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 in an enabled quote currency, 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. The crypto quote is created when an asset and network are selected, not at invoice creation. Your integration does not manage deposit private keys.

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, shops:write; invoices:read, invoices:write
Idempotency	Invoice creation: save one key per attempt; reuse it with the identical body on retries.
Security	IP allow-lists are optional. Keep API keys and signing secrets on your server.

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 shops owned by the authenticated merchant. Invoice creation uses the API key shop.

Scope	shops:read
Related	POST /v1/shops/ (shops:write); GET /v1/shops/{shopId}/ (shops:read); PUT /v1/shops/{shopId}/ (shops:write).
Setup	Create the first shop in the dashboard. websiteUrl or telegramBotUsername is required. Each new shop returns webhookSecretPlain; save it securely. Callback URLs are supplied per invoice.

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 (positive decimal string, max 12 integer digits and 8 decimals), currencySymbol; optional orderId, orderDescription, customerEmail, redirectUrl, webhookUrl, timeForPayment (integer 5-180 minutes), acceptedCurrencyNetworkIds.
Policy	shopId is optional and must match the key shop. Sandbox is inherited from the shop, not an invoice request flag. Callback URLs must use its registered website host or a subdomain.
Retries	Reuse the same Idempotency-Key and request body. A changed body or in-flight request returns 409.
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 · PartiallyExpired · Expired · Canceled
Returns	currencyAmount, currencySymbol, orderId, link, remainingCurrencyAmount, status, paymentUrl. Match these to the saved order; fulfil only once for Paid under its tolerance policy.

4 · Read-only conversion diagnostics

GET `/v1/quotes/context/?currencySymbol=IRT`

Read conversion availability for the API key shop. No invoice or funds are created.

Scope	currencies:read
Returns	supported, available, reason, mode, rateMode, targets and rate provenance. Unavailable rates return available=false; do not invent a fallback rate.

POST `/v1/quotes/preview/`

Preview a checkout amount without reserving a wallet, funds or a rate.

Scope	currencies:read
Body	currencySymbol (uppercase), currencyAmount (decimal string), currencyNetworkId (positive integer). Other fields, including shopId or rate overrides, are rejected.
Returns	indicative=true; cryptoAmount, finalCryptoAmount, fee amounts, rate.ratio and rate.expiresAt. A preview is not a payment quote or proof of payment; unavailable live rates return 503.

5 · 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. Persist events in a durable inbox with a UNIQUE event_id before returning 200; duplicates are successful no-ops. Match invoice_link, order_id, currency_amount and currency_symbol to the stored order and fulfil once for invoice.paid with status Paid. Browser redirects never prove payment.

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_link, status, order_id, ... }. Paid events include currency_amount and currency_symbol (snake_case). Additional fields vary by event.

Signature

HMAC	sign the exact bytes received, keyed by your per-shop webhook secret
Retries	Delivery retries retain event_id but may refresh the timestamp and signature. Deduplicate by event_id, not the signature. Versioned invoice events can also include schema_version and aggregate.revision.

event_type values

invoice.created · invoice.pending · invoice.paid · invoice.partially_paid · invoice.partially_expired · invoice.expired · invoice.canceled (invoice.paid = Paid under the configured payment/tolerance policy)

Get a per-shop API key from your dashboard

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