM section above.

Bad Request - Invalid Timestamp	Means the `Timestamp` parameter you have provided in your request is not valid.	Ensure the `Timestamp` parameter is of the expected format e.g. **YYYMMDDHHmmss** for LnM requests as explained in the LnM section above. Be careful of the specific representation of time in your specific language of implementation.
Bad Request - Invalid Amount	Means the `Amount` parameter you have provided in your request is not valid.	Ensure the `Amount` parameter is numeric (integer or float), not a string. Also ensure its a valid amount.
Merchant not allowed to carry out transaction type CUSTOMERxxxONLINE	Occurs on Lipa na MPesa API. Means the shortcode being used has been allowed on the API, but is not allowed to perform that specific transaction type e.g. a **Paybill** trying to perform `CUSTOMERBUYGOODSONLINE` transaction type yet has been assigned a `CUSTOMERPAYBILLONLINE` type, or a **Till** trying to perform a `CUSTOMERPAYBILLONLINE` transaction yet has been assigned a `CUSTOMERBUYGOODSONLINE` transaction type.	When going live with the LnM API, ensure you communicate to the API team the specific transaction type you need for your shortcode i.e. if you got a paybill, you need a `CUSTOMERPAYBILLONLINE` transaction type, and if you got a Till, you need a `CUSTOMERBUYGOODSONLINE` transaction type. Also ensure you only use the transaction type assigned to you.

Transaction failed, M-PESA cannot complete payment of Ksh 123.45 to John Doe and Co. Organization receiving the payment is unavailable, try again later.	This is the generic SMS received by a paying customer during C2B or Lipa na M-Pesa transactions. When a user pays to an organization Paybill and receives the above message, it means the organization being paid has enabled External Validation on their paybill but was unable to validate the transaction, thus it was cancelled. This can happen for either of the below reasons: • The Validation endpoints of the organization supposed to receive the funds were unreachable, thus MPesa performed the default action specified during URL registration, in this case cancelling the transaction (see the Register URL section above). • The organization received the validation request and decided to cancel the transaction, most probably due to wrong inputs by the paying customer. • The organization received the validation request but took too long to respond, thus MPesa cut short the operation and performed the default action specified during URL registration, in this case cancelling the transaction.	• As the client, ensure you have input the correct values for amount and account number during payment. These are checked by the receiving organization and a response sent to M-Pesa whether to complete the transaction or not. • The organization should check their systems and ensure they are up and running, and reachable. They should also ensure their systems are fast enough to respond to MPesa validation requests within the given time period, usually < 8 seconds
Internal Server Error In Crq Creation, Please Try Again	Happens on the Go Live process. This is an error on the back-end	Nothing you can do about this. Just keep retrying till it works.
System Internal Error	Also an error on the back-end	Also keep retrying till it works. If it persists, escalate to the support team.

Utilities

Some utilities to assist you:

Callback Tester (https://bennito254.com/home/playground.html) for testing your callbacks' reachability. Simply enter your callback URL and check if you are receiving the data on your end (Credit to **Bennito254**).

Telegram group (https://t.me/payments_api) where you can get answers to your API questions from fellow developers and get assistance on the API.

Test Cases Form (export.php) for quickly filling and downloading the **most common** test cases.

Production Password Encryption tool. Password is encrypted on the browser, never saved and never leaves your machine. Check out source code for how it is done in JS:

Plaintext Password

Plain Password

Encrypt

Encrypted Password

Copy to Clipboard

A Blog (https://peternjeru.co.ke/blog/) where you can find more info about M-Pesa, its working and tutorials to other issues most commonly faced by other developers when integrating the API e.g. M-Pesa Portal certificates.

Comments