zGate integration guide

zPayLink developer kit, 2026-10-15. Everything below is generated from the verified zPayLink developer documentation; examples use fictional values.

1. Product overview

USDT (BEP-20) payments on BNB Smart Chain: create an invoice, show its unique deposit address, hosted page or payment link, and receive signed webhooks as the payment is detected, confirmed and settled.

> The test environment on BNB Smart Chain Testnet is open to merchants with a key. On Mainnet, invoices are accepted only within a small, operator-set limit per merchant while the launch is limited. Merchant balance withdrawals, sweeps and refunds are not enabled.

Aspect	Status (2026-10-15)
Implemented	yes
Reachable from the internet	yes (see the connectivity note)
Testing	yes, BNB Smart Chain Testnet with a test key
Production	limited Mainnet launch: invoices within operator-set limits
Disabled or restricted	merchant withdrawals, sweeps, refunds; keys and webhook endpoints are set up by zPayLink

Connectivity: zPayLink servers answer from most networks, but some international networks currently cannot open a connection to them. If your servers time out, test from another network and send zPayLink your public IP and a traceroute.

2. Prerequisites

- Keys are issued by zPayLink when your merchant account is set up: test keys work only on the test URL, live keys only on production. The full key is shown once; zPayLink stores only a keyed hash. Default scopes: invoices:read, invoices:write, transactions:read, webhooks:write, balance:read; a read-only key has the read scopes only.
- A server-side environment (the APIs do not allow browser calls) with HTTPS outbound access to the base URL.
- An HTTPS endpoint on a public address for webhooks (where the product sends them).

3. Authentication

Send the key on every request: `Authorization: Bearer <key>`. Key format: `zgk_test_… / zgk_live_…`.

```
export ZGATE_API_KEY="<your key>"   # keep it in a secret store or environment variable
curl https://gate.zpaylink.com/staging/api/v1/invoices -H "Authorization: Bearer $ZGATE_API_KEY"
```

Environment	Base URL
Test (BNB Smart Chain Testnet, chain 97)	`https://gate.zpaylink.com/staging/api/v1`
Production (BNB Smart Chain, chain 56)	`https://gate.zpaylink.com/api/v1`

4. Step-by-step integration

1. Ask zPayLink for a merchant account; you receive a test key (zgk_test_) and, when agreed, a live key (zgk_live_).
2. Call GET /network on the test base URL to read the token contract and the confirmations required.
3. Create an invoice with POST /invoices and an Idempotency-Key derived from your order id.
4. Show the customer hosted_url, or deposit_address with the exact amount, or payment_uri (EIP-681) as a QR code.
5. Give zPayLink your HTTPS webhook URL; verify every ZGate-Signature and fulfil the order on invoice.paid after confirming with GET /invoices/{id}.
6. Pay a test invoice on BNB Smart Chain Testnet with the test token from GET /network, then repeat the flow with the live key on production within your limits.

5. Idempotency, retries and rate limits

POST /invoices accepts an optional Idempotency-Key header (1 to 128 characters: letters, digits and . _ : -), unique per merchant and kept for the life of the invoice. The same key with the same body returns 200 with the current invoice and Idempotent-Replayed: true; the same key with a different body returns 409 idempotency_conflict.

300 requests per minute per key and 60 invoice creations per minute per key; 600 requests per minute per IP before authentication and 30 failed authentications per minute per IP. Responses carry RateLimit-Limit and RateLimit-Remaining; a 429 carries Retry-After. The edge also limits each IP to 20 requests per second (that 429 is a plain page, not JSON).

- Retry network errors, 5xx and 429 with the same idempotency key and growing pauses; on 429 wait at least Retry-After seconds.
- Do not retry 4xx validation errors unchanged.

6. API reference

Method	Path	Endpoint	Scope
POST	`/invoices`	Create an invoice	`invoices:write`

Method	Path	Endpoint	Scope
GET	`/invoices/{id}`	Retrieve an invoice	`invoices:read`
GET	`/invoices`	List invoices	`invoices:read`
GET	`/transactions`	List transactions	`transactions:read`
GET	`/network`	Network status	any key
POST	`/webhooks/test`	Send a test webhook	`webhooks:write`
GET	`/events`	List events	`invoices:read`
POST	`/events/{id}/redeliver`	Redeliver an event	`webhooks:write`

Create an invoice: `POST /invoices`

Creates a USDT invoice with its own deposit address. Unknown body fields are rejected. Amounts are decimal strings (up to 18 decimals); the default range is 1 to 100000 USDT and merchant or launch limits can be lower.

Item	Value
Method and path	`POST /invoices`
Base URL	`https://gate.zpaylink.com/staging/api/v1`, `https://gate.zpaylink.com/api/v1`
Authentication	`Authorization: Bearer <key>`, scope `invoices:write`
Idempotency-Key header	optional
Success	`201 Created`

Body (JSON)

Name	Type	Required	Description
`merchant_order_id`	`string`	yes	Your order reference: 1 to 64 characters, letters, digits and . _ : -
`amount`	`string`	yes	Decimal string such as "25.00"; no sign or exponent
`asset`	`string`	yes	Must be "USDT"
`network`	`string`	yes	"bsc-testnet" on the test URL, "bsc" on production
`expires_in_minutes`	`integer`		5 to 1440; default set by zPayLink (30)
`description`	`string | null`		Up to 255 characters, no control characters
`metadata`	`object | null`		Your data, up to 4096 bytes as JSON and 4 levels deep

Example request

```
curl -X POST https://gate.zpaylink.com/staging/api/v1/invoices \
  -H "Authorization: Bearer $ZGATE_API_KEY" \
  -H "Idempotency-Key: $(uuidgen)" \
  -H "Content-Type: application/json" \
  -d '{"merchant_order_id":"ORDER-1001","amount":"25.00","asset":"USDT","network":"bsc-te
stnet","description":"Order 1001","metadata":{"customer_ref":"C-77"}}'
```

Example response (201 Created)

```
{
  "id": "inv_7k2m9qx4r8t3w6y5p2n4",
  "object": "invoice",
  "merchant_id": "mch_9w4k7m2x5q8r3t6y2p4n",
  "merchant_order_id": "ORDER-1001",
  "status": "pending",
  "late": false,
  "final": false,
  "network": "bsc-testnet",
  "chain_id": 97,
  "asset": "USDT",
  "token_contract": "0xTOKEN_CONTRACT",
  "token_decimals": 18,
  "amount": "25.00",
  "amount_raw": "25000000000000000000",
  "amount_received": "0.00",
  "amount_received_raw": "0",
  "amount_pending": "0.00",
  "amount_pending_raw": "0",
  "deposit_address": "0xDEPOSIT_ADDRESS",
  "required_confirmations": 15,
  "payment_uri": "ethereum:0xTOKEN_CONTRACT@97/transfer?address=0xDEPOSIT_ADDRESS&uint256
=25000000000000000000",
  "hosted_url": "https://gate.zpaylink.com/staging/pay/inv_7k2m9qx4r8t3w6y5p2n4",
  "description": "Order 1001",
  "metadata": { "customer_ref": "C-77" },
  "created_at": "2026-10-15T09:00:00.000Z",
  "expires_at": "2026-10-15T09:30:00.000Z",
  "updated_at": "2026-10-15T09:00:00.000Z",
  "detected_at": null,
  "finalized_at": null,
  "expired_at": null
}
```

Errors

- `422 invalid_request`
- `422 merchant_limit`
- `409 idempotency_conflict`
- `403 merchant_inactive`
- `503 payments_not_activated`
- `503 wallet_unavailable`
- `429 rate_limited`

Notes

- Show the customer hosted_url, or deposit_address with the exact amount, or payment_uri (EIP-681) as a QR code. Each invoice has its own address.
- Fetch token_contract and required_confirmations from GET /network instead of hard-coding them.

Retrieve an invoice: `GET /invoices/{id}`

Returns the invoice with its on-chain transactions (up to 100, including orphaned ones after a reorganisation). Use it to confirm a status before you fulfil an order.

Item	Value
Method and path	`GET /invoices/{id}`
Base URL	`https://gate.zpaylink.com/staging/api/v1`, `https://gate.zpaylink.com/api/v1`
Authentication	`Authorization: Bearer <key>`, scope `invoices:read`
Success	`200 OK`

Path parameters

Name	Type	Required	Description
`id`	`string`	yes	Invoice id (inv_…)

Example request

```
curl https://gate.zpaylink.com/staging/api/v1/invoices/inv_7k2m9qx4r8t3w6y5p2n4 \
  -H "Authorization: Bearer $ZGATE_API_KEY"
```

Example response (200 OK)


```
{
    "id": "inv_7k2m9qx4r8t3w6y5p2n4",
  "object": "invoice",
  "merchant_id": "mch_9w4k7m2x5q8r3t6y2p4n",
  "merchant_order_id": "ORDER-1001",
  "status": "pending",
  "late": false,
  "final": false,
  "network": "bsc-testnet",
  "chain_id": 97,
  "asset": "USDT",
  "token_contract": "0xTOKEN_CONTRACT",
  "token_decimals": 18,
  "amount": "25.00",
  "amount_raw": "25000000000000000000",
  "amount_received": "0.00",
  "amount_received_raw": "0",
  "amount_pending": "0.00",
  "amount_pending_raw": "0",
  "deposit_address": "0xDEPOSIT_ADDRESS",
  "required_confirmations": 15,
  "payment_uri": "ethereum:0xTOKEN_CONTRACT@97/transfer?address=0xDEPOSIT_ADDRESS&uint256
=250000000000000000",
  "hosted_url": "https://gate.zpaylink.com/staging/pay/inv_7k2m9qx4r8t3w6y5p2n4",
  "description": "Order 1001",
  "metadata": { "customer_ref": "C-77" },
  "created_at": "2026-10-15T09:00:00.000Z",
  "expires_at": "2026-10-15T09:30:00.000Z",
  "updated_at": "2026-10-15T09:00:00.000Z",
  "detected_at": null,
  "finalized_at": null,
  "expired_at": null,
  "transactions": [
    {
      "id": "trf_3n8p2q5r9t4w7y2k6m3x",
      "object": "transaction",
      "invoice_id": "inv_7k2m9qx4r8t3w6y5p2n4",
      "network": "bsc-testnet",
      "tx_hash": "0xTRANSACTION_HASH",
      "log_index": 4,
      "block_number": 51234567,
      "block_hash": "0xBLOCK_HASH",
      "token_contract": "0xTOKEN_CONTRACT",
```

```
      "from": "0xPAYER_ADDRESS",
      "to": "0xDEPOSIT_ADDRESS",
      "amount": "25.00",
      "amount_raw": "25000000000000000000",
      "status": "confirmed",
      "confirmations": 15,
      "after_expiry": false,
      "detected_at": "2026-10-15T09:04:10.000Z",
      "confirmed_at": "2026-10-15T09:05:00.000Z",
      "orphaned_at": null,
      "explorer_url": "https://testnet.bscscan.com/tx/0xTRANSACTION_HASH"
    }
  ]
}
```


Errors

- `404 not_found`
- `401 unauthorized`

List invoices: `GET /invoices`

Newest first, with cursor pagination: pass the last id of a page as starting_after to get the next one.

Item	Value
Method and path	`GET /invoices`
Base URL	`https://gate.zpaylink.com/staging/api/v1`, `https://gate.zpaylink.com/api/v1`
Authentication	`Authorization: Bearer <key>`, scope `invoices:read`
Success	`200 OK`

Query parameters

Name	Type	Required	Description
`status`	`string`		pending, detected, confirming, paid, underpaid, overpaid, expired or failed
`merchant_order_id`	`string`		Exact match
`limit`	`integer`		1 to 100, default 20
`starting_after`	`string`		An invoice id

Example request

```
curl https://gate.zpaylink.com/staging/api/v1/invoices \
  -H "Authorization: Bearer $ZGATE_API_KEY"
```

Example response (200 OK)

```
{
  "object": "list",
  "data": [
    { "id": "inv_7k2m9qx4r8t3w6y5p2n4", "object": "invoice", "status": "paid", "amount":
"25.00" }
  ],
  "has_more": false
}
```

Errors

- `422 invalid_request`

Notes

- List items are full invoice objects; the example is shortened.

List transactions: `GET /transactions`

On-chain transfers to your deposit addresses, newest detected first. Status is seen, confirmed or orphaned.

Item	Value
Method and path	`GET /transactions`
Base URL	`https://gate.zpaylink.com/staging/api/v1`, `https://gate.zpaylink.com/api/v1`
Authentication	`Authorization: Bearer <key>`, scope `transactions:read`
Success	`200 OK`

Query parameters

Name	Type	Required	Description
`invoice_id`	`string`		Only this invoice
`limit`	`integer`		1 to 100, default 20
`starting_after`	`string`		A transaction id (trf_…)

Example request

```
curl https://gate.zpaylink.com/staging/api/v1/transactions \
  -H "Authorization: Bearer $ZGATE_API_KEY"
```

Example response (200 OK)


```
{
  "object": "list",
  "data": [
    {
      "id": "trf_3n8p2q5r9t4w7y2k6m3x",
      "object": "transaction",
      "invoice_id": "inv_7k2m9qx4r8t3w6y5p2n4",
      "network": "bsc-testnet",
      "tx_hash": "0xTRANSACTION_HASH",
      "log_index": 4,
      "block_number": 51234567,
      "block_hash": "0xBLOCK_HASH",
      "token_contract": "0xTOKEN_CONTRACT",
      "from": "0xPAYER_ADDRESS",
      "to": "0xDEPOSIT_ADDRESS",
      "amount": "25.00",
      "amount_raw": "25000000000000000000",
      "status": "confirmed",
      "confirmations": 15,
      "after_expiry": false,
      "detected_at": "2026-10-15T09:04:10.000Z",
      "confirmed_at": "2026-10-15T09:05:00.000Z",
      "orphaned_at": null,
      "explorer_url": "https://testnet.bscscan.com/tx/0xTRANSACTION_HASH"
    }
  ],
  "has_more": false
}
```


Errors

- `422 invalid_request`

Network status: `GET /network`

Token contract, confirmations required and the health of the chain connection (ok, degraded, down, misconfigured or not_started). Works with any valid key.

Item	Value
Method and path	`GET /network`
Base URL	`https://gate.zpaylink.com/staging/api/v1`, `https://gate.zpaylink.com/api/v1`
Authentication	`Authorization: Bearer <key>`, any valid key

Item	Value
Success	`200 OK`

Example request

```
curl https://gate.zpaylink.com/staging/api/v1/network \
  -H "Authorization: Bearer $ZGATE_API_KEY"
```

Example response (200 OK)

```
{
  "network": "bsc-testnet",
  "chain_id": 97,
  "asset": "USDT",
  "token_contract": "0xTOKEN_CONTRACT",
  "required_confirmations": 15,
  "status": "ok"
}
```

Send a test webhook: `POST /webhooks/test`

Queues a signed webhook.test event to every active endpoint of your account (10 per minute per key).

Item	Value
Method and path	`POST /webhooks/test`
Base URL	`https://gate.zpaylink.com/staging/api/v1`, `https://gate.zpaylink.com/api/v1`
Authentication	`Authorization: Bearer <key>`, scope `webhooks:write`
Success	`202 Accepted`

Example request

```
curl -X POST https://gate.zpaylink.com/staging/api/v1/webhooks/test \
  -H "Authorization: Bearer $ZGATE_API_KEY"
```

Example response (202 Accepted)

```
{
  "object": "event",
  "id": "evt_5q2w8e4r7t3y6u9p2k4m",
  "type": "webhook.test",
  "deliveries_queued": 1
}
```

Errors

- `403 insufficient_scope`
- `429 rate_limited`

List events: `GET /events`

The webhook events of your account with the exact signed payload and the state of every delivery. Use it to reconcile after downtime.

Item	Value
Method and path	`GET /events`
Base URL	`https://gate.zpaylink.com/staging/api/v1` , `https://gate.zpaylink.com/api/v1`
Authentication	`Authorization: Bearer <key>` , scope `invoices:read`
Success	`200 OK`

Query parameters

Name	Type	Required	Description
`invoice_id`	`string`		Only events of this invoice
`type`	`string`		An event type such as invoice.paid
`limit`	`integer`		1 to 100, default 20
`starting_after`	`string`		An event id

Example request

```
curl https://gate.zpaylink.com/staging/api/v1/events \
  -H "Authorization: Bearer $ZGATE_API_KEY"
```

Example response (200 OK)


```
{
  "object": "list",
  "data": [
    {
      "id": "evt_8m3k6p2q9r4t7w2y5x3n",
      "object": "event",
      "type": "invoice.paid",
      "invoice_id": "inv_7k2m9qx4r8t3w6y5p2n4",
      "created_at": "2026-10-15T09:05:01.000Z",
      "payload": { "id": "evt_8m3k6p2q9r4t7w2y5x3n", "object": "event", "type": "invoice.
paid" },
      "deliveries": [
        { "url": "https://shop.example.com/webhooks/zgate", "status": "delivered", "attem
pts": 1, "last_http_status": 200, "last_error": null, "last_attempt_at": "2026-10-15T09:0
5:02.000Z", "delivered_at": "2026-10-15T09:05:02.000Z", "next_attempt_at": null }
      ]
    }
  ],
  "has_more": false
}
```


Notes

- payload is the full signed body; the example is shortened.

Redeliver an event: `POST /events/{id}/redeliver`

Sends an event again now to your active endpoints, with the same event id (10 per minute per key). The attempt counter is not reset.

Item	Value
Method and path	`POST /events/{id}/redeliver`
Base URL	`https://gate.zpaylink.com/staging/api/v1`, `https://gate.zpaylink.com/api/v1`
Authentication	`Authorization: Bearer <key>`, scope `webhooks:write`
Success	`202 Accepted`

Path parameters

Name	Type	Required	Description
`id`	`string`	yes	Event id (evt_…)

Example request

```
curl -X POST https://gate.zpaylink.com/staging/api/v1/events/evt_8m3k6p2q9r4t7w2y5x3n/redeliver \
  -H "Authorization: Bearer $ZGATE_API_KEY"
```

Example response (202 Accepted)

```
{
  "object": "event",
  "id": "evt_8m3k6p2q9r4t7w2y5x3n",
  "type": "invoice.paid",
  "deliveries_queued": 1
}
```

Errors

- `404 not_found`
- `429 rate_limited`

7. Webhooks

> A status can move on after an event: an underpaid invoice can be topped up, and a late transfer to an expired invoice (within 30 days) sets late: true and moves it to paid. Each change sends a new event with a new id.

Event	When
`invoice.created`	An invoice was created
`payment.detected`	A transfer to the deposit address was seen on chain
`payment.confirming`	The transfer has started collecting confirmations
`invoice.paid`	The full amount is confirmed (fulfil the order here)
`invoice.underpaid`	Less than the amount arrived; a top-up before expiry can still complete it
`invoice.overpaid`	More than the amount arrived
`invoice.expired`	The invoice expired without full payment
`webhook.test`	Sent by POST /webhooks/test

Signature header: `ZGate-Signature: t=<unix seconds>,v1=<hex HMAC-SHA256(secret, "<t>.<raw body>")>`. Return any 2xx within 10 seconds. Redirects are not followed and count as failures. Up to 12 attempts. After a failure the next attempt waits about 1, 2, 4, 8, 16, 32 and 64 minutes, then about 2, 4 and 6 hours (with 10% jitter), for about 20 hours in total. Delivery order is not guaranteed. zPayLink registers

your HTTPS endpoint (port 443 or 8443, public address) and gives you its whsec_ secret once; every endpoint receives all event types. Up to 5 endpoints per merchant.

Full details and verification code: webhooks.md.

8. Error handling

Use the machine-readable code, not the message.

```
{
  "error": {
    "code": "invalid_request",
    "message": "Invalid invoice request.",
    "details": { "amount": "must be a decimal string, e.g. \"25.50\"" }
  }
}
```

All codes: error-codes.md.

9. Important integration notes

- Amounts are decimal strings with up to 18 decimals; compare amounts as strings or exact decimals, never as floats.
- An invoice can change after a webhook: an underpaid invoice can be topped up, and a late transfer within 30 days of expiry marks it late: true and moves it on. final: true on an expired or underpaid invoice is therefore not absolute.
- Webhook delivery order is not guaranteed; deduplicate on ZGate-Event-Id and rely on the invoice status from the API.

10. Security

- Keep API keys and webhook secrets on your server, in a secret store or environment variables; never in front-end code, mobile apps or source control.
- Use separate keys per environment and per system, with the narrowest scopes you need (zGate read-only keys, zCheck verifications:submit only where you upload server to server).
- Rotate keys when people or systems change: zCheck and zPay keep the old key valid for a grace period during rotation; revoke at once if a key may be exposed.
- Verify every webhook signature with a constant-time comparison and a 5 minute timestamp window, and deduplicate on the event id.
- Webhook endpoints must be HTTPS on a public address; zPayLink refuses private and internal addresses and does not follow redirects.

- Treat webhooks as notifications: confirm amounts and statuses with the API before you credit a user or ship an order.
- zCheck never returns images or full national IDs through the API or webhooks; keep the personal data you collect to the minimum.

11. Troubleshooting

Symptom	What to do
401 unauthorized / Invalid API key	Send exactly "Authorization: Bearer <key>"; check you use a test key on the test URL and a live key on production.
403 insufficient_scope	The key lacks the scope listed for the endpoint; ask for a key with that scope.
422 validation_error / invalid_request	Fix the request: details lists the fields. Unknown body fields are rejected.
409 idempotency_conflict	The Idempotency-Key was used with a different request; use a new key for a new operation.
429 rate_limited	Wait at least Retry-After seconds, then retry with the same Idempotency-Key.
Webhook signature does not verify	Verify the raw body bytes (before JSON parsing), use the secret of that endpoint and keep your server clock in sync (5 minute window).
Connection timeouts from your servers	Some international networks cannot reach zPayLink servers at the moment. Test from another network and contact zPayLink with your server's public IP and a traceroute.