Nexa Connect API

Integration Guide for Server-Side SDKs

Version 2.4.0 | Revised March 2026 | api.nexa.dev

Overview

Nexa Connect is the unified API for payments, identity verification, and ledger reconciliation used by finance teams building on the Nexa platform. This guide covers authentication, the core REST endpoints, official SDKs for Python and Node.js, webhook signature verification, and error-handling conventions. It is written for backend engineers integrating Nexa Connect into an existing service — prior familiarity with REST and JSON is assumed. All requests are made over HTTPS to the single base URL `https://api.nexa.dev/v2`; there is no per-feature subdomain. Responses are JSON, timestamps are ISO 8601 in UTC, and monetary amounts are always integer minor units (e.g. cents) to avoid floating-point rounding errors.

Median Latency	Auth Model	Rate Limit
118 ms	HMAC + Key	600 / min
p50, create-payment	signed webhooks	per API key

1 Authentication

Every request must carry a bearer API key issued from the Nexa Dashboard under **Settings → API Keys**. Keys are environment-scoped: a `test_` key never touches production data, and a `live_` key cannot be used against the sandbox host.

Authenticated request

```
curl https://api.nexa.dev/v2/payments \
  -H "Authorization: Bearer live_9k2Lp7XwQrT4vB1nZcHf" \
  -H "Content-Type: application/json"
```

Caution

Do not embed `live_` keys in client-side or mobile code. Keys are shown once at creation time; a lost key must be revoked and reissued, since Nexa does not store the plaintext value after generation.

1.1 Key rotation

- Rotate keys at least every 180 days; old and new keys overlap for 24 hours.
- Revoking a key invalidates it within 30 seconds across all edge regions.

2 Endpoint Reference

The table below covers the core resources exposed under `/v2`. Full parameter schemas for each endpoint are listed in the OpenAPI document linked from the developer dashboard.

Method	Path	Description	Auth
GET	`/payments`	List payments, filterable by status and date range.	🔒
POST	`/payments`	Create a payment intent for a customer or card token.	🔒
GET	`/payments/{id}`	Retrieve a single payment by its identifier.	🔒
PUT	`/payments/{id}`	Update metadata or capture an authorized payment.	🔒
DELETE	`/payments/{id}`	Cancel a payment that has not yet settled.	🔒
GET	`/customers`	List customers with pagination cursors.	🔒

Method	Path	Description	Auth
POST	/customers	Create a customer record with contact and tax details.	🔒
POST	/webhooks/endpoints	Register a URL to receive signed event callbacks.	🔒
GET	/ledger/entries	Fetch reconciliation entries for a settlement period.	🔒

Note

All list endpoints are cursor-paginated via `?after={cursor}` and return at most 100 records per page. Use the `has_more` field in the response envelope to decide whether to continue paging — do not infer completion from an empty-looking page size.

3 SDK Quickstart

Official SDKs are published for Python and Node.js and wrap retry logic, idempotency-key generation, and typed response models. Both snippets below perform the same action: creating a $42.50 payment for an existing customer.

Python – nexa-connect

```python
from nexa_connect import Client

client = Client(api_key="live_9k2Lp7XwQrT4vB1nZcHf
    ")

payment = client.payments.create(
    amount=4250,
    currency="usd",
    customer="cus_82nF3kLp",
    description="Invoice 10432",
    idempotency_key="inv-10432-charge",
)

print(payment.id, payment.status)
```

Node.js – @nexa/connect

```javascript
import { NexaConnect } from "@nexa/connect";

const client = new NexaConnect({
  apiKey: "live_9k2Lp7XwQrT4vB1nZcHf",
});

const payment = await client.payments.create({
  amount: 4250,
  currency: "usd",
  customer: "cus_82nF3kLp",
  description: "Invoice 10432",
  idempotencyKey: "inv-10432-charge",
});

console.log(payment.id, payment.status);
```

Caution

Always pass an explicit `idempotency_key` on payment creation. Without one, a retried request after a network timeout can create a duplicate charge; both SDKs surface this parameter directly for that reason.

4 Response Schema

A successful `POST /payments` call returns a `payment` object. Field names and types are stable across API versions; new fields may be added but existing ones are never removed or repurposed within a major version.

Field	Type	Description
id	string	Unique ID, prefixed pay_.
amount	integer	Minor units (cents).
currency	string	ISO 4217, lowercase.
status	enum	See status values below.
customer	string	Customer identifier.
created_at	string	ISO 8601 UTC timestamp.

Example response

```
{
  "id": "pay_5xQ9mKz1TdRp",
  "amount": 4250,
  "currency": "usd",
  "status": "requires_capture",
  "customer": "cus_82nF3kLp",
  "created_at": "2026-03-11T09:14:02Z"
}
```

4.1 Status values

- `requires_capture` — authorized, awaiting the merchant to capture funds.

- `succeeded` — funds captured and settlement scheduled.
- `failed` — the issuer or ledger declined the transaction.
- `canceled` — voided before capture; no funds were moved.

5 Error Handling

Nexa Connect uses conventional HTTP status codes: `2xx` for success, `4xx` for client-side request errors, and `5xx` for platform-side failures. Every error body includes a machine-readable `code` in addition to the HTTP status, since a single status like `402` can map to several distinct `code` values.

HTTP	code	Meaning
400	`invalid_request`	A required field was missing or malformed.
401	`invalid_api_key`	The bearer key is missing, revoked, or from the wrong environment.
402	`card_declined`	The issuer declined authorization; safe to retry with a different instrument.
402	`insufficient_funds`	Declined specifically for balance; do not silently retry.
404	`resource_missing`	The referenced ID does not exist in this environment.
409	`idempotency_conflict`	Same idempotency key reused with different parameters.
429	`rate_limited`	Too many requests; back off per the `Retry-After` header.
500	`internal_error`	Unexpected platform fault; safe to retry with backoff.

Error response shape

```
{
  "error": {
    "code": "card_declined",
    "message": "The card was declined by the issuing bank.",
    "request_id": "req_3Lm8pQxV2n"
  }
}
```

Note

Always log `request_id` alongside application errors. Support cannot look up a failed call without it, since request bodies are not retained beyond 72 hours.

6 Webhooks

Register an HTTPS endpoint under `/webhooks/endpoints` to receive asynchronous events — settlement, chargebacks, and payouts all arrive this way instead of through polling. Every delivery is signed with HMAC-SHA256 so the endpoint can verify it genuinely originated from Nexa.

Verifying the signature (Node.js)

```
import crypto from "crypto";

function verifySignature(rawBody, header, secret) {
  const [tPart, sigPart] = header.split(",");
  const timestamp = tPart.split("=")[1];
  const signature = sigPart.split("=")[1];

  const expected = crypto
    .createHmac("sha256", secret)
    .update(`${timestamp}.${rawBody}`)
    .digest("hex");

  return crypto.timingSafeEqual(
    Buffer.from(expected),
    Buffer.from(signature)
  );
}
```

6.1 Key event types

- `payment.succeeded` — capture completes, funds scheduled for settlement.
- `payment.failed` — an authorization or capture attempt is declined.
- `payout.paid` — a scheduled payout to the merchant's bank clears.
- `chargeback.opened` — a cardholder disputes a settled payment.

Caution

Respond to webhook deliveries with a `2xx` within 10 seconds. Nexa retries undelivered events with exponential backoff for 72 hours, then marks the event as permanently failed and surfaces it in the dashboard's delivery log.

6.2 Rate limits & pagination

Rate limits are enforced per API key on a rolling 60-second window; usage is always returned on every response via headers, so clients need no separate endpoint to check remaining quota. `X-RateLimit-Limit` and `X-RateLimit-Remaining` report the window totals, and a `429` response adds `Retry-After` with the seconds to wait before retrying. Bulk operations (customer imports, ledger exports) should use `/ledger/entries` cursor pagination rather than requesting a higher ceiling — 600/min is rarely the real bottleneck once paging is implemented.

7 Changelog

Version	Date	Summary
2.4.0	2026-03-02	Added `idempotency_conflict` error code; SDKs auto-retry `5xx` with jitter.
2.3.0	2025-11-18	Introduced `/ledger/entries` for settlement reconciliation.
2.1.0	2025-02-14	Initial public release of the Node.js SDK alongside Python.

Developer support: api-support@nexa.dev | Full reference: api.nexa.dev/docs | Status: status.nexa.dev