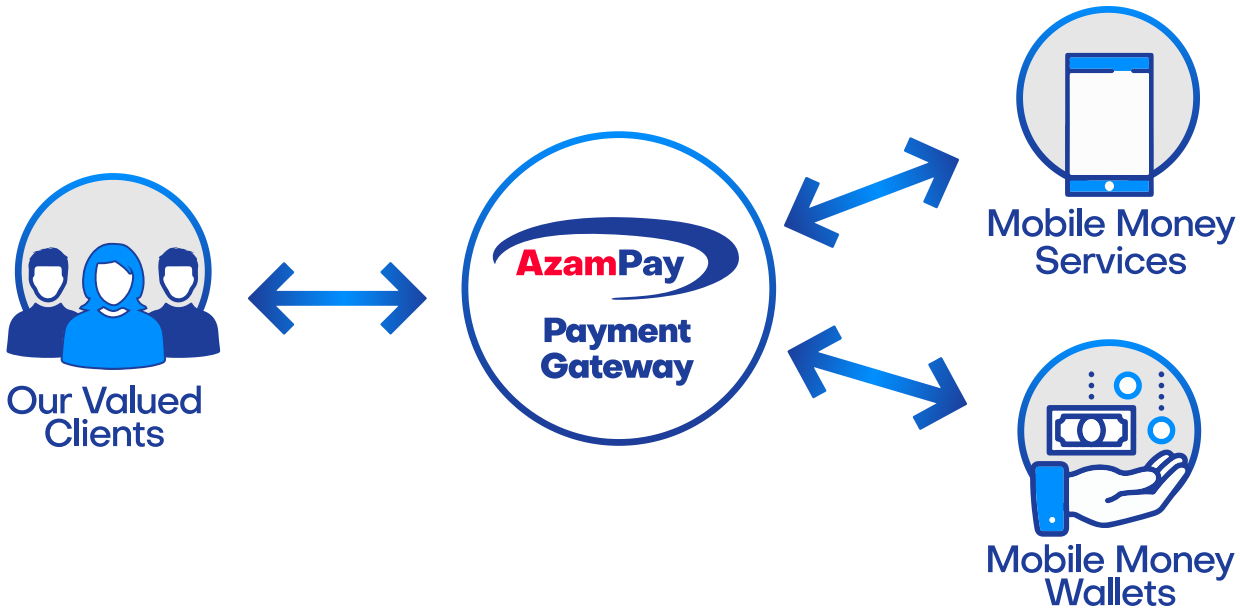
Azampay API (v1)

Download OpenAPI specification: Download

Introduction

AzamPay is specialized in the development of end-to-end online payment management solutions for companies operating in East Africa. Our range of digital solutions and services are carefully designed not only to streamline your payment and collection processes, but to also allow easy integration with your current Accounting or Enterprise Resource Planning (ERP) systems thus leaving you time to focus on your customers. AzamPay offers bespoke solutions that guarantee optimal business performance and efficiency whether you are transacting locally, regionally, or internationally.

We strive to consistently improve our products to better meet the needs of a dynamic East African payments environment. As an AzamPay client, you will be able to leverage your presence across East Africa and extend your services regionally. Remember, we endeavour to follow you throughout your business adventure.



.

Base Urls

Sandbox

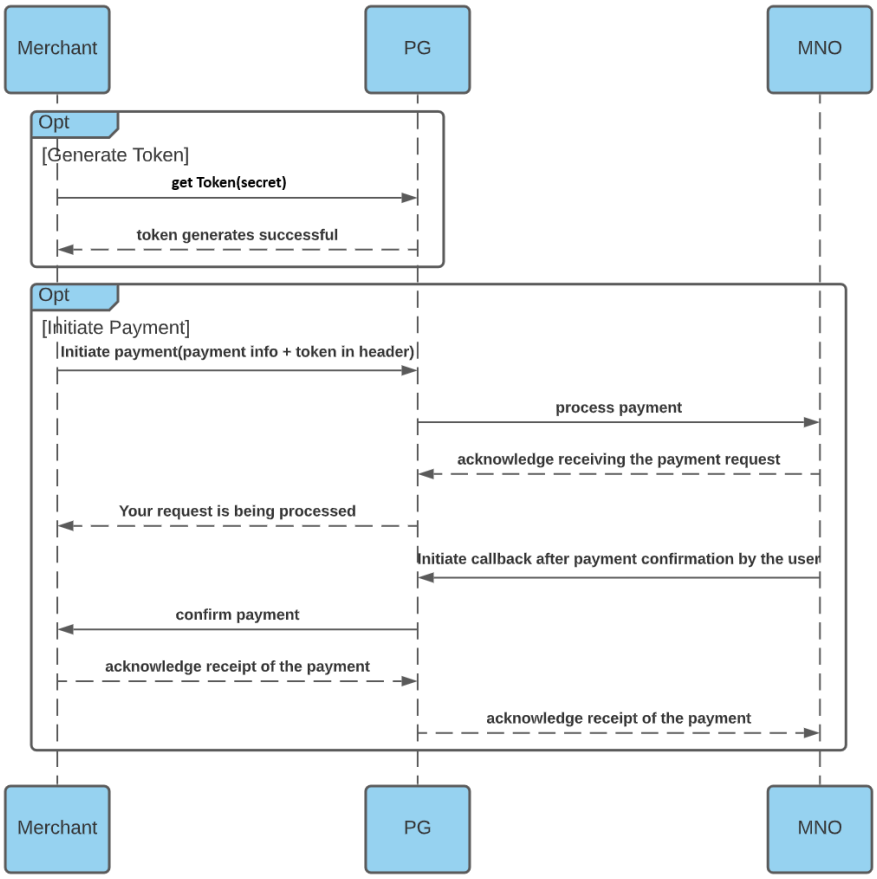
- **Authenticator Sandbox Base Url: https://authenticator-sandbox.azampay.co.tz.**
- **Azampay Sandbox Checkout Base Url: https://sandbox.azampay.co.tz.**

Azampay API Flow

All Azampay APIs follow two step process:

- Get token against the application authentication credentials.

Following diagram shows the general flow on how to consume the Azampay api.



Authentication

Azampay offers one form of authentication to ensure secure access to your account:

- Bearer Auth - an open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications.

Bearer Token is the JWT token that you get against your application Name, Client Id and Client Secret. For Sandbox Environment, You can get these application credentials from Sandbox portal. For production environment, you will be provided these keys after you submit your business KYC to Azampay from Sandbox portal.

Token Generation

Generate Token For App

POST `/AppRegistration/GenerateToken`

Generate the access token in order to access Azampay public end points.

REQUEST BODY SCHEMA: application/json

`appName` required	string It will be the name of application.
`clientId` required	string It will be the client id which generated during application registration.
`clientSecret` required	string It will be the secret key which generated during application registration.

Responses

> **200** Success

> **423** Invalid detail

— **500** Internal Server Error

Request samples

Payload | Node JS | .Net

Content type

application/json

Copy

```
{
    "appName": "string",
    "clientId": "string",
    "clientSecret": "string"
}
```

Response samples

200 | **423**

Content type

application/json

Copy Expand all Collapse all

```
{
 - "data": {
    + "accessToken": { … },
    + "expire": { … }
   },
   "message": "Token generated successfully",
   "success": true,
   "statusCode": 200
}
```


Checkout API

Mno Checkout

POST /azampay/mno/checkout

Checkout and make payment to requested provider.

AUTHORIZATIONS: *Bearer Auth*

REQUEST BODY SCHEMA: application/json

accountNumber required	string This is the account number/MSISDN that consumer will provide. The amount will be deducted from this account.
additionalProperties	object or null Total serialized size limit: 4 Kilobytes (4096 bytes).
amount required	number Must contain numeric characters only. Value range: 0 to 5,000,000.
currency required	string Maximum length: 32 characters.

`externalId` required	string Maximum length: 128 characters.
`provider` required	string (Provider) Enum: `"Airtel"` `"Tigo"` `"Halopesa"` `"Azampesa"` `"Mpesa"`

Responses

> **200** Success

> **400** Bad Request

— **500** Internal Server Error

Request samples

Payload | Node JS | .Net

Content type
application/json

Copy Expand all Collapse all


```
{
   "accountNumber": "string",
 - "additionalProperties": {
      "property1": null,
      "property2": null
   },
   "amount": 0,
   "currency": "string",
   "externalId": "string",
   "provider": "Airtel"
}
```


Response samples

200 400

Content type
application/json

Copy

```
{
    "success": true
}
```


Bank Checkout

POST /azampay/bank/checkout

Checkout and make payment to requested provider.

AUTHORIZATIONS: > *Bearer Auth*

REQUEST BODY SCHEMA: application/json

additionalProperties >	object or null Total serialized size limit: 4096 bytes (4 Kilobytes).
amount required	number Must contain numeric characters only. Value range: 0 to 5,000,000.
currencyCode required	string Cannot be null or empty.
merchantAccountNumber required	string Maximum length: 100 characters.
merchantMobileNumber required	string Maximum length: 100 characters.

`merchantName`	string or null Maximum length: 100 characters.
`otp` required	string Marked as Sensitive Data; ensure secure handling.
`provider` required	string (BankProvider) Enum: `"CRDB"` `"NMB"`
`referenceId`	string or null Maximum length: 128 characters.

Responses

> **200** Success

> **400** Bad Request

— **500** Internal Server Error

Request samples

Payload | **Node JS** | **.Net**

Content type

application/json

Copy Expand all Collapse all


```
{
 - "additionalProperties": {
      "property1": null,
      "property2": null
   },
   "amount": 0,
   "currencyCode": "string",
   "merchantAccountNumber": "string",
   "merchantMobileNumber": "string",
```

```
    "merchantName": "string",
    "otp": "string",
    "provider": "CRDB",
    "referenceId": "string"
}
```


Response samples

200 400

Content type
application/json

Copy


```
{
    "transactionId": "string",
    "message": "string",
    "success": true
}
```


Generate CRDB OTP

How to get CRDB OTP to activate your bank account

Dial *150*03# and Enter your SIM Banking PIN

Press 7 other services

Press 5 for azampay the select any of the below

Link Azampay Account > to generate OTP

Unlink Azampay Account > unlink linked account

Disconnect > disable linking

Generate NMB OTP

How to get NMB OTP to activate your bank account

Dial *150*66#

Press 8 More

Press 5 Register Sarafu

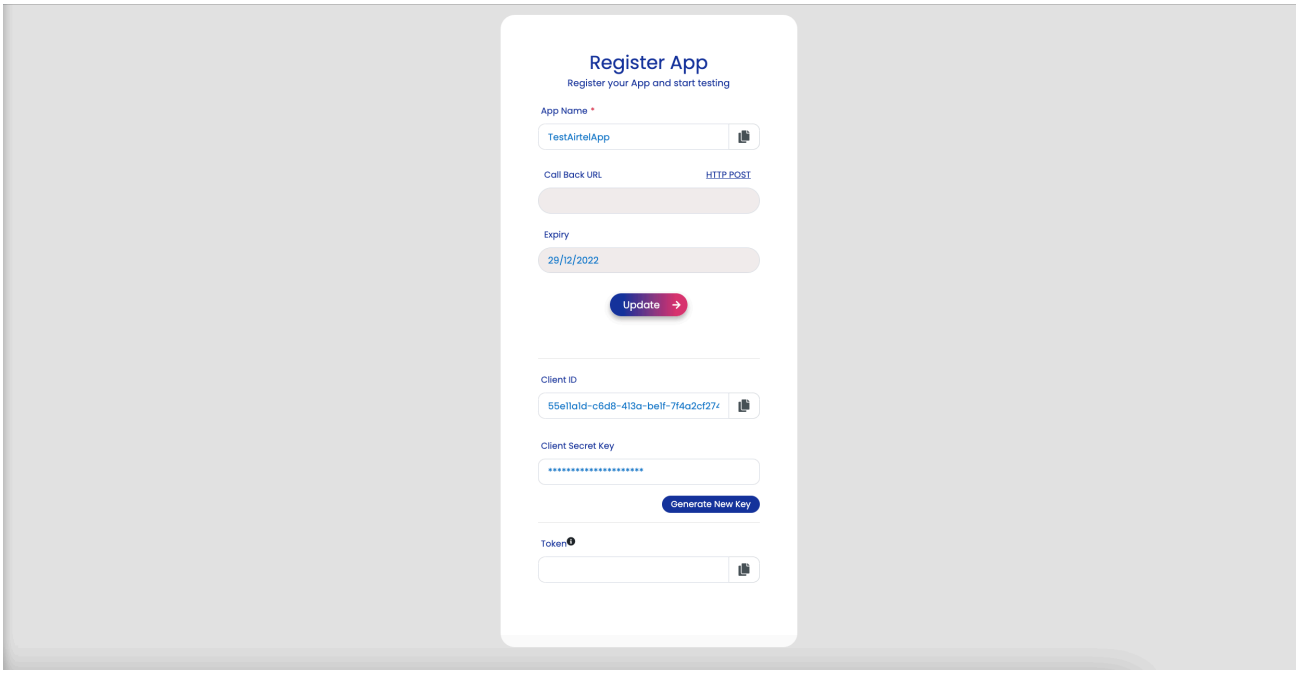
Press 1 Select Account No.

Callback

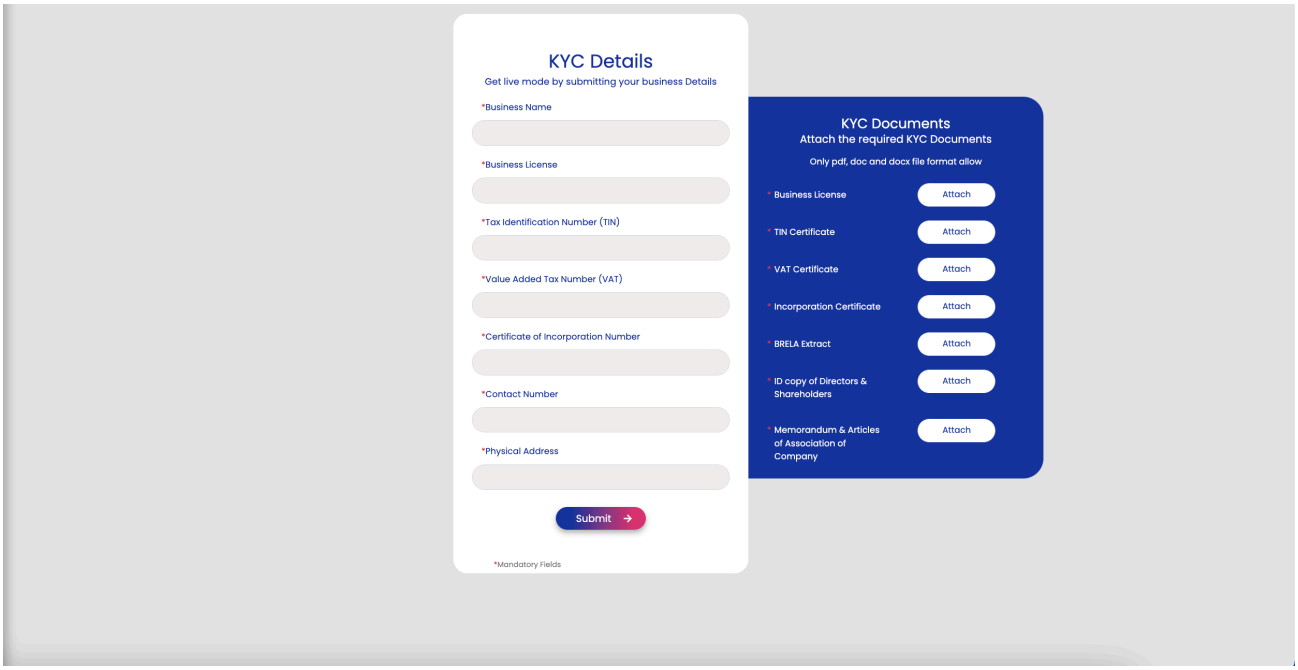
POST `/api/v1/Checkout/Callback`

This endpoint must be available in the your application all the time. This application will send transaction completion status to merchant application upon confirmation by user.

For Sandbox environment, the URL for this callback can be provided upon registering the app



For Production, after approval of submitted KYC



You will be asked to provide the production URL for the callback by the Payment Gateway Customer Care team to integrate.

Callback endpoint must follow below provided schema

REQUEST BODY SCHEMA: application/json

`additionalProperties` >	object or null Additional properties for extra data
`amount` required	string Transaction amount
`clientId` required	string Client identifier
`externalreference` required	string External reference ID
`message` required	string Transaction description message
`mnoreference` required	string Mobile network operator reference
`msisdn` required	string Mobile Subscriber ISDN Number

operator required	string Mobile network operator
password required	string Password for authentication
reference required	string Transaction reference ID
submerchantAcc	any or null Sub-merchant account (reserved for future use)
transactionstatus required	string Transaction status (success or failure)
transid required	string Transaction ID
user required	string User identifier
utilityref required	string Utility reference ID that belongs to the calling application

Responses

— **200** Success

— **500** Internal Server Error

Request samples

Payload | Node JS | .Net

Content type

application/json

Copy Expand all Collapse all

```
{
    "message": "string",
    "user": "string",
    "password": "string",
    "clientId": "string",
    "transactionstatus": "string",
    "operator": "string",
    "reference": "string",
    "externalreference": "string",
    "utilityref": "string",
    "amount": "string",
    "transid": "string",
    "msisdn": "string",
    "mnoreference": "string",
    "submerchantAcc": null,
  - "additionalProperties": {
        "property1": null,
        "property2": null
    }
}
```

Payment Partners

GET `/api/v1/Partner/GetPaymentPartners`

This endpoint will return the registered partners of the provided merchant

Partner List endpoint must follow below provided schema

Responses

> **200** Success

> **400** Bad Request

Request samples

Node JS | .Net

Copy

```
const fetch = require('node-fetch');
const data = {'currency':'TZS', 'id':'497f6eca-6276-4993-bfeb-53cbbbba6f08', l
const  headers = { 'Content-Type': 'application/json'};
fetch(`{BaseUrl}/api/v1/Partner/GetPaymentPartners`, {
 method: 'GET',
 headers,
 body: data,
})
.then(res => { /* response */})
.catch(err => {/* error */});
```

Response samples

200 | 400

Content type
application/json

Copy Expand all Collapse all


```
[
 - {
      "logoUrl": "string",
      "partnerName": "string",
      "provider": 0,
      "vendorName": "string",
      "paymentVendorId": "1213c943-b30e-4c9e-ac2f-d34796f01d2d",
      "paymentPartnerId": "70cd6bba-7f81-4ac8-9276-d5c0a189f2d4",
      "currency": "string"
   }
]
```

Checkout Pages

Post Checkout

POST `/api/v1/Partner/PostCheckout`

For this post request, send all params that are mentioned below to this end point.

This end point will respond back with the URL of your payments. Merchant Application can open this url in a new window to continue with the checkout process of the transaction

REQUEST BODY SCHEMA: application/json

`amount` required	string This is amount that will be charged from the given account.
`appName` required	string This is the application name
`cart` > required	object Shopping cart with multiple item
`clientId` required	string Client id is unique id for identify client
`currency` required	string Currency code that will convert amount into specific currency format
`externalId` required	string 30 characters long unique string
`language` required	string Language code for translate the application
`redirectFailURL` required	string URL that you want to redirected to at transaction failure

`redirectSuccessURL` required	string URL that you want to redirected to at transaction success
`requestOrigin` required	string URL which the request is being originated
`vendorId` required	string <uuid> Unique id for validate vendor
`vendorName` required	string Name of vendor

Responses

> **200** Success

Request samples

Payload | **Node JS**

Content type
application/json

Copy Expand all Collapse all


```
{
    "appName": "string",
    "clientId": "string",
    "vendorId": "e9b57fab-1850-44d4-8499-71fd15c845a0",
    "language": "string",
    "currency": "string",
    "externalId": "string",
    "requestOrigin": "string",
    "redirectFailURL": "string",
    "redirectSuccessURL": "string",
    "vendorName": "string",
    "amount": "string",
```

```
  - "cart": {
      + "items": [ … ]
    }
}
```


Response samples

200

Content type
application/json

Copy


```
"string"
```

Disbursement

Welcome to the API documentation for AzamPay, the premier payment platform for businesses in Tanzania. With our cutting-edge technology, we offer two essential APIs for businesses looking to manage their financial transactions seamlessly: our money transfer API for transfers from other countries to Tanzania and our disburse API for internal transfers within Tanzania.

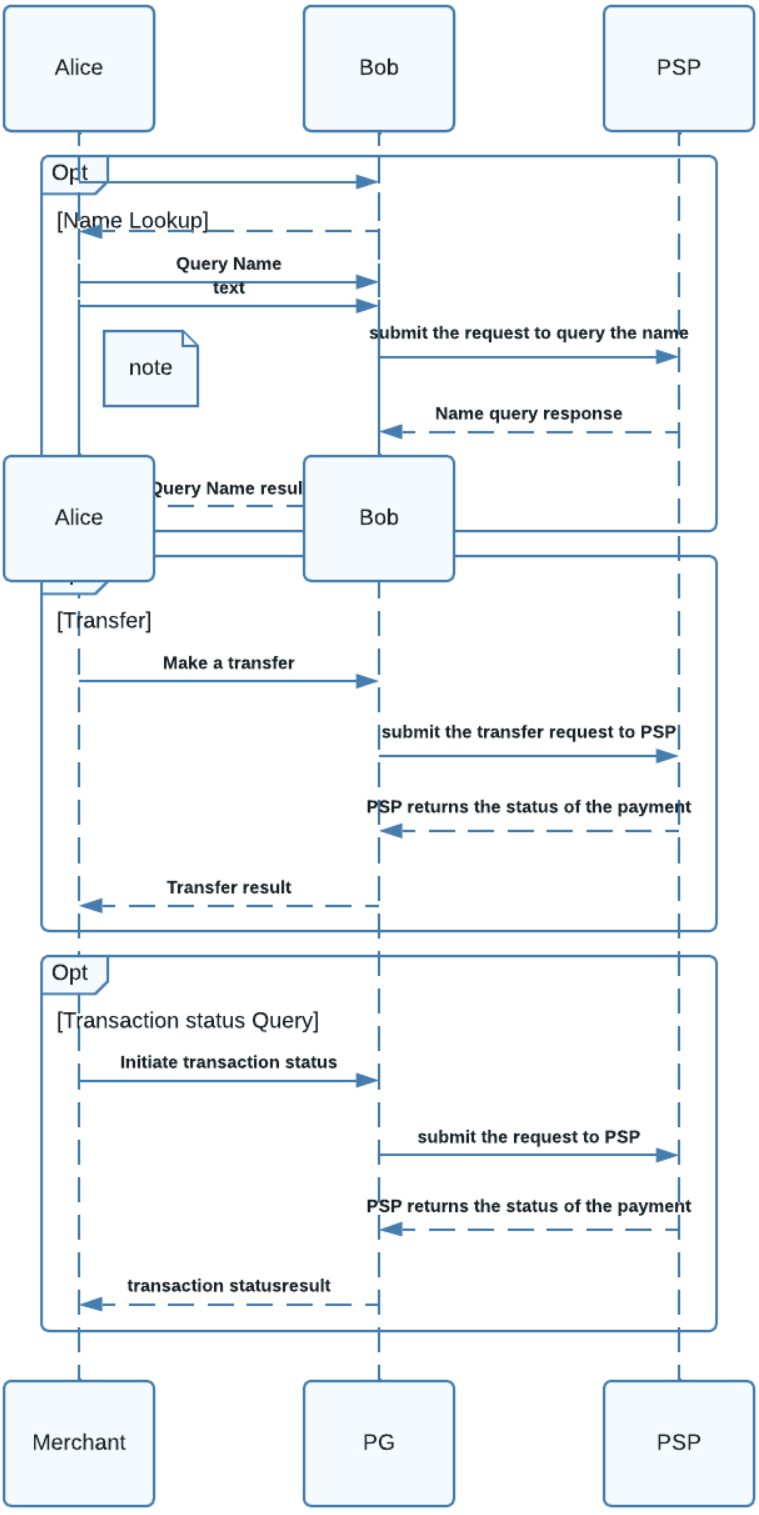
Our money transfer API offers businesses a reliable, fast, and secure way to receive international payments, with support for various payment methods and currencies. With our disburse API, businesses can easily disburse funds to employees, vendors, or other stakeholders within Tanzania, making payments simple, fast, and reliable.

This API documentation provides all the necessary information for developers to integrate these APIs into their applications easily. It includes details on authentication, endpoints, input parameters, response formats, and error handling, as well as code samples and SDKs to make integration a breeze.

With AzamPay's money transfer API and disburse API, businesses in Tanzania can now manage their finances more effectively, providing a hassle-free payment experience to their customers and stakeholders.

Disbursement API Flow

Following diagram shows the general flow on how to consume the Disbursement api.



Disburse

POST `/api/v1/azampay/disburse`

This API allows for the transfer of money from other countries to Tanzania. It requires the authorization token generated above, passed as a header in the request. The request should also contain details of the source, destination, and transfer details. Additionally, the request can include an external reference ID and remarks.

NOTE

Please contact us for the fields that will be used to calculate checksum.

AUTHORIZATIONS: > *Bearer Auth*

REQUEST BODY SCHEMA: application/json

`additionalProperties` >	object or null This is additional JSON data that calling application can provide. This is optional.
`checksum`	string A string containing the checksum.
`destination` > required	object Contains information about the destination account.
`externalReferenceId` required	string <= 30 characters An external reference ID to track the transaction which the initiator party sends in the request.
`remarks`	string or null Any remarks to be included with the transaction.
`source` > required	object Contains information about the source account.
`transferDetails` > required	object Contains information about the transfer.

Responses

> **200** Success

> **400** Bad Request

> **401** Unauthorized

Request samples

Payload | cURL | PHP | 400 | 400 | Node JS | Ruby | Java | GO

Content type

application/json

Copy Expand all Collapse all


```
{
  - "source": {
        "countryCode": "string",
        "fullName": "string",
        "bankName": "tigo",
        "accountNumber": "string",
        "currency": "string"
    },
  - "destination": {
        "countryCode": "string",
        "fullName": "string",
        "bankName": "tigo",
        "accountNumber": "string",
        "currency": "string"
    },
  - "transferDetails": {
        "type": "string",
        "amount": 0.1,
        "dateInEpoch": 0.1
    },
    "externalReferenceId": "string",
  - "additionalProperties": {
        "property1": null,
        "property2": null
    },
    "checksum": "string",
    "remarks": "string"
}
```

Response samples

200 | 400 | 401 | 400 | 400

Content type
application/json

Copy


```
{
    "pgReferenceId": "b42aeas4hl3d4f58bhfk4007782cb452",
    "message": "Your transaction is in process",
    "success": true,
    "statusCode": 200
}
```

Name Lookup

POST /api/v1/azampay/namelookup

This API is used to lookup the name associated with a bank account or Mobile Money account.

NOTE

Please contact us for the fields that will be used to calculate checksum.

AUTHORIZATIONS: *Bearer Auth*

REQUEST BODY SCHEMA: application/json

`accountNumber`	string or null Bank account number or Mobile Money number.
`bankName`	string or null Bank name or Mobile Money name associated with the account.
`checksum`	string A string containing the checksum.

Responses

> **200** Success

> **401** Unauthorized

Request samples

Payload | cURL | Python | PHP | Node JS | .Net | Java | Ruby | GO

Content type
application/json

Copy

```
{
    "bankName": "string",
    "accountNumber": "string",
    "checksum": "string"
}
```

Response samples

200 | 401

Content type
application/json

Copy

```
{
    "fname": "string",
    "lname": "string",
    "name": "string",
    "message": "string",
    "status": true,
    "statusCode": 200,
    "accountNumber": "string",
    "bankName": "string"
```

```
}
```

Get Transaction Status

GET `/api/v1/azampay/transactionstatus`

This API allows you to retrieve the status of a disbursement transaction made through AzamPay.

AUTHORIZATIONS: > *Bearer Auth*

QUERY PARAMETERS

`bankName`	string The name of the mobile network operator (MNO) you used to make the disbursement request.
`pgReferenceId`	string To be generated by Azampay and will be used to track the status of the transaction, same will be returned in the callback.

Responses

> **200** Success

> **401** Unauthorized

Request samples

cURL | PHP | 200 | .Net | Node JS | Java | Ruby | GO

Copy

```
curl -X 'GET'
'${BaseUrl}/azampay/gettransactionstatus?pgReferenceId=TransactionId&bankName=
```

```
-H 'Authorization: Bearer YOUR_AUTHORIZATION_TOKEN_HERE'
```

Response samples

200 | 401 | 200

Content type
application/json

Copy

```
{
    "pgReferenceId": "b42aeas4hl3d4f58bhfk4007782cb452",
    "message": "Your transaction is in process",
    "success": true,
    "statusCode": 200
}
```

Checksum Calculation

- Calculate checksum using following method: Base64(RSA(SHA512(string))).
- For RSA Encryption, use PKCS1 Padding.
- Contact our team to obtain the necessary public key for the encryption process.

Request samples

PHP | Python | .Net | Node JS | Java | Ruby | GO

```
<?php

// Load the public key
$publicKey = openssl_pkey_get_public(file_get_contents('public_key.pem')); //

// Input string
$inputString = 'YourInputStringHere';
```

```
// Step 1: Calculate SHA-512 hash
$sha512Hash = hash('sha512', $inputString, true);

// Step 2: Encrypt with RSA using PKCS1 padding
openssl_public_encrypt($sha512Hash, $encryptedData, $publicKey, OPENSSL_PKCS1_

// Step 3: Encode the encrypted result in Base64
$base64Checksum = base64_encode($encryptedData);

echo 'Encrypted Checksum: ' . $base64Checksum;

?>
```

Callback

POST `/callback`

This endpoint must be available in the your application all the time. This application will send transaction completion status to merchant application upon confirmation by user.

Please contact us to configure the callback URL for your application.

REQUEST BODY SCHEMA: application/json

`additionalProperties` >	object or null This is additional JSON data that calling application can provide. This is optional.
`amount`	string This is amount that will be charged from the given account.
`fspReferenceId`	string It is the reference ID from partner FSP (Financial Service Provider)
`initiatorReferenceId`	string An external reference ID to track the transaction which the initiator party sends in the request
`message`	string

	This is transaction description message
`operator`	string (DisburseProvider) Enum: `"Airtel"` `"Azampesa"` `"Tigo"`
`pgReferenceId`	string To be generated by Azampay and will be used to track the status of the transaction, same will be returned in the callback.
`status`	string success or failure

Responses

— **200** Success

— **500** Internal Server Error

Request samples

Payload

Content type

application/json

Copy Expand all Collapse all


```
{
 - "additionalProperties": {
      "property1": null,
      "property2": null
   },
   "initiatorReferenceId": "string",
   "fspReferenceId": "string",
   "pgReferenceId": "string",
   "amount": "string",
   "status": "string",
   "message": "string",
   "operator": "Airtel"
```

```
}
```

Bill Pay API

Authentication

The API requires that a JWT token is passed in the Authorization header. This token will be generated by our system using the HMAC algorithm and hashed with SHA256. This is a symmetric algorithm, shortened as HS256, that uses a shared secret (to be provided) to sign the token. The generated token will have a maximum validity of 120 seconds. The JWT token will contain `userId`, expirationTime (`exp`), and issuedAtTime (`iat`). It is the merchant's responsibility to validate the token and the hash. **The secret key will be shared privately.**

Authentication Verification

To verify the JWT token, the merchant will need to:

1. Decode the JWT token using the shared secret key and HS256 algorithm.
2. Validate the `userId`, expirationTime (`exp`), and issuedAtTime (`iat`) claims.
3. Ensure that the expirationTime (`exp`) is within the 120-second validity period.
4. Confirm that the token has not expired by comparing the expirationTime (`exp`) with the current time.

Hash Calculation

The hash will be generated using all request data fields concatenated in a consistent order. The resulting string will then be hashed using SHA256 and signed using the same secret key used for JWT generation. The hash ensures the integrity and authenticity of the request data.

Example of hash calculation:

1. Convert the data object to a minified JSON string.
2. Compute the SHA256 hash of the JSON string.
3. Sign the hash with the secret key using HMAC-SHA256.

Hash Verification

To verify the hash, the merchant will need to:

1. Convert the data object to a JSON string.
2. Compute the SHA256 hash of the JSON string.
3. Sign the computed hash with the shared secret key using HMAC-SHA256.
4. Compare the resulting hash with the Hash field provided in the request. If they match, the request is verified; otherwise, it is considered tampered with.

Name Lookup API

POST `/api/merchant/name-lookup`

Merchants provide an endpoint for name lookup based on a BillIdentifier.

Hash Verification Example:

1. Convert the Data object to a minified JSON string.

```
{"BillIdentifier":"123456789","Currency":"TZS","Language":"en","Country
```

2. Compute the SHA256 hash of the JSON string.
3. SHA256 hash:

```
fe6173685ce463cc966f6fce41fd59a92f181d1d7e63b8a17a3e42e41e381cbd
```

4. Sign the hash with the secret key using HMAC-SHA256.

   Secret key (Used in sample):

```
TRlWYMsr0GQU15REn6KRVn2DDv55MMW9VjLfpQn9RVg6m8QaCGNfY9IuGy6sCB1
```

   HMAC-SHA256 signature:

```
711bd3c3e54488523683182aa01c8a83544e45b7ba2c45652e039bf830e5daf2
```

5. Compare the computed hash with the Hash field in the request. If they match, the request is verified.

AUTHORIZATIONS: *Bearer Auth*

REQUEST BODY SCHEMA: application/json

Data > required	object
Hash required	string SHA256 hash of the request data

Responses

> **200** Success

Request samples

Payload | cURL | .Net | Node JS | Java

Content type
application/json

Copy Expand all Collapse all


```
{
 - "Data": {
       "BillIdentifier": "123456789",
       "Currency": "TZS",
       "Language": "en",
       "Country": "Tanzania",
       "TimeStamp": "2024-06-12T10:20:30Z",
     + "AdditionalProperties": { … },
       "BillType": "Electricity"
   },
   "Hash": "78b1e51f6318f57f43995b08b48c2de6f84fbe94b2c0c4b3a7e77a7430b9b104"
}
```


Response samples

200

Content type
application/json

Copy


```
{
    "Name": "John Doe",
    "BillAmount": 150,
    "BillIdentifier": "123456789",
    "Status": "Success",
    "Message": "Name found for the provided BillIdentifier.",
    "StatusCode": 0
}
```

Payment API

POST /api/merchant/payment

Merchants provide an endpoint to process payments.

Hash Verification Example:

1. Convert the Data object to a JSON string.


```
{"BillIdentifier":"123456789","Currency":"TZS","Language":"en","Country
```


2. Compute the SHA256 hash of the minified JSON string.
3. SHA256 hash:


```
9e6c0c62bdbacc4aef78b422dcd142e2f1bf82b0afbe2224be627a0761ee5825
```


4. Sign the hash with the secret key using HMAC-SHA256.

   Secret key (Used in sample):


```
TRlWYMsr0GQU15REn6KRVn2DDv55MMW9VjLfpQn9RVg6m8QaCGNfY9IuGy6sCB1
```


   HMAC-SHA256 signature:


```
ca3f49b2ee7740bf68062445ee7027d33baa4dbb0e456cdc7534ddd34bf19fb1
```

5. Compare the computed hash with the Hash field in the request. If they match, the request is verified.

AUTHORIZATIONS: > *Bearer Auth*

REQUEST BODY SCHEMA: application/json

`Data` > required	object
`Hash` required	string The hash value for verifying the integrity of the request

Responses

> **200** Success

Request samples

Payload | cURL | .Net | Node JS | Java

Content type

application/json

Copy Expand all Collapse all

```
{
 - "Data": {
      "FspReferenceId": "fsp123456",
      "PgReferenceId": "pg123456",
      "Amount": 100,
      "BillIdentifier": "987654321",
      "PaymentDesc": "Utility Bill Payment",
      "FspCode": "FSP123",
      "Country": "Tanzania",
      "TimeStamp": "2024-06-12T10:20:30Z",
      "BillType": "Water",
   + "AdditionalProperties": { … }
   },
   "Hash": "9e4fbd74ae54f9f3fbd2e3d4c4cbe2b1b8a5675f78bbd2c13c5fba32d2f4c5b8"
}
```


Response samples

200

Content type
application/json

Copy


```
{
   "MerchantReferenceId": "merchant123456",
   "Status": "Success",
   "StatusCode": 0,
   "Message": "Payment successful."
}
```

Status Check API

POST /api/merchant/status-check

Merchants provide an API to check the status of a payment using the MerchantReferenceId.

AUTHORIZATIONS: > *Bearer Auth*

REQUEST BODY SCHEMA: application/json

`MerchantReferenceId` required	string The reference ID provided by the merchant

Responses

› **200** Success

Request samples

Payload | cURL | .Net | Node JS | Java

Content type

application/json

Copy

```
{
    "MerchantReferenceId": "merchant123456"
}
```

Response samples

200

Content type

application/json

Copy

```
{
    "MerchantReferenceId": "merchant123456",
    "Status": "Success",
    "StatusCode": 0,
    "Message": "Payment successful."
}
```