MarzPay

Merchant API Integration Guide

Africa-focused payments platform — collections, disbursements, bank transfers, bills, airtime, webhooks, and more.

Currently live in Uganda · Expanding across Africa

Version 1.0 | August 2, 2026

Online documentation: https://wallet.wearemarz.com/documentation

Table of Contents

1. Introduction

MarzPay is an **Africa-focused** payment platform that enables businesses to collect and disburse funds across the continent. The platform supports mobile money, card payments, bank transfers, utility bills, airtime/data bundles, phone verification — through the merchant REST API documented in this guide.

> **Regional availability:** MarzPay is built for Africa. **Uganda (UG)** is the first live market (currency: UGX). Country codes, providers, and utilities in this guide reflect the current Uganda rollout. Additional African markets will use the same API patterns as they go live — check `GET /collect-money/services` and `GET /services` for your account's enabled countries.

This guide walks you through integrating the merchant API from account setup to production go-live, covering every product available today.

Product overview

Product	Endpoint	IP whitelist
Collections (mobile money)	POST /collect-money	No
Card payments	POST /collect-money (method: card)	No
Send money	POST /send-money	Yes
Bank transfer	POST /bank-transfer	Yes
Bill payments	POST /bill-payment	Yes (POST)
Airtime & data	POST /airtime-data	Yes (POST)
Phone verification	POST /phone-verification/verify	No
Balance	GET /balance	Yes
Transactions	GET /transactions	No

Webhooks	POST /webhooks	No
Payment links	Dashboard (hosted checkout)	—

MarzPay is also available on **mobile app**, **WhatsApp**, and **USSD** for businesses and customers — see Section 9. Those channels are not part of the merchant API in this guide.

> **Before you integrate:** Subscribe to each product in the MarzPay dashboard service marketplace. Some endpoints also require IP whitelisting and disbursement permissions.

2. Getting Started

2.1 Account setup

1. Create a business account at https://wallet.wearemarz.com
2. Verify your email and complete your business profile
3. Upload required business documents for verification
4. Subscribe to the payment services you need (Collections, Send Money, etc.)

2.2 API credentials

1. Log in to the dashboard and navigate to **API Keys**
2. Click **Generate New API Key** and provide a descriptive name
3. Copy your API key and secret immediately — the secret is shown only once
4. Store credentials securely (environment variables, secrets manager)

2.3 Sandbox mode

New accounts start in **sandbox mode**. In sandbox, API calls return simulated responses without processing real money. Test your full integration flow before requesting live mode activation.

> **Important:** Never commit API keys to source control. Use `.env` files and keep production credentials separate from development.

3. Authentication & Security

3.1 HTTP Basic authentication

Every API request must include an Authorization header with your API key and secret:

```
Authorization: Basic base64(api_key:api_secret)
Accept: application/json
Content-Type: application/json
```

Example: if your key is `mk_live_abc` and secret is `sk_live_xyz`:

```
Authorization: Basic bWtfbGl2ZV9hYmM6c2tfbGl2ZV94eXo=
```

3.2 IP whitelisting

The following endpoints require your server IP to be whitelisted in the dashboard:

- GET /balance
- POST /send-money
- POST /bank-transfer
- POST /bill-payment
- POST /airtime-data

3.3 Disbursement permission

Send money and bank transfer require **API disbursement** to be enabled on your business account. Contact support or enable this in your dashboard settings.

3.4 Recommended environment variables

```
MARZPAY_API_BASE=https://wallet.wearemarz.com/api/v1
MARZPAY_API_KEY=your_api_key
MARZPAY_API_SECRET=your_api_secret
MARZPAY_CALLBACK_URL=https://your-app.com/webhooks/marzpay
```

4. API Fundamentals

4.1 Base URL

```
https://wallet.wearemarz.com/api/v1
```

All paths in this guide are relative to this base URL.

4.2 Request format

- Send JSON bodies with `Content-Type: application/json`
- Some collection endpoints also accept `multipart/form-data`
- **Current live market:** Uganda (`country: UG`), amounts in UGX (Ugandan Shillings)
- Phone numbers use E.164 format, e.g. `+256712345678` for Uganda
- As new African markets launch, country codes and currencies will expand — always check services endpoints for your account

4.3 Success response envelope

```
{
  "status": "success",
  "message": "Optional human message",
  "data": { ... }
}
```

4.4 Wallets

Every business has two wallets:

Wallet	wallet_source	Funded by
Main wallet	main (default)	Mobile money collections
Card wallet	card	Card payment collections

5. Collections (Mobile Money)

Collect payments from customers via MTN or Airtel Mobile Money. Provider is auto-detected from the phone number.

5.1 Endpoints

Method	Path	Description
POST	/collect-money	Initiate collection
GET	/collect-money/services	Available countries/providers
GET	/collect-money/{uuid}	Transaction status

5.2 Create collection — request

```
{
  "amount": 5000,
  "phone_number": "+256712345678",
  "reference": "c97fae8b-9b7f-4192-9f72-6f0859d33e67",
  "country": "UG",
  "description": "Order #1042",
  "callback_url": "https://your-app.com/webhooks/marzpay"
}
```

5.3 Field requirements

Field	Required	Rules
amount	Yes	500 – 10,000,000 UGX
reference	Yes	UUID v4, unique per API collection
country	Yes	UG
phone_number	Yes	E.164 format (+256...)
description	No	Max 255 characters

callback_url	No	HTTPS webhook URL

> **Reference rules:** Every API collection must use a new UUID v4 reference. Reusing a reference returns `DUPLICATE_REFERENCE`. Payment links auto-generate references; API collections do not.

5.4 Integration flow

1. Generate a new UUID v4 reference for each collection
2. POST /collect-money with customer phone and amount
3. Customer receives USSD/SMS prompt on their phone
4. Wait for webhook callback or poll GET /collect-money/{uuid}
5. Mark order paid only on **completed/successful** status

Note: `provider_reference` is often null on create. Use `collection.provider_transaction_id` from webhook payloads for provider IDs.

6. Card Payments

Use the same `POST /collect-money` endpoint with `"method": "card"`. Omit `phone_number`.

```
{
  "amount": 5000,
  "method": "card",
  "reference": "123e4567-e89b-12d3-a456-426614174000",
  "country": "UG",
  "description": "Order payment",
  "callback_url": "https://your-app.com/thank-you"
}
```

The response includes `data.redirect_url` — redirect the customer to that URL to complete payment on the card gateway. Card funds settle in the **card wallet**.

7. Send Money & Bank Transfer

7.1 Send money (mobile money disbursement)

Send funds to a customer’s mobile money wallet (MTN or Airtel — auto-detected from phone number).

Method	Path	Description
POST	/send-money	Disburse to phone
GET	/send-money/services	Limits and allowed phones
GET	/send-money/{uuid}	Transaction status

```
POST /send-money
{
  "amount": 10000,
  "phone_number": "+256712345678",
  "reference": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "country": "UG",
```

```
  "description": "Salary payment",
  "callback_url": "https://your-app.com/webhooks/marzpay"
}
```

Requires: IP whitelist, API disbursement permission, disbursement service subscription, sufficient main-wallet balance. Some businesses restrict payouts to pre-registered withdrawal numbers. Always check `GET /balance` first.

7.2 Bank transfer

Push funds from your MarzPay wallet to a supported bank account. Available in the current Uganda rollout; designed to extend to additional African banking networks as markets launch.

Method	Path	Description
POST	/bank-transfer/validate	Validate account before sending (recommended)
POST	/bank-transfer	Create bank transfer
GET	/bank-transfer/banks	List supported banks
GET	/bank-transfer/services	Limits and tiered pricing
GET	/bank-transfer/{reference}	Transfer status

Step 1 — Validate account

```
POST /bank-transfer/validate
{
  "bank_name": "Equity Bank",
  "account_number": "60001256421"
}
```

Step 2 — Create transfer

```
POST /bank-transfer
{
  "amount": 100000,
  "description": "Vendor payment",
```

```
  "bank_name": "Equity Bank",
  "bank_account_number": "60001256421",
  "bank_account_name": "John Doe",
  "bank_branch": "Kampala",
  "wallet_source": "main"
}
```

Important: You pay *amount + tiered charge*; the recipient receives the stated amount. Minimum transfer is approximately 2,500 UGX (confirm via `GET /bank-transfer/services`). Balance is debited immediately; failed transfers are refunded. Poll `GET /bank-transfer/{reference}` for final status. Use `wallet_source: main|card` to debit the main or card wallet.

Requires: IP whitelist, API disbursement permission, Bank Transfer service subscription.

8. Bill Payments, Airtime & Phone Verification

8.1 Bill payments

Pay utility and subscription bills in the current Uganda market: electricity (LIGHT/UMEME), water (NWSC), and TV (DSTV, GOTV).

Method	Path	Description
POST	/bill-payment/verify	Verify meter or account
POST	/bill-payment	Pay bill
GET	/bill-payment	List transactions
GET	/bill-payment/services	Utility requirements
GET	/bill-payment/nwsc/areas	NWSC service areas
GET	/bill-payment/dstv/bouquet-codes	DSTV packages and prices
GET	/bill-payment/gotv/bouquet-codes	GOTV packages and prices
GET	/bill-payment/{reference}	Payment status

Verify before paying

```
LIGHT:  { "utility_code": "LIGHT", "meter_number": "12345678901" }
NWSC:   { "utility_code": "NWSC", "meter_number": "12345678901", "area": "Kampala" }
DSTV:   { "utility_code": "DSTV", "meter_number": "7039132763" }
```

Pay bill (LIGHT example)

```
POST /bill-payment
{
  "reference": "550e8400-e29b-41d4-a716-446655440000",
  "utility_code": "LIGHT",
  "meter_number": "12345678901",
  "phone_number": "+256700000000",
  "amount": 10000,
```

```
  "customer_name": "John Doe",
  "email": "john@example.com",
  "callback_url": "https://your-app.com/webhooks/marzpay"
}
```

NWSC: include `area` from `/bill-payment/nwsc/areas`. **DSTV/GOTV:** include `bouquet_code` from bouquet-codes endpoint; amount must exactly match bouquet price. Wallet debited for amount + bill payment charge (~1,500 UGX). Auto-refund on failure.

8.2 Airtime & data

Purchase MTN, Airtel, or Lyca airtime and data bundles. **No merchant webhooks** — poll status endpoint for final delivery (especially Airtel data bundles).

Method	Path	Description
GET	/airtime-data/catalog	Bundles by network
GET	/airtime-data/detect-network	Preview network from MSISDN
POST	/airtime-data	Purchase airtime or bundle
GET	/airtime-data	List purchases
GET	/airtime-data/{reference}	Purchase status

Network	Airtime	Data bundles	Delivery
MTN	Yes	Yes	Immediate
Airtel	Yes	Yes	Data may be async (pending)
Lyca	Yes	No	Immediate

```
POST /airtime-data  (airtime)
{
  "reference": "550e8400-e29b-41d4-a716-446655440000",
  "purchase_type": "airtime",
  "msisdn": "256771234567",
  "amount": 5000
```

```
}
```

Network is auto-detected from MSISDN — do not send a separate network field. Use a unique UUID reference per purchase.

8.3 Phone verification

Verify mobile numbers and retrieve registered subscriber name (KYC) before onboarding customers.

Method	Path	Description
POST	/phone-verification/verify	Lookup subscriber details

```
POST /phone-verification/verify
{
  "phone_number": "+256712345678"
}
```

Currently supports Uganda mobile numbers. Additional African markets will be added as verification providers are connected per country.

8.4 Payment links (dashboard)

Create hosted payment pages from the MarzPay dashboard (fixed or flexible amount, mobile money + card). Share `https://wallet.wearemarz.com/pay/{uuid}` with customers. Configure callback and redirect URLs on the link. No `POST /payment-links` on standard API-key routes — create links from the MarzPay dashboard.

9. Mobile App, WhatsApp & USSD

In addition to the merchant API in this guide, MarzPay is available on **mobile app**, **WhatsApp**, and **USSD**. These are ready-made channels for businesses and their customers — you do not integrate them via the API endpoints in sections 5–8.

9.1 Mobile app (iOS & Android)

Download the MarzPay app to check balances, send money, buy airtime, manage payments, and run your business on the go. Sign in with the same account you use on the web dashboard.

- **App Store (iOS):** https://apps.apple.com/us/app/marzpay/id6769404051
- **Google Play (Android):** https://play.google.com/store/apps/details?id=com.marzinnovations.marzpay&pli=1
- **Download page:** https://wallet.wearemarz.com/download

9.2 WhatsApp

MarzPay on WhatsApp lets businesses and customers collect, pay, check balance, and complete everyday payments through chat. To enable WhatsApp for your business or learn your business WhatsApp number, contact MarzPay support.

9.3 USSD

MarzPay on USSD lets customers and businesses access key payment services from any mobile phone by dialling a short code (e.g. `*XXX#`). To provision USSD for your business, contact MarzPay support.

> **For API integrators:** This PDF covers the merchant REST API only. App, WhatsApp, and USSD are separate products — use the store links above for the app, or contact support for WhatsApp and USSD access.

10. Balance, Transactions & Services

10.1 Balance

```
GET /balance
```

Returns main wallet, card wallet, account status, and monthly summaries. Requires IP whitelist.

10.2 Transactions

```
GET /transactions?per_page=25&type=collection&status=successful
GET /transactions/{uuid}
```

Filter by type, status, provider, date range, and reference. Use as fallback when webhooks are delayed.

10.3 Services

```
GET /services
GET /services/{uuid}
```

Check which products your business is subscribed to before building integration UIs.

11. Webhooks & Callbacks

11.1 Per-request callback_url

Pass `callback_url` on collect-money, send-money, bill-payment, and similar endpoints. MarzPay POSTs JSON to your URL when the transaction reaches a final status.

11.2 Dashboard webhooks (API-managed)

Method	Path	Description
GET	/webhooks	List webhooks
POST	/webhooks	Create webhook
GET	/webhooks/{uuid}	Get details
PUT	/webhooks/{uuid}	Update
DELETE	/webhooks/{uuid}	Delete

11.3 Webhook handler checklist

1. Accept POST requests and return **HTTP 200** quickly
2. Process asynchronously if needed (queue the work)
3. Make handlers **idempotent** — the same reference may be delivered more than once
4. Update your database by `reference` or transaction UUID
5. Use HTTPS endpoints in production
6. Optionally verify HMAC signatures if enabled in dashboard

11.4 Example webhook payload (collection completed)

```
{
  "event_type": "collection.completed",
  "transaction": {
    "uuid": "...",
    "reference": "...",
```

```
    "status": "completed",
    "amount": { "raw": 5000, "currency": "UGX" }
  },
  "collection": {
    "provider": "mtn",
    "phone_number": "+256712345678",
    "provider_transaction_id": "MTN_..."
  }
}
```

12. Error Handling

12.1 Error response format

```
{
  "status": "error",
  "message": "Human-readable explanation",
  "error_code": "VALIDATION_ERROR",
  "errors": { "field": ["message"] }
}
```

12.2 Common error codes

error_code	Meaning	Action
VALIDATION_ERROR	Invalid input	Fix fields in errors object
DUPLICATE_REFERENCE	Reference reused	Generate new UUID
INSUFFICIENT_BALANCE	Low wallet balance	Top up or reduce amount
SERVICE_NOT_SUBSCRIBED	Missing subscription	Subscribe in dashboard
UNAUTHORIZED	Bad credentials	Check API key/secret
FORBIDDEN	Permission denied	Check IP whitelist / permissions
NOT_FOUND	Resource missing	Verify UUID/reference
ACCOUNT_FROZEN	Account frozen	Contact support
SERVER_ERROR	Internal error	Retry with exponential backoff

12.3 HTTP status codes

Code	Meaning
200 / 201	Success
400	Bad request

401	Unauthorized — invalid credentials
403	Forbidden — IP or permission issue
404	Not found
422	Validation error
500	Server error — retry later

13. Sandbox Testing & Go-Live Checklist

13.1 Sandbox testing

- Confirm account is in sandbox mode
- Test collection create → webhook → status update flow
- Test duplicate reference rejection
- Test error responses (invalid phone, missing fields)
- Use the online API playground: https://wallet.wearemarz.com/documentation/api-playground

13.2 Go-live checklist

☐ Business verification completed

☐ Live mode activated on account

☐ Subscribed to all required services

☐ Production API keys generated and stored securely

☐ Server IPs whitelisted for balance/disbursements

☐ Disbursement permission enabled (if sending money)

☐ Webhook endpoint live on HTTPS with idempotent handler

☐ Callback URLs use production domains (not localhost)

☐ Logging and monitoring in place for failed transactions

☐ Customer support process for failed payments defined

13.3 Implementation best practices

1. Generate a **new UUID v4** for every collection and airtime/data reference
2. Never mark an order paid on the create response alone — wait for webhook or confirmed GET status
3. Store credentials in environment variables only
4. Validate meters and bank accounts before initiating payments

5. Use the correct `wallet_source` for bank transfers
6. Poll airtime/data status when response is pending

14. Resources & Support

Resource	URL
Documentation home	https://wallet.wearemarz.com/documentation
Getting started	https://wallet.wearemarz.com/documentation/getting-started
API reference	https://wallet.wearemarz.com/documentation/api
API playground	https://wallet.wearemarz.com/documentation/api-playground
Integration tools (OpenAPI, Postman, AI skill)	https://wallet.wearemarz.com/documentation/ai-integration
Collections guide	https://wallet.wearemarz.com/documentation/collections
Send money guide	https://wallet.wearemarz.com/documentation/send-money
Bank transfer guide	https://wallet.wearemarz.com/documentation/bank-transfer
Bill payments guide	https://wallet.wearemarz.com/documentation/bill-payments
Airtime & data guide	https://wallet.wearemarz.com/documentation/airtime-data
Phone verification guide	https://wallet.wearemarz.com/documentation/phone-verification
Webhooks guide	https://wallet.wearemarz.com/documentation/webhooks
API base URL	https://wallet.wearemarz.com/api/v1

Code example (PHP / Laravel)

```
use Illuminate\Support\Facades\Http;
use Illuminate\Support\Str;

$response = Http::withBasicAuth(
```

```
    config('services.marzpay.key'),
    config('services.marzpay.secret')
)->post(config('services.marzpay.base_url').'/collect-money', [
    'amount'        => 5000,
    'phone_number'  => '+256712345678',
    'reference'     => (string) Str::uuid(),
    'country'       => 'UG',
    'callback_url'  => config('services.marzpay.callback_url'),
]);
```

Code example (Node.js)

```
const auth =
Buffer.from(`${process.env.MARZPAY_API_KEY}:${process.env.MARZPAY_API_SECRET}`).toString
('base64');

const res = await fetch(`${process.env.MARZPAY_API_BASE}/collect-money`, {
  method: 'POST',
  headers: { Authorization: `Basic ${auth}`, 'Content-Type': 'application/json' },
  body: JSON.stringify({
    amount: 5000, phone_number: '+256712345678',
    reference: crypto.randomUUID(), country: 'UG',
    callback_url: process.env.MARZPAY_CALLBACK_URL,
  }),
});
```

> For the latest endpoint details, request/response schemas, and product-specific guides, always refer to the online documentation at https://wallet.wearemarz.com/documentation. Download OpenAPI and Postman files from the Integration Tools page.