zPay integration guide

zPayLink developer kit, 2026-10-16. Everything below is generated from the verified zPayLink developer documentation; examples use fictional values.

1. Product overview

Rial payments by card-to-card and SHEBA transfer: create a payment request with the card or SHEBA your user pays from, show them the destination account, and zPay matches the incoming deposit to the request, values it in USDT at a locked rate and reports it with a signed deposit.matched webhook. One environment: every new key is a standard key.

> The zPay API v1 is open to merchants that zPayLink onboards: zPayLink adds your server's public IP address to the zPay access list, and requests from other addresses are refused with HTTP 401 before they reach the API. Live deposit matching and webhook delivery are switched off in production for now: payment requests made with a standard key stay pending, and webhooks are recorded and held, until zPayLink switches them on. Existing ZPay 1.0 integrations keep working unchanged.

Aspect	Status (2026-10-16)
Implemented	yes (zPay 2.0 API v1)
Reachable from the internet	from server IPs on the zPay access list (added by zPayLink during onboarding); other addresses get HTTP 401
Testing	one environment: no new test keys; legacy zpm_test_ keys keep working with simulated deposits
Production	standard keys (zpm_live_) create live payment requests; live deposit matching and webhook delivery are switched off until zPayLink switches them on
Disabled or restricted	live matching and webhook sending (requests stay pending, webhooks are held)

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

- An organization admin creates keys in the zPay dashboard on the merchant's page (Merchants, the merchant, API keys, Create key). Every new key is a standard key (zpm_live_), shown once and stored only as a SHA-256 hash. Optional IP allowlist (up to 20 addresses or CIDR ranges) and expiry. Rotation issues a new key and keeps the old one valid for a grace period (24 hours by default, 0 to 168). A key works only while its merchant is active. Scopes: payment_requests:write, payment_requests:read, rates:read.
- 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: `zpm_live_… (legacy zpm_test_…)`.

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

Environment	Base URL
One environment; your server IP must be on the zPay access list	`https://pay.zpaylink.com/api/v1`

4. Test and live keys

Key	Issued	Requests	Matching
`zpm_live_`	Every new key (standard key)	mode: "live"; a standard key sees only live requests	Against real deposits on your organization's bank account, once zPayLink switches live matching on (off today: requests stay pending)
`zpm_test_`	No longer issued; keys created earlier keep working	mode: "test"; a test key sees only test requests	Never against real deposits; POST /sandbox/deposits simulates one (deprecated)

5. Step-by-step integration

1. Ask zPayLink to onboard you: zPayLink sets up your organization and its bank account and adds your server's public IP address to the zPay access list (other addresses get HTTP 401 before they reach the API).
2. An organization admin creates and activates the merchant in the zPay dashboard, sets the HTTPS callback URL and stores the 64-character webhook signing secret (shown once).
3. Create a standard key (zpm_live_) on the merchant's page, ideally with an IP allowlist, and keep it in a secret store.
4. Create a payment request with POST /payment-requests (source = the card, SHEBA or account the user pays from) and an Idempotency-Key; show the returned destination for payer_window_seconds.
5. Verify X-ZPay-Signature-V2 on deposit.matched, deduplicate on event_id (X-ZPay-Event-Id), confirm with GET /payment-requests/{ref} and credit your user once.
6. Live deposit matching and webhook delivery are switched off in production until zPayLink switches them on: until then requests stay pending and webhooks are held.

6. Lifecycle and statuses

Status	Meaning
`pending`	Waiting for a deposit from the declared source. Show the destination for payer_window_seconds (1 hour by default); the request stays open until expires_at (7 days by default, set by the organization). A live request does not expire while the bank collector is catching up, for at most 7 more days.
`matched` (final)	A deposit from the declared source was credited: received amount, USDT values, the locked rate and the sender are set and never change. deposit.matched is sent.
`manual_review`	A person decides: a deposit of exactly the suggested amount came from another source, the amount or bank time is unknown, the deposit is dated before the request, it may be a duplicate, or no usable settlement rate exists. It ends as matched or rejected, or goes back to pending.
`expired` (final)	No matching deposit arrived before expires_at.
`rejected` (final)	Closed by a reviewer without a credit.
`cancelled` (final)	Reserved status (accepted by the status filter); no API call or dashboard action sets it today.

Allowed transitions, enforced by the database: pending to matched, manual_review, expired or cancelled; manual_review to matched, rejected or pending. A request is credited at most once.

7. Idempotency, retries and rate limits

POST /payment-requests accepts an optional Idempotency-Key header (8 to 100 characters: letters, digits and _ - : .), scoped to the merchant and the key's mode. The same key with the same source, client_id and amount_rial returns the original request with 200 and Idempotent-Replayed: true; a different request returns 409 idempotency_conflict. Without the header every call creates a new request.

300 requests per minute per key, in a fixed one-minute window (429 rate_limited with Retry-After: 60). Every response carries an X-Request-Id header; errors repeat it as request_id.

- 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.

8. API reference

Method	Path	Endpoint	Scope
POST	`/payment-requests`	Create a payment request	`payment_requests:write`
GET	`/payment-requests`	List payment requests	`payment_requests:read`
GET	`/payment-requests/{ref}`	Retrieve a payment request	`payment_requests:read`
GET	`/rate`	USDT settlement rate	`rates:read`
POST	`/sandbox/deposits`	Legacy test keys: simulate a deposit	`payment_requests:write`

Create a payment request: `POST /payment-requests`

Registers the card, SHEBA or account your user will pay from and returns the destination account to show them. A replay with the same Idempotency-Key returns 200.

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

Body (JSON)

Name	Type	Required	Description
`source`	`string`	yes	The payer's card number, SHEBA or account number, 1 to 64 characters. Text containing IR or 22 or more digits is a SHEBA (last 24 digits), 16 or more digits a card (first 16), anything else an account number
`client_id`	`string`		Your user or order reference, up to 128 characters (returned as client_id and merchant_reference)
`amount_rial`	`integer`		Expected amount in rial, a positive integer (optional; the received amount is what counts)

Example request

```
curl -X POST https://pay.zpaylink.com/api/v1/payment-requests \
  -H "Authorization: Bearer $ZPAY_API_KEY" \
  -H "Idempotency-Key: $(uuidgen)" \
  -H "Content-Type: application/json" \
  -d '{"source":"6037990000000000","client_id":"user-2041","amount_rial":25000000}'
```

Example response (201 Created)


```
{
  "id": "ZX-R-8F3A2C91",
  "object": "payment_request",
  "mode": "live",
  "status": "pending",
  "client_id": "user-2041",
  "source": { "type": "card", "masked": "•••• 0000" },
  "suggested_amount_rial": 25000000,
  "received_amount_rial": null,
  "usdt_amount": null,
  "rate": null,
  "sender": null,
  "created_at": "2026-10-16T09:00:00.000Z",
  "expires_at": "2026-10-23T09:00:00.000Z",
  "matched_at": null,
  "merchant_reference": "user-2041",
  "currency": "IRR",
  "requested_amount_irr": 25000000,
  "paid_amount_irr": null,
  "difference_irr": null,
  "equivalent_usdt": null,
  "credited_usdt": null,
  "exchange_rate": null,
  "bank_transaction_id": null,
  "destination": { "holder": "Example Co", "card": "6219860000000000", "sheba": "IR000000
00000000000000000000", "account": "0000000000", "bank": "Blu" },
  "payer_window_seconds": 3600
}
```


Errors

- `401 unauthorized`
- `403 merchant_inactive`
- `403 ip_not_allowed`
- `409 idempotency_conflict`
- `422 validation_error`
- `429 rate_limited`

Notes

- Unknown body fields are rejected with 422. A source that contains no digits, or more than 40, is refused with 422 ("Could not read the source card/SHEBA.").

- A request is matched only to a deposit from the declared source that arrives after the request (with a 10 minute clock tolerance by default); the received amount is credited even if it differs from amount_rial. A deposit of exactly the suggested amount from another source goes to manual review.
- The destination is returned only by this call: keep it with the request on your side.

List payment requests: `GET /payment-requests`

The merchant's requests in the key's mode, newest first by default, with cursor pagination (next_cursor).

Item	Value
Method and path	`GET /payment-requests`
Base URL	`https://pay.zpaylink.com/api/v1`
Authentication	`Authorization: Bearer <key>`, scope `payment_requests:read`
Success	`200 OK`

Query parameters

Name	Type	Required	Description
`status`	`string`		pending, matched, manual_review, expired, rejected or cancelled
`created_from / created_to`	`ISO date-time`		Range on created_at; created_to is exclusive
`order`	`string`		desc (default) or asc
`limit`	`integer`		1 to 100, default 20
`cursor`	`integer`		next_cursor of the previous page

Example request

```
curl https://pay.zpaylink.com/api/v1/payment-requests \
  -H "Authorization: Bearer $ZPAY_API_KEY"
```

Example response (200 OK)

```
{
  "object": "list",
  "data": [
    { "id": "ZX-R-8F3A2C91", "object": "payment_request", "mode": "live", "status": "pend
ing" }
  ],
  "has_more": true,
  "next_cursor": 57
}
```

Errors

- `401 unauthorized`
- `422 validation_error`

Notes

- Each item is a full payment request object (shortened here). Unknown query parameters are rejected with 422.

Retrieve a payment request: `GET /payment-requests/{ref}`

The current status of one request. Once matched it carries the received amount, the USDT values at the locked rate and the sender name with the last 4 card digits.

Item	Value
Method and path	`GET /payment-requests/{ref}`
Base URL	`https://pay.zpaylink.com/api/v1`
Authentication	`Authorization: Bearer <key>`, scope `payment_requests:read`
Success	`200 OK`

Path parameters

Name	Type	Required	Description
`ref`	`string`	yes	Request id (ZX-R-…)

Example request

```
curl https://pay.zpaylink.com/api/v1/payment-requests/ZX-R-8F3A2C91 \
  -H "Authorization: Bearer $ZPAY_API_KEY"
```

Example response (200 OK)


```
{
  "id": "ZX-R-8F3A2C91",
  "object": "payment_request",
  "mode": "live",
  "status": "matched",
  "client_id": "user-2041",
  "source": { "type": "card", "masked": "•••• 0000" },
  "suggested_amount_rial": 25000000,
  "received_amount_rial": 25000000,
  "usdt_amount": "23.75",
  "rate": { "toman_per_usdt": "105250.00000000", "source": "manual" },
  "sender": { "name": "Sara Rezaei", "card_last4": "0000" },
  "created_at": "2026-10-16T09:00:00.000Z",
  "expires_at": "2026-10-23T09:00:00.000Z",
  "matched_at": "2026-10-16T09:20:00.000Z",
  "merchant_reference": "user-2041",
  "currency": "IRR",
  "requested_amount_irr": 25000000,
  "paid_amount_irr": 25000000,
  "difference_irr": 0,
  "equivalent_usdt": "23.752969",
  "credited_usdt": "23.75",
  "exchange_rate": { "source": "manual", "usdt_price_toman": "105250", "usdt_price_irr":
"1052500", "captured_at": "2026-10-16T09:20:00+00:00" },
  "bank_transaction_id": 1042
}
```


Errors

- `401 unauthorized`
- `404 not_found`

Notes

- 404 not_found also means a request of another merchant or of the other mode (a test key never sees live requests and the reverse).

USDT settlement rate: `GET /rate`

The toman per USDT rate zPay would use now to value a matched deposit: the organization's manual rate, or the latest quote. A matched request keeps the rate locked when it was matched.

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

Example request

```
curl https://pay.zpaylink.com/api/v1/rate \
  -H "Authorization: Bearer $ZPAY_API_KEY"
```

Example response (200 OK)

```
{
  "object": "rate",
  "pair": "USDT/IRT",
  "toman_per_usdt": "105250",
  "source": "manual",
  "as_of": "2026-10-16T09:00:00.000Z",
  "stale": false
}
```

Errors

- `401 unauthorized`

Notes

- When no settlement rate is available, toman_per_usdt is null, source is "bitpin" and stale is true; a live deposit is then never credited automatically (manual review).

Legacy test keys: simulate a deposit: `POST /sandbox/deposits`

Deprecated; works only with a legacy test key (zpm_test_). Records a simulated deposit on test data and runs test matching, so a test request can reach matched. A standard key gets 403 live_key.

Item	Value
Method and path	`POST /sandbox/deposits`
Base URL	`https://pay.zpaylink.com/api/v1`
Authentication	`Authorization: Bearer <key>`, scope `payment_requests:write`
Success	`201 Created`

Body (JSON)

Name	Type	Required	Description
`source`	`string`	yes	The same source as the test request
`amount_rial`	`integer`	yes	Deposit amount in rial
`sender_name`	`string`		Up to 120 characters

Example request

```
curl -X POST https://pay.zpaylink.com/api/v1/sandbox/deposits \
  -H "Authorization: Bearer $ZPAY_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"source":"6037990000000000","amount_rial":25000000,"sender_name":"Sara Rezaei"}'
```

Example response (201 Created)

```
{
  "object": "sandbox_deposit",
  "bank_transaction_id": 1042,
  "matcher": { "expired": 0, "matched": 1, "review": 0 }
}
```

Errors

- `403 live_key`
- `409 no_bank_account`
- `422 validation_error`

Payment request object

Returned by every payment request endpoint. Money values are exact: rial as integers, USDT and rates as decimal strings. The create response adds destination and payer_window_seconds.

Field	Type	Description
`id`	`string`	Request id, ZX-R- and 8 hex characters
`object`	`string`	Always payment_request
`mode`	`string`	live (standard key) or test (legacy test key)
`status`	`string`	See Lifecycle and statuses
`client_id`	`string`	What you sent; an empty string when you sent none
`source.type / source.masked`	`string`	card, sheba or account; masked as •••• and the last 4 digits, or IR… and the last 6 for a SHEBA

Field	Type	Description
`suggested_amount_rial`	`integer | null`	amount_rial you sent
`received_amount_rial`	`integer | null`	What actually arrived (credited even if it differs)
`usdt_amount`	`string | null`	Credited USDT, 2 places: received rial / 10 / rate, half up
`rate`	`object | null`	toman_per_usdt (decimal string, up to 8 places) and source (manual or bitpin) of the locked rate
`sender`	`object | null`	When matched: name and card_last4 of the payer as the bank reported them
`created_at / expires_at / matched_at`	`ISO date-time`	UTC; matched_at is null until matched
`merchant_reference`	`string`	Same as client_id
`currency`	`string`	Always IRR
`requested_amount_irr / paid_amount_irr`	`integer | null`	Requested (amount_rial) and paid amount in rial
`difference_irr`	`integer | null`	paid minus requested (negative = underpaid); null without a requested amount
`equivalent_usdt`	`string | null`	paid / 10 / rate with 6 places, half up (informational)
`credited_usdt`	`string | null`	The credited value, 2 places (same number as usdt_amount)
`exchange_rate`	`object | null`	source, usdt_price_toman, usdt_price_irr and captured_at of the rate locked at match time
`bank_transaction_id`	`integer | null`	zPay id of the matched bank deposit
`destination`	`object`	Create response only: holder, card, sheba, account and bank of the account to pay to
`payer_window_seconds`	`integer`	Create response only: how long to show the destination (organization setting, 3600 by default)

deposit.matched body

The ZPay 1.0 keys come first, unchanged, followed by the zPay 2.0 keys. The body is built once and every retry sends the same bytes.

Field	Type	Description
`event / status`	`string`	deposit.matched / matched
`request_ref`	`string`	The payment request id (ZX-R-…)
`client_id`	`string | null`	Your reference

Field	Type	Description
`source_card`	`string`	The full declared source digits (card, SHEBA or account)
`sender_name / sender_card`	`string | null`	Payer name and full card number as the bank reported them
`amount_rial / amount_toman`	`integer`	Received amount (toman = rial / 10, rounded down)
`amount_usdt`	`number | null`	Legacy JSON number of the credited USDT; read credited_usdt instead
`rate_used`	`integer | null`	Legacy rounded toman per USDT
`matched_at`	`string`	UTC, for example 2026-10-16T09:20:00+00:00
`event_id / idempotency_key`	`string`	The event id (evt_…), same as X-ZPay-Event-Id and stable across retries: deduplicate on it
`merchant_reference … bank_transaction_id`	`mixed`	The same dual-currency fields as the payment request object
`bank_reference`	`string | null`	The bank's reference number when it was captured

ZPay 1.0 compatible endpoints

Frozen routes for integrations built on ZPay 1.0, with the X-ZPay-Key header and the ZPay 1.0 request and response shapes. They are opened per integration on request; build new integrations on /api/v1.

Method	Path	Endpoint
POST	`/wp-json/zpay/v1/gateway/requests`	Create a payment request (ZPay 1.0 body)
GET	`/wp-json/zpay/v1/gateway/requests`	The merchant's 20 latest requests
GET	`/wp-json/zpay/v1/gateway/requests/{ref}`	One request
GET	`/wp-json/zpay/v1/gateway/rate`	Reference rate

9. Webhooks

> Webhook delivery is switched off in production for now: events are recorded and held, and zPayLink releases them explicitly once sending is switched on. Verify X-ZPay-Signature-V2, which includes the timestamp; X-ZPay-Signature (HMAC of the raw body only) remains for ZPay 1.0 receivers. The body is PHP-compatible JSON ("/" is sent as "\/", non-ASCII as \uXXXX): verify the raw bytes, never a re-serialised object.

Event	When
`deposit.matched`	A deposit was matched to a payment request (automatically or after manual review)

Signature header: `X-ZPay-Signature-V2: t=<unix seconds>,v1=<hex HMAC-SHA256(secret, "<t>.<raw body>")>`. Return any 2xx within 12 seconds. Redirects are not followed and count as failures; the

response body is ignored. Up to 11 attempts: after a failure the next waits 1, 2, 5, 10 and 30 minutes, then 1, 2, 6, 12 and 24 hours (about 46 hours in total), then the delivery stops (dead letter). Delivery is at least once: after a crash the same event can arrive again. Set the merchant's callback URL in the zPay dashboard: HTTPS, a public address, no credentials in the URL, up to 500 characters. The 64-character hex signing secret is shown once when the merchant is created and when it is rotated; pending deliveries are signed with the new secret. A merchant without a callback URL receives no webhooks.

Full details and verification code: webhooks.md.

10. Error handling

Use the machine-readable code, not the message.

```
{
  "error": {
    "code": "validation_error",
    "message": "source: Required",
    "details": [{ "field": "source", "message": "Required" }]
  },
  "request_id": "req_a3f1c9e25b7d4e8a9c1f"
}
```

All codes: error-codes.md.

11. Important integration notes

- One environment: every new key is a standard key (zpm_live_). Legacy test keys (zpm_test_) keep working; their requests have mode "test", are never matched against real deposits, and only they can use the deprecated POST /sandbox/deposits.
- The webhook body is PHP-compatible JSON: "/" is escaped as "\/" and non-ASCII characters as \uXXXX. Verify the signature over the raw bytes before parsing; never re-serialise.
- The deposit.matched payload has no mode field: test and live events go to the same callback URL. Look the request up with the key of the matching mode (GET /payment-requests/{ref}) to know which one it is.
- source_card in the webhook is the full declared source (card, SHEBA or account digits); sender_card is the payer card from the bank. Store and display them as card data.
- A request matches only a deposit from its declared source; the received amount is what counts and may differ from amount_rial (see difference_irr). A deposit of exactly the suggested amount from another source goes to manual review.
- Money values are exact: rial as integers, USDT and rates as decimal strings (amount_usdt in the webhook is a legacy JSON number: read credited_usdt or equivalent_usdt). The rate is locked when the deposit is matched and never changes.
- The destination account is returned only by POST /payment-requests; keep it with the request.

12. 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.
- Give each key an IP allowlist with your servers' addresses; a call from another address is refused with 403 ip_not_allowed even with a valid key.
- Credit a user only once per event_id and once per request_ref, inside one database transaction, and answer 2xx only after it is committed.
- The webhook body contains the full payer card and source digits: store and display them as card data.

13. Troubleshooting

Symptom	What to do
401 with WWW-Authenticate: Basic realm="zPayLink staging"	zPay: your server's IP address is not on the zPay access list. It is not your API key; send zPayLink the public IP of the server that calls the API.
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.