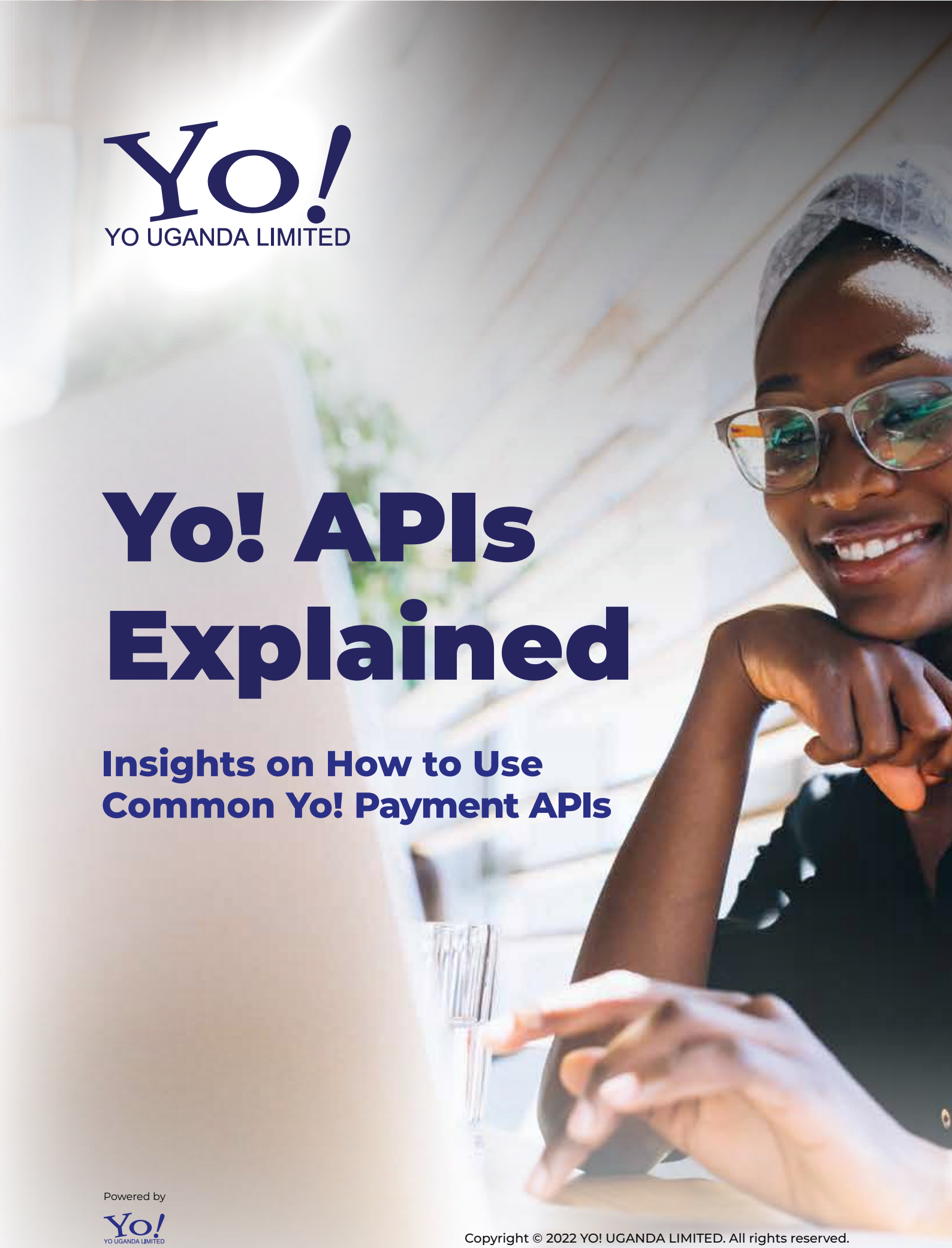
YO UGANDA LIMITED

Yo! APIs Explained

Insights on How to Use Common Yo! Payment APIs



Powered by

Yo! Payments: API Specification

Table of Contents

Document History

Revision Date	Comments	Reviewer
10/06/2022	Initial draft	Joseph T
10/06/2022	Added sections: 1. Withdraw through Bank. 2. Withdraw through Internal Transfer. 3. Deposit funds through Bank.	Joseph T

1 Introduction

Yo! Payments is a revolutionary mobile payments gateway service. Yo! Payments enables businesses to receive payments from their customers via mobile money, as well as make mobile money payments to any mobile money account holder. Yo! Payments also has the capability to send mobile calling credit ("airtime") directly to users. Yo! Payments offers a rich API which enables seamless integration with websites, IVR services, SMS services and any other medium through which businesses interact with their customers. Yo! Payments also offers an "internal transfer" service which enables account holders to cheaply transfer funds amongst each other.

1.1 Document Purpose

This document is written to provide a comprehensive overview of the Yo! Payments APIs so you can easily understand how and when to use them before conducting any integration work in your system. You will need the official Yo! Payments API documentation to further guide your implementation process. This document is written for software developers who intend to integrate Yo! Payments in their system.

1.2 Perquisite

As you prepare to integrate Yo! Payments, you will need the following:

- A sandbox account which you can sign up using the following link: *https://sandbox.yo.co.ug/services/yopaymentsdev/signup/start/?sid=1*
- To verify the signature which comes in the Instant Payment Notification (IPN) callback, you will need the public key which you may obtain from Yo Uganda support team.
- To compute the signature for the withdraw API, you need to generate an asymmetric key pair after which you share the public key with the Yo Uganda support team so they can configure it on your account. The private key should securely be kept on your server so you can use it later to compute the signature. The provided public key at Yo! Payments shall then be used to verify the signature computed using your private key. This strengthens security on your account as Yo! Payments system will only process withdraw requests that have been signed by your system.

1.3 API Format

The Yo! Payments API is written in XML format. The way it works is that your system initiates an HTTP POST request, with the raw data presented in the XML format as defined in the official Yo! Payments API document for a specific API method you would like to invoke at Yo! Payments. Yo! Payments then validates, processes and returns the response in form of an XML content. You then need an XML parser to extract the response fields from the response XML so you can use them further to update your system.

1.4 Support

If you are stuck and you need clarification on something in this document, seek assistance from Yo Support team through the following support channels.

- WhatsApp: +256 788 238665
- Email: support@yo.co.ug
- Skype: yougsupport

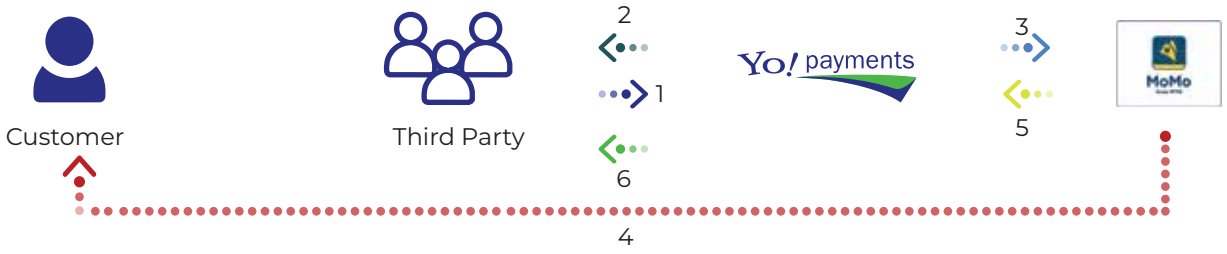
2 Depositing Funds

In the Yo! Payments terms, depositing funds means receiving funds on your Yo! Payments account through Mobile Money payments or Bank transfer. It can be your customers paying you or it could be yourself funding the account to accumulate balances for other purposes such as bulk payments. Some other examples where Yo! Payments' deposit funds APIs would be used includes the following:

- When receiving payments for goods and services.
- Customers making loan repayments.
- Customers paying for subscription services.
- Customers making payments for prepaid services.
- Paying for immediate goods/services, such as funding betting accounts, paying for online premium contents such as music files/videos, and so much more.

2.1 Depositing Funds Asynchronous

Use the Yo! Payments acdepositfunds method with the NonBlocking field set to TRUE so that you get an immediate response after the request has been received by the Yo! Payments gateway. Sending your requests this way ensures that the Yo! Payments server does not hold the request to wait for your customer to approve the payment on their phone. Instead, it acknowledges the transaction request and returns the response immediately. You will then wait for the Instant Payment Notification (IPN) callback for a transaction update or use a transaction check status API to check for the status of the transaction. See the sequence diagram below.



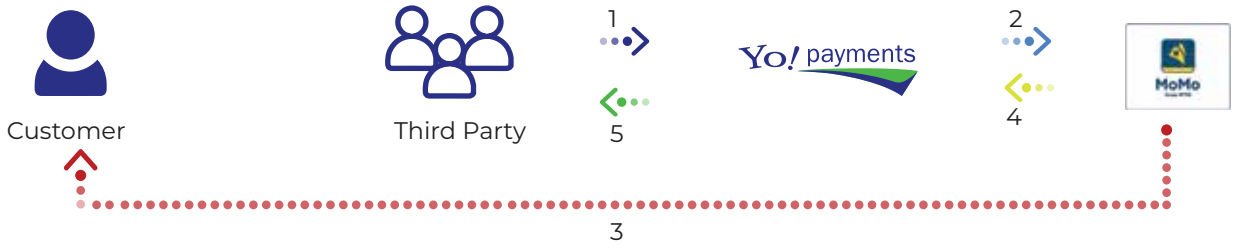
Description

1. 3Party initiates an acdepositfunds with the NonBlocking set to TRUE.
2. Yo! Payments receives the request, authenticates and returns the response to 3party
3. Yo! Payments initiates a payment request against the customer's mobile money number through the Mobile Money wallet provider.
4. The Mobile Money wallet provider registers the transaction and initiates a USSD prompt on the customer's phone to approve this transaction by entering their PIN.
5. Upon successful approval, Yo! Payments gets updated about this transaction from the network.
6. Yo! Payments posts an Instant Payment Notification (IPN) to the 3party's system. The 3party can then provide the service to the customer.

2.2 Depositing Funds Synchronously (Depreciating Soon)

Use the acdepositfunds method with the NonBlocking field set to FALSE (or omit the NonBlocking field) so that your request is held until the customer approves the transaction on their phone. If the customer approves the transaction on their phone in time, you will most likely receive the final transaction status (which can be SUCEEDED or FAILED) under the TransactionStatus field of the XML. On a rare occasion that Yo! Payments fails to determine the final status of the transaction from the mobile money wallet provider, the TransactionStatus field will be set to INDETERMINATE, meaning this transaction will be resolved within an hour. Refer to the Yo! Payments API documentation for this API's XML field description.

Note: We strongly recommend that you use the asynchronous method of running an acdepositfunds transaction as there is a limit on the maximum number of synchronous transactions that you can initiate from your system, which is not the case with the asynchronous method.



Description

1. 3Party initiates an acdepositfunds request with the NonBlocking set to FALSE.
2. Yo! Payments Gateway receives the request, authenticates and forwards the request to the mobile money wallet provider to initiate a payment against the customer's phone number.
3. The Mobile Money wallet provider registers the transaction and initiates a USSD prompt on the customer's phone to approve this transaction by entering their PIN.
4. Upon approval, Yo! Payments gets the updates from the mobile money wallet provider.
5. Yo! Payments returns a response to the 3party with the final status of the transaction.

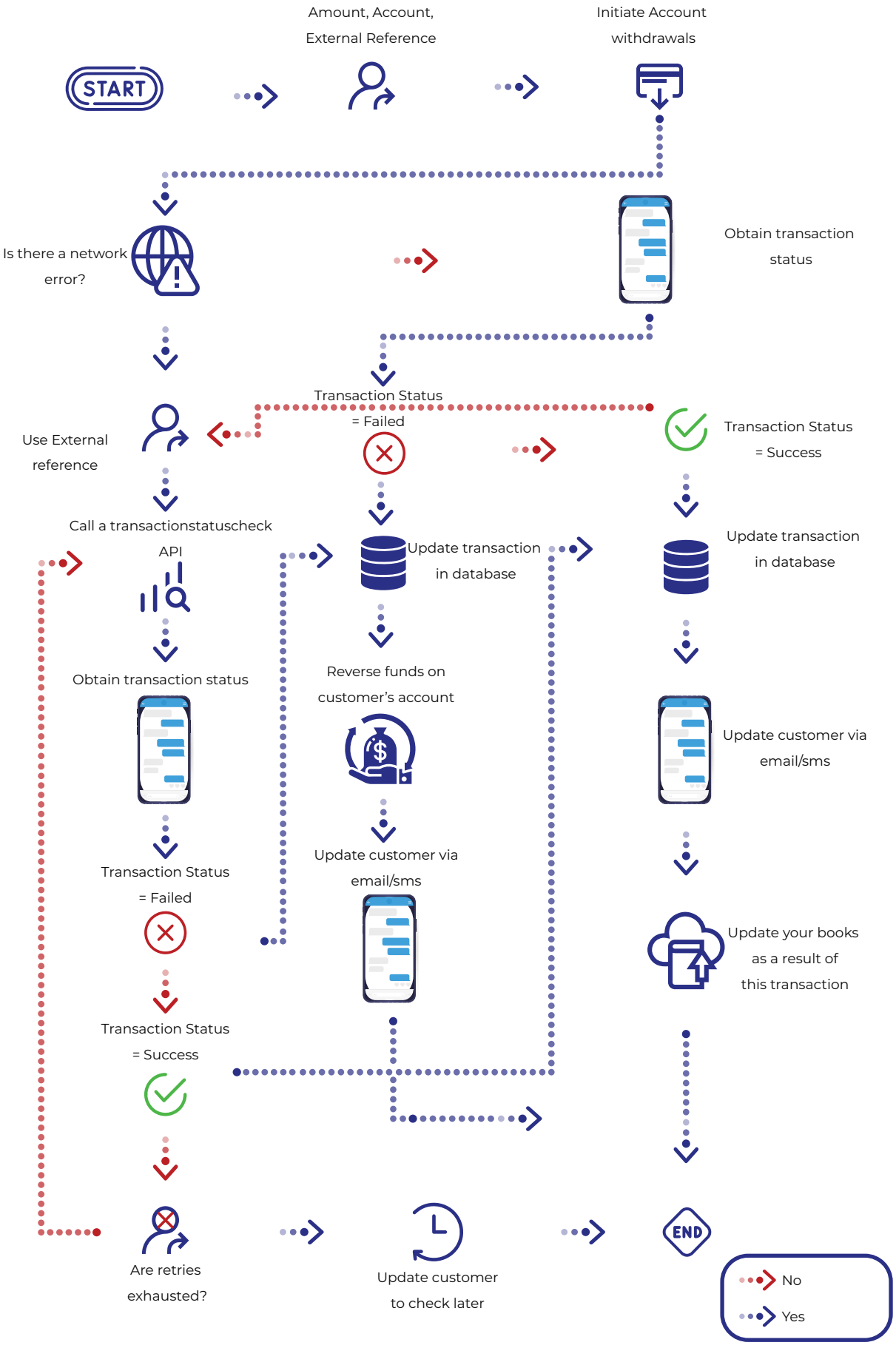
Note: The above occurs within the same request made by the 3party to Yo! Payments. If the network connection between the 3party and Yo! Payments breaks before the customer approves the transaction, it's advisable that your program handles this condition properly by falling back on to using actransactioncheckstatus API method to check for the status of the transaction.

2.3 Initiating Deposit Funds API Synchronously

The proposed pseudocode below should guide you through the process of initiating and handling an acdepositfunds request with the NonBlocking field set to FALSE.

1. Obtain the request details (such as Amount, Account, Narrative, ExternalReference, etc.)
2. Save the transaction in your database tables or any other permanent storage for later access
3. Initiate an acdepositfunds API request to Yo! Payments.
4. If the request returns an expected response, update your database/permanent storage and provide the service to the customer.
5. However, If the request breaks due to the network error, update your database/permanent storage, setting the status to INDETERMINATE (or anything else that can represent a transaction whose status is unknown in your database). This is to help you later check to for the status of this transaction.
6. Within a loop, run an actransactioncheckstatus API to check for the status of the transaction in an interval of 15 seconds for at least 3 times.
   - Within the check status request, check whether the transaction exists with the possible TransactionStatus values set to PENDING, INDETERMINATE, FAILED or SUCCEEDED.
   - If the TransactionStatus is set to PENDING or INDETERMINATE, continue with the loop.
   - If the transaction is set to SUCCEEDED, the transaction was successful, update your permanent storage and provide the service/product to the customer.
   - If the TransactionStatus is set to FAILED, the transaction at Yo! Payments failed. Update your permanent storage and update the customer – informing them that their payment failed.
   - If you receive a response with Status field set to ERROR and StatusCode set to -30 which indicates that the transaction was not found at Yo! Payments, then you know that the initial request didn't even reach Yo! Payments, hence the transaction should be marked with a FAILD status. Update your database and inform the customer that the transaction failed.
   - If the check status request runs in to a network failure error, try again after 15 seconds.
   - If the check status retries are exhausted, inform the customer that you are unable to determine the status of the transaction, the payment will be updated later.

The above pseudocode has been depicted in the flow diagram on the next page.

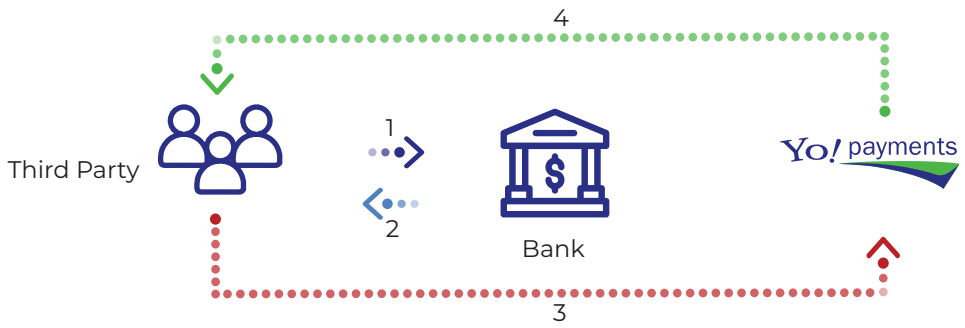

START
Amount, Account, External Reference
Initiate Account withdrawals
Is there a network error?
Obtain transaction status
Transaction Status = Failed
Use External reference
Transaction Status = Success
Call a transactionstatuscheck API
Update transaction in database
Update transaction in database
Obtain transaction status
Reverse funds on customer's account
Update customer via email/sms
Transaction Status = Failed
Update customer via email/sms
Update your books as a result of this transaction
Transaction Status = Success
Are retries exhausted?
Update customer to check later
END
No
Yes

2.4 Deposit Funds via Bank Transfer

Yo! Payments allows you to deposit funds via bank transfer method where you transfer funds to a designated bank account through EFT, RTGS, or even direct deposit over the counter, then you initiate a deposit request via the web portal so the transfer is further processed for the funds to reflect on your Yo! Payments account.

This API is quite useful in a situation where you need to load a large amount of money to facilitate other business activities such as bulk payments. For example, you may run an organization that pays salaries to a large group of workers, (say over 100) each with an amount of 500,000 UGX. This means you need over 50,000,000 UGX on your account to process all the payments, implying that depositing funds via Mobile Money method would be practically difficult as there is a limit of a maximum of 20,000,000 UGX that you can transfer per day. You should instead deposit money using the bank transfer method because the limits here favor the amounts you can possibly transfer to your account per day.

The process of filling an online form on the Yo! Payments portal to create a bank deposit request can be achieved via the bank deposit API which is helpful as far as automating this business process is concerned. For example, you may have a custom accounting software, already integrated with your banker's system to help you initiate an EFT/RTGS off your company's bank account to the Yo Uganda's bank account. Then, instead of asking your accountant to separately log into Yo! Payments portal to physically create a bank deposit request, they only click a button on your custom accounting system which in turn takes the EFT details and submits the bank deposit API request to Yo! Payments.



Description

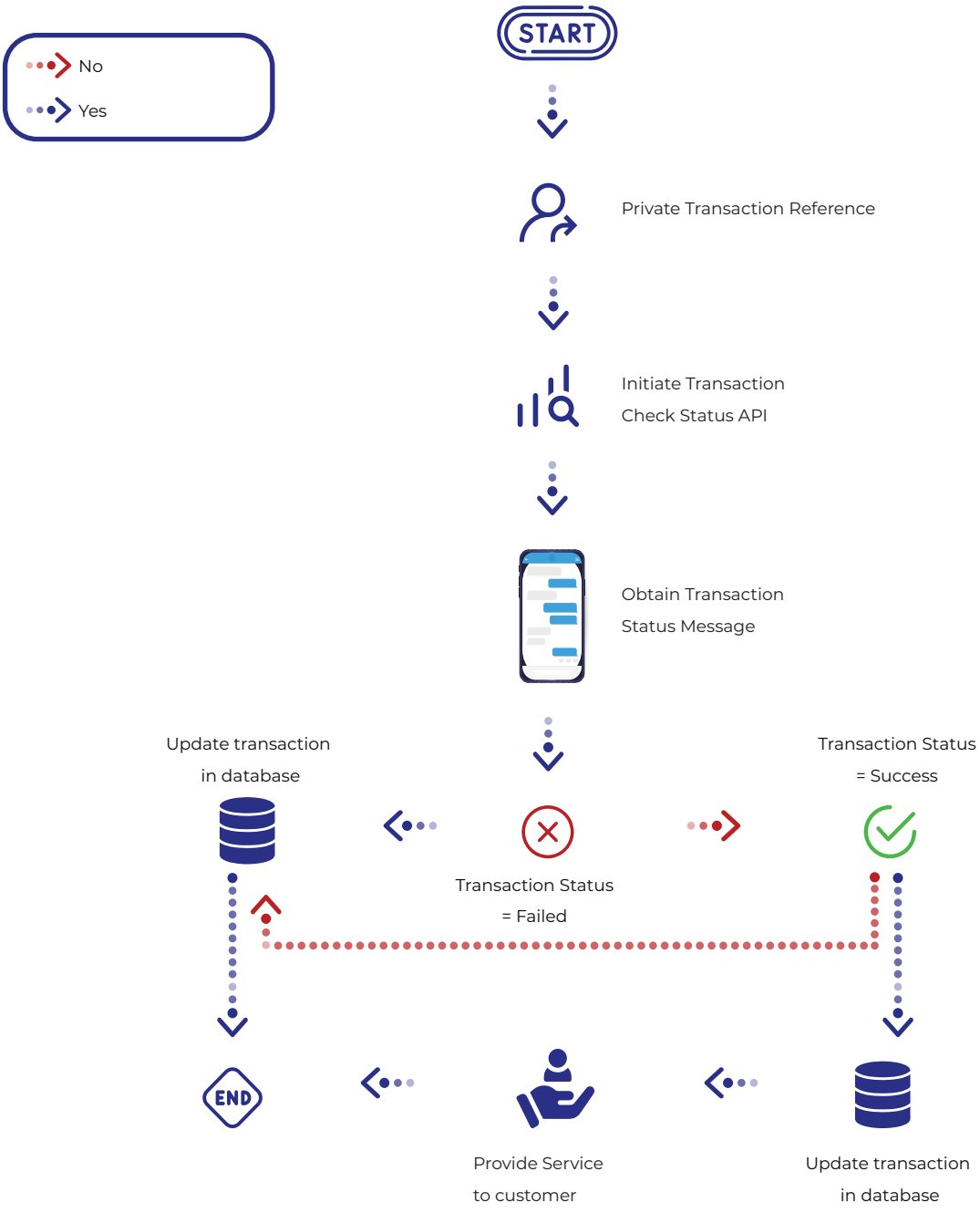
1. 3party transfer the money to Yo Uganda's bank account.
2. The 3party's bank confirms that the transfer has been processed successfully.
3. 3party initiates the bank transfer API
4. Yo! Payments acknowledges the request and internal processes are further started to load the funds on to the 3party's Yo!Payments account. Within this response, you will get the settlement identifier which you can use later to check for the status of the transaction.

Note: Because the bank transfer process might take quite some time to complete, it's advisable that you implement a background service which periodically checks for the status of the above transaction using the accheckbanksettlementstatus, refer to the main API document for the accheckbanksettlementstatus method.

3 Transaction Check Status API

Use the actransactioncheck status API to check for the status of an earlier submitted transaction. This method requires you to have at least your ExternalReference or the unique transaction reference from Yo! Payments. Use the ExternalReference which you submitted in the earlier submitted transaction under the PrivateTransactionReference field in case you never got a proper XML response from Yo! Payments when you initiated your acdepositfunds or acwithdrawfunds API request.

You may have a background service, a cron job or a time scheduled task which is executed periodically to check for the status of the transaction. In this service, you may implement a similar logic like the one depicted in the flow chart below for each transaction in your database whose status is PENDING or INDETERMINATE.



4 Withdraw Funds

Yo! Payments API offers three different ways to withdraw money to a beneficiary's account and these include:

1. Through Mobile Money (from Yo! Payments to the beneficiary's mobile money wallet).
2. Through Bank transfer (from Yo! Payments to beneficiary's bank account).
3. Using Internal transfer (from Yo! Payments to another Yo! Payments account).

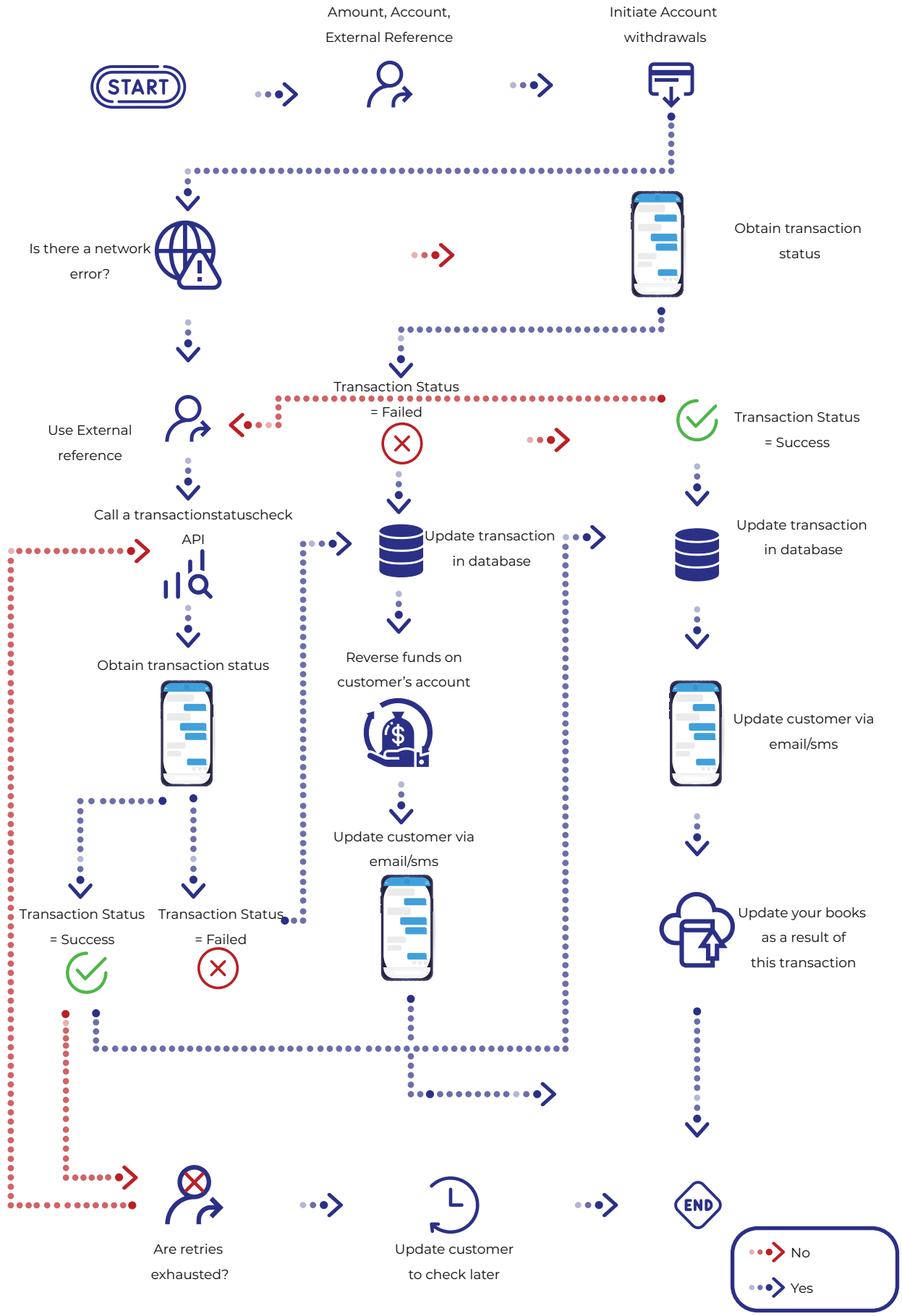
All the above transactions result in a debit on your Yo! Payments account (money leaving your account) and a credit of money at the beneficiary's account with the mobile money wallet provider, bank or Yo! Payments.

4.1 Withdraw Through Mobile Money API

The 3party system can withdraw funds to a beneficiary's Mobile Money number using the available balance on the 3party's Yo! Payments platform. Each mobile money balance used corresponds to a wallet provider (such as MTN Mobile Money, Airtel Money, etc.). You must have enough balance (which should cover the Yo! Payments charges too) in order to run a successful withdraw transaction. To initiate a withdraw transaction via mobile money channel, you should submit an acwithdrawfunds API request, indicating the Amount, Account, Narrative, ExternalReference and other fields as per the API document. Refer to section 4 of the official Yo! Payments documentation for detail on how to use acwithdrawfunds API method.

You will use an acwithdrawfunds API for customers who hold accounts with you on your platform. For example, if you are running an online betting platform, a customer who normally wins their bets would most likely need to cash their money out. You will probably debit their account on your platform and initiate an acwithdrawfunds which credits their Mobile Money wallet at the Mobile Money wallet provider's system. Another example would be a daily wage worker whom you pay at the end of the day after their work. You can use this API method to initiate a withdraw of funds off your Yo! Payments account to this worker's Mobile Money wallet.

Acwithdrawfunds API in most cases runs synchronously unless, if on a rare occasion, some transactions are interrupted at the Mobile Money wallet provider. If you provide a service for which the beneficiary holds some balance on your platform, we strongly encourage you to implement a flow where you first debit this beneficiary's account on your platform before you submit an acwithdrawfunds. Then, after you have confirmed that the transaction at Yo! Payments has been successful, you commit to update the customer. However, If Yo! Payments has confirmed the transaction failed, you reverse the funds so that this beneficiary gets back their money on your platform. The next page illustrates a flow where you initiate an acwithdrawfunds API method to the beneficiary's Mobile Money wallet through Yo! Payments and how you would handle the failure state of the transaction


START
Amount, Account, External Reference
Initiate Account withdrawals
Is there a network error?
Obtain transaction status
Transaction Status = Failed
Transaction Status = Success
Use External reference
Call a transactionstatuscheck API
Update transaction in database
Update transaction in database
Obtain transaction status
Reverse funds on customer's account
Update customer via email/sms
Update customer via email/sms
Transaction Status = Success
Transaction Status = Failed
Update your books as a result of this transaction
Are retries exhausted?
Update customer to check later
END
No
Yes

4.2 Withdraw Through Bank Transfer

You can withdraw your money using the Yo! Payments' bank withdraw API. You will need a verified bank account registered first on your Yo! Payments account, amount, currency, bank name, and the bank account number.

To register a verified bank account, you need to use the accreateverifiedbankaccount API method. Refer to the official Yo! Payments documentation on how to use accreateverifiedbankaccount API method to create a verified bank account.

To initiate a withdraw transaction via bank transfer, use acwithdrawfundstobank API method, refer to the API document for details on how to use this API.

Description

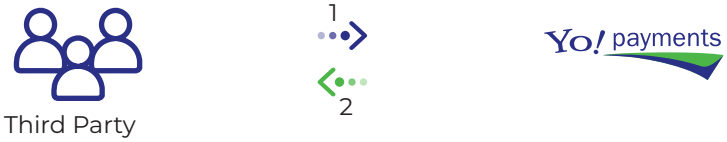
1. 3party initiates a bank deposit API.
2. Yo! Payments returns a response with SettlementRequestIdentifier which you later use to check for the status of the transaction. Refer to the official Yo! Payments API document for details on how to use accheckbanksettlementstatus API method to check for the status of a pending bank transfer request.
3. Yo Payment processes the transfer to the bank.

4.3 Withdraw Through Internal Transfer

You can withdraw money using the Yo! Payments' internal transfer API method. Internal transfer API allows you to transfer funds from your Yo! Payments account to another Yo! Payments account. All you need is the beneficiary's Yo! Payments account number, the beneficiary's email address associated to their Yo! Payments account, amount, narrative, and other required fields, refer to the official Yo! Payments API document for how to use the acinternaltransfer details.

The internal transfer API can be useful especially if you want to minimize the cost involved in sending money through other channels like mobile money and bank transfers. If for example you have members that you pay using mobile money, you could ask them to open up personal Yo! Payments accounts and you initiate payments through the internal transfer API.

There are more advantages of using the Internal transfer API apart from being cheaper and these include reliability as it does not rely on any other system to get executed, faster as it occurs within Yo! Payments and simplicity.



Description

1 3party initiates an internal transfer API request.
2 Yo! Payments validates, processes and returns the response to 3party.