BluPay

INTEGRATION CATALOG

REST API & Payment Webhooks

The web services a site needs to connect the BluPay gateway and confirm payments — with authentication, the call sequence, and the signed confirmation webhook.

INSIDE

How it works · Authentication · Web services · Payment webhook

How it works

BluPay is a custodial crypto payment gateway. Your site integrates over a single typed REST API: authenticate with a per-shop API key, create an invoice (you quote in fiat — the crypto amount is locked at creation), and redirect the buyer to the hosted checkout returned as paymentUrl. BluPay allocates the deposit address, watches the chain, and confirms the payment by POSTing an HMAC-signed webhook to your URL. You can also poll the invoice for its status. There are no wallets or private keys to manage.

1. **Create**
   POST /v1/invoices/ with the amount and your order reference. You get back an invoice link and a paymentUrl.
2. **Pay**
   Redirect the buyer to paymentUrl (hosted checkout). They pick an asset + network and send funds.
3. **Confirm**
   Receive a signed invoice.paid webhook at your webhookUrl — and/or GET /v1/invoices/{link}/ to poll the status.

Base URL & authentication

Base URL	`https://blupay.me/api`
Auth header	`X-API-Key: <your per-shop key>`
Scopes	`currencies:read · shops:read/write · invoices:read/write`
Idempotency	`send an Idempotency-Key header on POSTs for safe retries`
Security	`production keys require an IP allow-list`

Web services

1 · Discover & set up

GET `/v1/currencies/{type}/`

List the enabled currencies. {type} is "crypto" or "fiat".

Scope	currencies:read
Use	discover supported assets/networks and each currencyNetworkId (for acceptedCurrencyNetworkIds)

GET `/v1/shops/`

List your shops. Returns each shopId (uuid) used when creating invoices.

Scope	shops:read
Related	POST /v1/shops/ (shops:write) · GET & PUT /v1/shops/{shopId}/

2 · Take a payment

POST `/v1/invoices/`

Create an invoice — the gateway charge. Quote in fiat; redirect the buyer to paymentUrl.

Scope	invoices:write · accepts Idempotency-Key
Body	currencyAmount, currencySymbol, shopId?, orderId?, orderDescription?, customerEmail?, redirectUrl?, webhookUrl?, timeForPayment? (5–180m), acceptedCurrencyNetworkIds?
Returns	link, status, currencyAmount, currencySymbol, remainingCurrencyAmount, paymentUrl

3 · Confirm a payment

GET **`/v1/invoices/{link}/`**

Fetch a single invoice and poll its status to confirm payment.

Scope	invoices:read
status	Created · Pending · Paid · Partially · Expired · Canceled
Returns	currencyAmount, remainingCurrencyAmount, status, paymentUrl

4 · Records (optional)

GET **`/v1/invoices/`**

List invoices with filters.

Scope	invoices:read
Query	status, shopId, from, to, search, source, page, pageSize

GET **`/v1/invoices/{link}/pdf/`**

Download a PDF copy of an invoice.

Scope	invoices:read

GET **`/v1/invoices/export/csv/`**

Stream all invoices as CSV.

Scope	invoices:read

Payment confirmation webhook

On every state change BluPay POSTs a JSON event to the invoice's webhookUrl. Verify the signature, enforce a 5-minute replay window on the timestamp, then respond 200. Failed deliveries are retried with exponential backoff.

Request headers

X-BluPay-Event-Id	unique id of this event (e.g. evt_…)
X-BluPay-Timestamp	unix-millis decimal string
X-BluPay-Signature	HMAC-SHA256(timestamp + rawBody), 64-char lowercase hex

Body

{ }	event_id, event_type, created_at, data:{ invoice }

Signature

HMAC	sign the exact bytes received, keyed by your per-shop webhook secret

event_type values

invoice.created · invoice.pending · invoice.paid · invoice.partially_paid · invoice.partially_expired · invoice.expired · invoice.canceled (invoice.paid = fully settled)

Get a per-shop API key from your dashboard

blupay.me · blupay.me/developers · blupay.me/contact