Vindex Grid

FOR DEVELOPERS

Vindex Grid API guide

Verify your customers and screen people and companies against sanctions and PEP lists, from inside your own product.

BASE URL

`https://api.vindexlegalrisk.com/v1`

API VERSION

`2026-11-05`

AUTHENTICATION

`Bearer vdx_test_… or vdx_live_…`

ONLINE

`vindexlegalrisk.com/docs/api`

INSIDE

1 The Vindex API
2 Sign up and sign in
3 Keys, limits and conventions
4 Verifications
5 Screenings
6 Webhooks
7 Test mode
8 Credits and billing
9 Errors
10 Go live
11 API reference

Published 12 November 2026. The online guide at **vindexlegalrisk.com/docs/api** is always the latest version.

Contents

1 **The Vindex API** 4

Verify your customers and screen names from your own product. Start with the 10-minute quickstart.

What you can build 4
How it works 4
Quickstart 5
Next steps 7

2 **Sign up and sign in** 8

Create a Vindex account, sign in to the Developers console, open developer access and invite your team.

People and servers 8
Create an account 8
Sign in 8
Open developer access 9
Invite your team 9
Confirm it is you 10
Forgotten password 10
Sign-up and sign-in API 10
When someone leaves 12

3 **Keys, limits and conventions** 14

API keys, test and live mode, versions, request ids, rate limits, safe retries, paging and metadata.

API keys 14
Test and live mode 14
Versions and request ids 15
Rate limits 15
Idempotent retries 16
Lists and pagination 16
Metadata 16

4 **Verifications** 18

Send a customer through Vindex's hosted flow, follow it to an outcome, and fetch the reports.

The life of a verification 18
Hosted links 19
What your customer sees 20
Credits for a verification 20
When Vindex needs more 20
Outcomes and your decision 21
Reports 21
Company verifications 21

5 **Screenings** 23

Screen a person or company against sanctions, PEP and adverse-media lists, one at a time or in batches.

One name or a batch 23
Reading a result 24
When the provider is down 25
Live screening 25

6 Webhooks 26

Get told when things happen: signed deliveries, retries, safe handling and every event.

Set up an endpoint 26
Verify the signature 26
Retries and failures 28
Handle events safely 29
Rotate the secret 29
Event catalogue 30

7 Test mode 32

Build and test for free: simulated outcomes, test names, test credits and test helpers.

What test mode does 32
Test credits 32
Choose an outcome 32
Screening test names 33
Test helpers 33

8 Credits and billing 35

How prepaid credits work: prices, holds and charges, buying, auto-recharge, expiry and refunds.

How credits work 35
Holds and charges 35
Running out 36
Buying credits 36
Auto-recharge 37
Expiry, refunds and disputes 37

9 Errors 38

The error format, validation errors, and every error code with what to do about it.

The error shape 38
Validation errors 38
Error codes 39

10 Go live 42

Apply for live access, the checklist before you switch, and what stays your responsibility.

Apply for live access 42
Before you switch 42
Your responsibilities 42

11 API reference 44

Every endpoint, field, error and webhook event, with code in curl, Node and Python.

Account 45
Verifications 46
Reports 69
Screenings 71
Balance 87
Events 89
Test helpers 93
Webhook events 104

CHAPTER 1

The Vindex Grid API

Verify your customers with Vindex, and screen people and companies against sanctions and PEP lists, from inside your own product. Build in test mode for free, then go live.

What you can build

Verify customers
Send your customer a Vindex link. They upload documents, consent and do a face check; Vindex reviews and decides.

Screen names
Check a person or company against sanctions, PEP and adverse-media lists, one by one or 100 at a time.

Get told
Signed webhooks tell your server when a customer submits, when Vindex needs more, and when it decides.

Pay as you go
Buy credits, spend them per verification or screening. Test mode is free.

How it works

1. YOUR SERVER
Create a verification
One POST. You get back an id and a hosted link.

2. YOUR CUSTOMER
Complete Vindex's flow
Documents, consent and a face check, branded with your name.

3. VINDEX
Screen and review
World-Check screening, document checks and a human review.

4. YOUR SERVER
Receive a webhook
verification.decided, signed so you know it's from Vindex.

5. YOUR SERVER
Fetch the outcome
Outcome, risk rating, and the outcome letter as a PDF.

Everything you can do has a **test mode** twin that costs nothing and never reaches Vindex's reviewers, so you can build the whole integration before you go live.

Quickstart

About ten minutes, all in test mode.

1. **Get a test key.** Create a Vindex account and sign in, open the Developers console and turn on developer access. You get a test key that starts `vdx_test_`; it is shown once, so keep it in your secrets store. Check it works:

```
Terminal
export VINDEX_API_KEY=vdx_test_...

curl https://api.vindexlegalrisk.com/v1/me -H "Authorization: Bearer $VINDEX_API_KEY"
```

2. **Create a verification** for a customer. Send what you already know; they confirm and complete the rest.

```
curl
curl https://api.vindexlegalrisk.com/v1/verifications \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -H "Idempotency-Key: $(uuidgen)" \
  -H "Content-Type: application/json" \
  -d '{
    "type": "individual",
    "external_reference": "crm-42",
    "individual": {
      "first_name": "Aarav",
      "last_name": "Sharma",
      "email": "aarav.sharma@example.com",
      "country": "IN"
    },
    "redirect_url": "https://acme.example/kyc/done"
  }'
```

```
Node
const response = await fetch("https://api.vindexlegalrisk.com/v1/verifications", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
    "Idempotency-Key": crypto.randomUUID(),
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    type: "individual",
    external_reference: "crm-42",
    individual: { first_name: "Aarav", last_name: "Sharma", email: "aarav.sharma@example.com", country:
"IN" },
    redirect_url: "https://acme.example/kyc/done",
  }),
});
const verification = await response.json();
console.log(verification.hosted_link.url); // send this to your customer
```

Python

```python
import os
import uuid
import requests

response = requests.post(
    "https://api.vindexlegalrisk.com/v1/verifications",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
        "Idempotency-Key": str(uuid.uuid4()),
    },
    json={
        "type": "individual",
        "external_reference": "crm-42",
        "individual": {"first_name": "Aarav", "last_name": "Sharma", "email":
"aarav.sharma@example.com", "country": "IN"},
        "redirect_url": "https://acme.example/kyc/done",
    },
    timeout=30,
)
response.raise_for_status()
verification = response.json()
print(verification["hosted_link"]["url"])  # send this to your customer
```

3. **Send the hosted link** (`hosted_link.url`) to your customer: by email, SMS or a button in your app. In test mode, open it yourself: accept the notice and choose **Complete as test customer**.

4. **Pick the outcome.** In test mode Vindex decides about 20 seconds after the customer submits. Choose the outcome with `"test_scenario": "reject"` when you create the verification, or give the customer a last name ending in `Approved`, `Conditional`, `Rejected` or `ActionRequired`. More test controls.

5. **Receive the webhook.** Add an endpoint under **Developers → Webhooks** (test mode), copy its signing secret, and check every delivery's signature with a Standard Webhooks library:

Python

```python
import os
from standardwebhooks.webhooks import Webhook

webhook = Webhook(os.environ["VINDEX_WEBHOOK_SECRET"].removeprefix("whsec_"))

def handle(raw_body: bytes, headers: dict) -> None:
    event = webhook.verify(raw_body, headers)  # raises if the signature is wrong or stale
    if event["type"] == "verification.decided":
        verification_id = event["data"]["id"]
        ...  # fetch it with your key and update your records
```

Node

```
import { Webhook } from "standardwebhooks";

const webhook = new Webhook(process.env.VINDEX_WEBHOOK_SECRET.replace(/^whsec_/, ""));

export function handle(rawBody, headers) {
  const event = webhook.verify(rawBody, headers); // throws if the signature is wrong or stale
  if (event.type === "verification.decided") {
    const verificationId = event.data.id;
    // fetch it with your key and update your records
  }
}
```

6 **Fetch the result and the PDF.** The webhook says which verification changed; your server reads the details with its key.

Terminal

```
curl https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy \
  -H "Authorization: Bearer $VINDEX_API_KEY"

curl -G https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/report \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -d type=outcome_letter \
  -o outcome-letter.pdf
```

Test mode is free and simulated

Test verifications never reach Vindex's reviewers, test screenings never reach World-Check, and test credits refill themselves. When your integration works, apply for live access.

Next steps

Sign up and sign in →
Create a Vindex account, sign in to the Developers console, open developer access and invite your team.

API reference →
Every endpoint, field, error and webhook event, with code in curl, Node and Python.

Verifications →
Send a customer through Vindex's hosted flow, follow it to an outcome, and fetch the reports.

Webhooks →
Get told when things happen: signed deliveries, retries, safe handling and every event.

Test mode →
Build and test for free: simulated outcomes, test names, test credits and test helpers.

Go live →
Apply for live access, the checklist before you switch, and what stays your responsibility.

CHAPTER 2

Sign up and sign in

Before you call the API you need a Vindex account. Create one, sign in to the Developers console, open developer access for your company, and invite the people who build with you.

People and servers

People sign in
With an email address and password, to the Developers console: keys, webhooks, logs, credits and team.

Servers use keys
Every API request carries a secret key. The API never takes a password, and there is no sign-in call to make.

Each person on your team signs in with their own account. Your servers never sign in: they send a key, as described in Keys, limits and conventions.

Create an account

1. Open vindexlegalrisk.com/app/register, or press **Sign in** on any Vindex page and then **Create an account**.
2. Enter your **work email address**, a password (twice), your name and your mobile number, then press **Create account**.
3. Vindex emails you a **6-digit confirmation code**. Type it on the confirmation page and press **Confirm my email**. Closed the page? Open vindexlegalrisk.com/app/verify, or press **Have a code? Confirm your email** under the sign-in form.
4. Sign in with your email address and password.

Use an address your company keeps
The account that opens developer access becomes its **owner**. Use a work address that will outlive any one job, and invite colleagues rather than sharing one sign-in.

Sign in

1. Open vindexlegalrisk.com/app/login. Under the form, check that it says you are signing in to the **Customer** workspace; if not, press **Change** and pick it.
2. Type your email address and password and press **Sign in**. If your account uses a second sign-in step, Vindex then asks for a 6-digit code.
3. Open the Developers console: **Developers** in the bar on the left.

Not confirmed yet	Vindex asks you to confirm your email first. Use the code from the email it sent when you made the account.
Wrong email or password	Vindex says only that the two don't match, never which one is wrong. Check both, or reset your password.
Signed out after a while	Sign-ins expire. Sign in again; nothing you saved is lost. Your API keys are not affected: they keep working until you revoke them.

Open developer access

The first time you open the Developers console it offers developer access for your company. It takes a few seconds.

1. Enter your **company or product name**, your country (2 letters, such as `IN`) and, if you like, your website.
2. Press **Open developer access**. Vindex creates your developer account in **test mode**, with free test credits, and shows your first **test key** (`vdx_test_…`).
3. Copy the key into your secrets store now: it is shown once. You can create more keys at any time under **API keys**.

> **Customer accounts only**
> Developer access belongs to customer accounts, and each person can open it once. Live keys come later, when Vindex approves your company: see Go live.

Invite your team

In the console, open **Settings**, then **Team**. Enter a colleague's email address, choose a role and press **Invite**. If they already have a Vindex account they join at once; if not, they join as soon as they sign up with that address. Owners and admins can change a role or remove someone at any time.

ROLE	CAN
Owner	Everything. The person who opened developer access; can't be removed, and is the only one who can add admins.
Admin	The team, test and live keys, webhooks, credits and billing, settings, and the application to go live.
Developer	Test keys and webhooks, the request log with request bodies, and usage. Can't create live keys.
Billing	Credits, auto-recharge, invoices and usage limits. Sees usage; no keys or webhooks.
Viewer	Reads everything in the console and changes nothing.

Vindex staff accounts can't join a partner team; invite your colleagues with their own work addresses.

Confirm it is you

A few changes touch your live account, so the console asks for your password again before it makes them, even though you are signed in:

- creating a live key, or rolling one;
- showing or rolling the signing secret of a live webhook endpoint.

Type your password (and your code, if your account uses one) in the **Confirm it is you** window. The check lasts 10 minutes, so a few changes in a row ask only once. Test keys and test endpoints never ask.

Forgotten password

1. On the sign-in page press **Forgot password?**, or open vindexlegalrisk.com/app/reset.
2. Enter your email address and press **Email me a code**.
3. Enter the code from the email and a new password, press **Save new password**, then sign in with it.

Lost access to the email address itself? Ask your company's owner or an admin to invite your new address, and tell Vindex support so the old account can be closed.

Sign-up and sign-in API

Every step above is also an API call, for building your own sign-up screen or scripting test accounts. These calls are on the Vindex app API, `https://api.vindexlegalrisk.com` (not under `/v1`), and take no API key. The tokens they return sign a person in to Vindex and the Developers console; the `/v1` API accepts only API keys.

CALL	SEND	GET BACK
Create an account `POST /auth/register`	`email`, `password`, `display_name`, `phone`. For a company add `signup_type: "organization"`, `organization_name` and `organization_country` (2 letters).	`account.email` and `verification.delivery: "email"`: Vindex emails a 6-digit code.
Confirm the email `POST /auth/verify-email`	`email`, `code`	`email_verified: true`
Sign in `POST /auth/login`	`email`, `password`, and `mfa_code` if the account has a second step	`status`, `access_token`, `refresh_token`, `token_type`, `expires_at` and `user` (`id`, `email`, `display_name`)
Ask for a reset code `POST /auth/password-reset/request`	`email`	`delivery: "email"`: Vindex emails a reset code.

CALL	SEND	GET BACK
Set a new password `POST /auth/password-reset/confirm`	`email`, `token` (the code from the email), `new_password`	`ok: true`

curl

```
# 1. Create the account (Vindex emails a 6-digit code)
curl https://api.vindexlegalrisk.com/auth/register \
  -H "Content-Type: application/json" \
  -d '{"email": "dev@acme.example", "password": "a-long-passphrase", "display_name": "Asha Rao", "phone":
"+91 98765 43210"}'

# 2. Confirm the email with the code
curl https://api.vindexlegalrisk.com/auth/verify-email \
  -H "Content-Type: application/json" \
  -d '{"email": "dev@acme.example", "code": "482913"}'

# 3. Sign in
curl https://api.vindexlegalrisk.com/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email": "dev@acme.example", "password": "a-long-passphrase"}'
```

Node

```
const base = "https://api.vindexlegalrisk.com";
const post = (path, body) =>
  fetch(base + path, { method: "POST", headers: { "Content-Type": "application/json" }, body:
JSON.stringify(body) })
    .then(async (response) => {
      const data = await response.json();
      if (!response.ok) throw new Error(data.detail);
      return data;
    });

await post("/auth/register", { email: "dev@acme.example", password: "a-long-passphrase", display_name: "Asha
Rao", phone: "+91 98765 43210" });
await post("/auth/verify-email", { email: "dev@acme.example", code: "482913" }); // the code from the email
const session = await post("/auth/login", { email: "dev@acme.example", password: "a-long-passphrase" });
console.log(session.status, session.expires_at); // "authenticated", then use session.access_token
```

Python

```python
import requests

BASE = "https://api.vindexlegalrisk.com"

def post(path: str, body: dict) -> dict:
    response = requests.post(BASE + path, json=body, timeout=30)
    if not response.ok:
        raise RuntimeError(response.json().get("detail"))
    return response.json()

post("/auth/register", {"email": "dev@acme.example", "password": "a-long-passphrase", "display_name": "Asha Rao", "phone": "+91 98765 43210"})
post("/auth/verify-email", {"email": "dev@acme.example", "code": "482913"})  # the code from the email
session = post("/auth/login", {"email": "dev@acme.example", "password": "a-long-passphrase"})
print(session["status"], session["expires_at"])  # "authenticated", then use session["access_token"]
```

Using the token	Send `access_token` as `Authorization: Bearer …` to the app API until `expires_at`, then sign in again.
Errors	Errors are JSON with a `detail` message: `401` wrong email or password, `403` email not confirmed or a password reset required, `409` an account already exists for that email, `422` a wrong or expired code.
Second sign-in step	If `status` is not `authenticated`, the account still has its second step to finish: send the 6-digit code as `mfa_code` with the email and password.

⚠ **Accounts are for your team, not your customers**

Never collect or store your customers' Vindex passwords. To verify a customer, create a **verification**: they complete Vindex's hosted flow without an account.

When someone leaves

Remove them	**Settings → Team**, then remove the person. They lose the console at once; the **Audit log** keeps what they did.
Roll the keys they saw	Keys belong to your company, not to a person, so removing someone doesn't stop a key they copied. Roll any key they could have seen, with an overlap so your servers switch without downtime.
If the owner leaves	The owner can't be removed in the console. Contact Vindex support before they go, so your company keeps an owner.

Keys, limits and conventions →
API keys, test and live mode, versions, request ids, rate limits, safe retries, paging and metadata.

Go live →
Apply for live access, the checklist before you switch, and what stays your responsibility.

Test mode
Build and test for free: simulated outcomes, test names, test credits and test helpers.

Credits and billing
How prepaid credits work: prices, holds and charges, buying, auto-recharge, expiry and refunds.

CHAPTER 3

Keys, limits and conventions

How requests are authenticated, which mode they act in, and the conventions every endpoint shares: versions, request ids, rate limits, safe retries, paging and metadata.

API keys

Every request carries a secret key in the `Authorization` header. Create keys in the Developers console.

Authenticating

```
curl https://api.vindexlegalrisk.com/v1/me \
  -H "Authorization: Bearer vdx_test_..."

# or, with curl's -u (the key as the user name, no password)
curl https://api.vindexlegalrisk.com/v1/me -u vdx_test_...:
```

What a key looks like	`vdx_test_` or `vdx_live_`, then 38 letters and digits. The last six are a checksum, so a mistyped key is refused straight away and secret scanners can recognise a leaked one.
Shown once	A key is shown in full only when it is created. Afterwards the console shows its start and last four characters. Keep it in your secrets store, never in code or chat.
Permissions	A key can do everything, or be limited: **verifications** only, **screenings** only, **read only**, or a custom set of `verifications:read`, `verifications:write`, `screenings:read`, `screenings:write`, `events:read` and `balance:read`. A call outside the key's permissions returns `403 insufficient_scope`.
Address allowlist	Optional. List the IP addresses or CIDR ranges your servers call from; anything else gets `403 ip_not_allowed`.
Rolling and revoking	**Roll** a key to replace it: the new key works at once and the old one keeps working for the overlap you choose (none to 7 days), so you can deploy without downtime. **Revoke** stops a key immediately. Keys can also expire on a date you set.

⚠ **Never use a key in a browser or app**

Keys are secrets for your server. A request that comes from a web page on another site is refused with `403 secret_key_in_browser`, and Vindex never sends browsers CORS permission to call the API. If a key leaks, revoke it in the console.

Test and live mode

The key decides the mode: a test key acts in test mode, a live key in live mode. The two never mix.

Test mode	Free and simulated. Verifications never reach Vindex's reviewers and screenings never reach World-Check. Test credits refill themselves. See Test mode.
Live mode	Real verifications and screenings, paid with credits. Available once Vindex approves your account; until then a live key gets `403 live_mode_not_enabled`.
Kept apart	Each mode has its own objects, events, webhook endpoints, signing secrets, request log and credits. Every object and event has `livemode` so you can tell them apart.

Versions and request ids

The API is `/v1`. Every response carries `Vindex-Version: 2026-11-05`, the date of the version that answered. Vindex adds fields and endpoints without warning, so ignore fields you don't know. Anything that would break an integration ships as a new dated version, announced in advance.

Every response also carries a `Request-Id` (`req_…`). Error bodies repeat it as `request_id`. Log it, and quote it to Vindex support: it finds the exact request in your request log. Responses are sent with `Cache-Control: no-store` because they can contain customer data.

Rate limits

Limits are per minute, per account and mode, shared by all your keys.

REQUESTS	LIVE, PER MINUTE	TEST, PER MINUTE
Read GET requests	600	300
Write POST requests, except screenings	120	60
Screening POST /screenings and /screening_batches	60	60

Every response says where you stand, in the IETF `RateLimit` header fields. Over the limit you get `429 rate_limited` with `Retry-After`: wait that many seconds, then retry. Need more? Ask Vindex.

Headers

```
RateLimit-Policy: "write";q=120;w=60
RateLimit: "write";r=97;t=23

HTTP/1.1 429 Too Many Requests
Retry-After: 23
```

Idempotent retries

Networks fail. Send an `Idempotency-Key` header on every POST (a UUID is ideal) and you can retry safely: Vindex does the work once and charges once.

Same key, same body	Within 24 hours, Vindex returns the first response again, with `Idempotent-Replayed: true`.
Same key, different body	Refused with `422 idempotency_key_reused`. Use a new key for a new request.
First request still running	`409 idempotency_key_in_use` with `Retry-After: 1`. Wait a second and retry.
Failed requests	Only successful responses are remembered. If a request fails (for example `402`), fix the cause and retry with the same key.
Scope	Keys are remembered per account and mode, up to 255 characters.

Lists and pagination

Lists come newest first, as `{ "object": "list", "data": [...], "has_more": true }`. Ask for up to 100 at a time with `limit` (default 10). For the next page, send `starting_after` with the id of the last item you received, while `has_more` is true.

The next page

```
curl -G https://api.vindexlegalrisk.com/v1/verifications \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -d limit=100 \
  -d starting_after=ver_0vZ3kq9XfJ1mB7c2Lr8sTy
```

Verifications can also be filtered by `status`, `external_reference`, `created_gte` and `created_lt`; events by `type` (a group works too, such as `verification.*`) and `object_id`.

Metadata

Verifications, screenings and batches take `metadata`: up to 50 keys of your own, each key up to 40 characters (no `[` or `]`) and each value a string of up to 500. Vindex stores it and returns it, nothing more. Use `external_reference` (up to 255 characters, searchable) for your own id.

> ⚠ **No personal data in metadata**
> Metadata and references appear in logs, events and the console. Put ids there, never names, emails, dates of birth or document numbers.

API reference → Errors →

Every endpoint, field, error and webhook event, with code in curl, Node and Python.

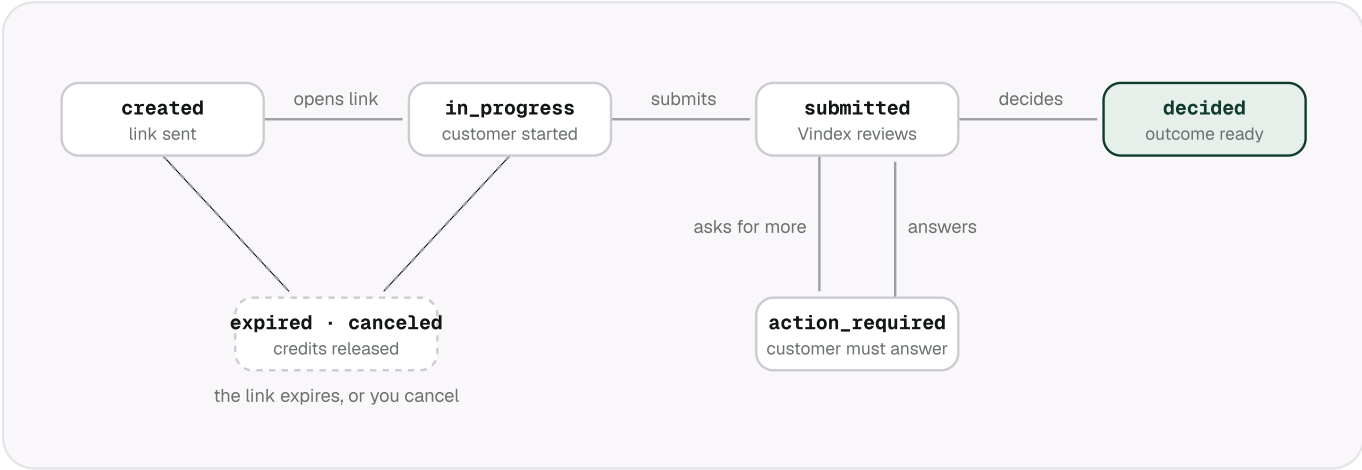
The error format, validation errors, and every error code with what to do about it.

CHAPTER 4

Verifications

A verification asks Vindex to verify one of your customers: a person (KYC) or a company (KYB). Your customer completes Vindex's hosted flow, Vindex screens and reviews, and you get the outcome and the reports.

The life of a verification



The statuses of a verification. Credits are held at `created` and charged at `submitted`.

`status` is the state to build on. `stage` is finer progress you can show your own team or your customer; it may gain values over time, so treat unknown stages as "in progress".

`awaiting_customer` status created	The link is waiting for your customer.
`customer_in_progress` status in_progress	Your customer opened the link and is filling it in.
`screening` status submitted	Submitted. Documents are being read and the customer screened.
`under_review` status submitted	Vindex's reviewers are looking at the case.
`final_review` status submitted	The case is with the final decision maker.
`awaiting_customer_response` status action_required	Vindex asked your customer for more.
`appeal_review` status submitted	The customer appealed a decision; it is being looked at again.

decided status decided	Vindex reached its outcome.

Nothing sensitive leaks through the stage

Enhanced due diligence and any suspicious-activity review all show as `under_review`. That keeps Vindex and you clear of tipping-off rules, and it means you can show the stage to your customer safely.

Hosted links

Creating a verification returns `hosted_link.url`. Send it to your customer however suits you: email, SMS, or a button in your app. It works on phones, where most customers complete it.

Only returned once	The URL is in the response to the call that created it and is never stored by Vindex. Later reads show only when it expires. To send it again, create a new one.
Expiry	7 days by default. Set `link_expires_in` (seconds) from one hour to 30 days. A verification whose link expires before the customer submits becomes `expired` and its credits are released.
A new link	`POST /v1/verifications/{id}/hosted_link` issues a fresh link and the previous one stops working. Use it when a link expired, was lost, or Vindex needs more from the customer.
Reopening	The customer can close the page and come back with the same link until it expires; progress is saved.
Where it lands	When they finish, the customer can return to your `redirect_url`, with `verification_id` and `status` added to it. Use it for a "thanks" page only, never as proof: always read the verification with your key.

curl

```
curl -X POST https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/hosted_link \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{ "link_expires_in": 259200 }'
```

Node

```
const response = await fetch(
  "https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/hosted_link",
  {
    method: "POST",
    headers: { Authorization: `Bearer ${process.env.VINDEX_API_KEY}`, "Content-Type": "application/json" },
    body: JSON.stringify({ link_expires_in: 259200 }), // 3 days
  },
);
const { hosted_link } = await response.json(); // the old link stops working
```

Python

```python
response = requests.post(
    "https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/hosted_link",
    headers={"Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}"},
    json={"link_expires_in": 259200},  # 3 days
    timeout=30,
)
hosted_link = response.json()["hosted_link"]  # the old link stops working
```

The token stays out of logs

The secret part of a link comes after `#`, which browsers never send to servers, so it doesn't end up in web logs or analytics. Treat the link like a password all the same: it lets someone complete your customer's verification.

What your customer sees

A Vindex page with your name and logo (set them under **Developers → Settings**). Before anything else it shows a data-protection notice that names **your company as the collector** and Vindex as processing on your behalf, the purpose, the data collected, who receives it, how long it is kept, and their rights. They confirm three things: they accept, they consent to the face check, and they are 18 or over.

Then they complete Vindex's usual filing: their details, identity documents, and a short video selfie. They don't pay anything; the case fee comes from your credits. Afterwards they see the status of their verification, never screening results.

Vindex doesn't email your customers

You own the relationship. Vindex sends nothing to customers who come through your link; tell them what happens next yourself.

Credits for a verification

The price is **held** from your balance when you create the verification, and **charged** when your customer submits. If the link expires, or you cancel before they submit, the hold is released and nothing is charged. The verification's `credits` object shows the price, what is held and what was charged. More in Credits and billing.

Cancel before submission

```bash
curl -X POST https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/cancel \
  -H "Authorization: Bearer $VINDEX_API_KEY"
```

When Vindex needs more

Sometimes a reviewer needs another document or an answer. The verification moves to `action_required` and you get a `verification.action_required` webhook. Send your customer back to Vindex: create a new hosted link if the

old one expired. When they answer, the verification returns to `submitted` (`verification.resubmitted`). Re-submission is never charged again.

Outcomes and your decision

`approved`	Vindex recommends onboarding.
`approved_with_conditions`	Recommended, with `conditions` you should apply (for example re-verifying an address within 6 months).
`rejected`	Vindex recommends not onboarding.

`risk_rating` (such as `low`, `medium` or `high`) comes with the outcome. Vindex's outcome is a recommendation: the decision to onboard stays yours. Record it, so your audit trail shows who decided and when:

Record your decision

```
curl https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/partner_decision \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{ "decision": "accepted", "decided_by": "Priya Nair, onboarding team" }'
```

Reports

Once a verification is decided, download its PDFs with your key:

`outcome_letter`	Vindex's signed outcome letter. Every decision has one.
`customer_report`	The customer's copy of the case report. Approvals only.

Download

```
curl -G https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/report \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -d type=customer_report \
  -o customer-report.pdf
```

A `report.ready` webhook tells you when each report can be downloaded; before that, the call returns `409 report_not_ready`. Reports contain personal data: store them as securely as your other customer records.

Company verifications

Send `"type": "company"` with a `company` object (legal name, registration number, country, and the contact who will complete the flow). Company verifications work fully in test mode today. Live company verifications are coming soon; until then a live request returns `403 feature_not_enabled`.

Never send Aadhaar numbers

Vindex refuses any request containing what looks like an Aadhaar number (`422 validation_failed`). Customers provide their documents in the hosted flow, where Aadhaar is masked and always optional.

Webhooks

Get told when things happen: signed deliveries, retries, safe handling and every event.

Test mode

Build and test for free: simulated outcomes, test names, test credits and test helpers.

API reference

Every endpoint, field, error and webhook event, with code in curl, Node and Python.

CHAPTER 5

Screenings

Check a person or a company against sanctions, politically exposed person (PEP) and adverse-media lists. Use it to triage before onboarding, or to re-check customers you already have.

One name or a batch

`POST /v1/screenings`	One person or company. The answer comes back in the response, usually within a few seconds.
`POST /v1/screening_batches`	Up to 100 subjects in the background. The response comes at once with the batch's id; you get a `screening_batch.completed` webhook when every subject is done, then read the batch with all its screenings.

curl

```
curl https://api.vindexlegalrisk.com/v1/screenings \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -H "Idempotency-Key: $(uuidgen)" \
  -H "Content-Type: application/json" \
  -d '{
    "subject": { "type": "person", "name": "Aarav Sharma", "date_of_birth": "1990-04-12", "country": "IN" },
    "purpose": "onboarding",
    "external_reference": "crm-42"
  }'
```

Node

```
const response = await fetch("https://api.vindexlegalrisk.com/v1/screenings", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
    "Idempotency-Key": crypto.randomUUID(),
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    subject: { type: "person", name: "Aarav Sharma", date_of_birth: "1990-04-12", country: "IN" },
    purpose: "onboarding",
    external_reference: "crm-42",
  }),
});
const screening = await response.json();
if (screening.result.outcome === "potential_match") {
  // hold onboarding and review, or start a full verification
}
```

Python

```python
response = requests.post(
    "https://api.vindexlegalrisk.com/v1/screenings",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
        "Idempotency-Key": str(uuid.uuid4()),
    },
    json={
        "subject": {"type": "person", "name": "Aarav Sharma", "date_of_birth": "1990-04-12", "country":
"IN"},
        "purpose": "onboarding",
        "external_reference": "crm-42",
    },
    timeout=30,
)
screening = response.json()
if screening["result"]["outcome"] == "potential_match":
    ...  # hold onboarding and review, or start a full verification
```

Give a person's `date_of_birth` and `country` when you have them: they rule out namesakes. Always say why you are screening with `purpose` (`onboarding`, `periodic_review`, `transaction_monitoring` or `other`); it is kept for audit.

Reading a result

A completed screening

```
{
  "id": "scr_0vZ3mT4cQw8nX2Lb9Pk1Ra",
  "object": "screening",
  "livemode": true,
  "status": "completed",
  "result": {
    "outcome": "no_match",
    "screened_at": "2026-11-05T10:31:12Z",
    "provider": "world_check",
    "attribution": "Screening data: LSEG World-Check. Results are an aid to your own checks, not a
decision."
  },
  "credits_charged": 1,
  "external_reference": "crm-42"
}
```

`no_match`	Nothing on the lists looks like this subject.
`potential_match`	One or more list entries look like this subject. It is not a finding: a person has to check whether it is really them. Hold the customer, review, or start a full verification, where Vindex's reviewers do that for you.

Results carry LSEG's `attribution`; show it wherever you show the result. Where Vindex's data agreement allows, results also include `match_count` and `categories` (`sanctions`, `pep`, `adverse_media`); test mode always includes them.

> ⚠ **Don't tell the person screened**
> A result is for your compliance team only. Telling someone they matched a sanctions or PEP list can break anti-money-laundering tipping-off rules.

When the provider is down

If the screening provider can't be reached, the call returns `503 provider_unavailable` and **nothing is charged**. Retry later with the same `Idempotency-Key`. In a batch, a subject that can't be screened is marked `failed` and not charged; the rest of the batch carries on.

Credits: a screening is charged only when it completes. A batch holds its full price up front, charges each subject as it is screened, and releases whatever is left at the end.

Live screening

Screening runs on LSEG World-Check. Test screening works now. Live screening through the API opens once LSEG approves it for Vindex's partners; until then a live request returns `403 feature_not_enabled`. Verifications already include screening, so nothing else is blocked.

> 💡 **Try it in test mode**
> Names containing `Sanctioned`, `Pep`, `Adverse` or `Multimatch` return matches, and `Unavailable` simulates an outage. See Test mode.

API reference →
Every endpoint, field, error and webhook event, with code in curl, Node and Python.

Test mode →
Build and test for free: simulated outcomes, test names, test credits and test helpers.

Credits and billing →
How prepaid credits work: prices, holds and charges, buying, auto-recharge, expiry and refunds.

CHAPTER 6

Webhooks

Vindex tells your server when something happens: a customer submitted, Vindex needs more from them, a decision was made, a report is ready. Every delivery is signed, so you know it came from Vindex.

Set up an endpoint

1. Build a route on your server that accepts `POST` requests with a JSON body, for example `https://acme.example/webhooks/vindex`.
2. In the Developers console, under **Webhooks**, add the URL and choose the events it should receive (or all of them). Each mode has its own endpoints.
3. Copy the endpoint's signing secret (`whsec_…`) into your server's secrets, then press **Send test event** to see a delivery arrive.

Live endpoints	`https://` on the standard port (443), to a host name, never an IP address.
Test endpoints	`https://` on any port. To reach your laptop while you build, use a tunnel such as ngrok.
Never allowed	Private, internal and cloud-metadata addresses, and URLs with a user name or password in them.

Verify the signature

Vindex signs with Standard Webhooks, so a ready-made library in your language does the checking. Each delivery has three headers:

A delivery

```
POST /webhooks/vindex HTTP/1.1
Content-Type: application/json
webhook-id: evt_0vZ3kq9XfJ1mB7c2Lr8sTy
webhook-timestamp: 1762511719
webhook-signature: v1,K5oZfzN95Z9UVu1EsfQmfVNQhnkZ2pj9o9NDN/H/pI4=

{"id":"evt_0vZ3kq9XfJ1mB7c2Lr8sTy","object":"event","type":"verification.decided",...}
```

`webhook-id`	The event's id. The same on every retry of that event, so use it to skip duplicates.
`webhook-timestamp`	When this attempt was signed, in Unix seconds. Refuse anything more than 5 minutes off.

`webhook-signature`	`v1,` and a base64 HMAC-SHA256 of `{id}.{timestamp}.{body}` made with your secret. While you rotate the secret there are two, separated by a space; accept the delivery if either matches.

Python

```python
# pip install standardwebhooks
import os
from standardwebhooks.webhooks import Webhook

webhook = Webhook(os.environ["VINDEX_WEBHOOK_SECRET"].removeprefix("whsec_"))

# Flask shown; any framework works. Read the body as raw bytes, before any JSON parsing.
@app.post("/webhooks/vindex")
def vindex_webhook():
    try:
        event = webhook.verify(request.get_data(), request.headers)
    except Exception:
        return "", 400  # wrong signature or too old: ignore it
    queue.enqueue(handle_event, event)  # do the work later
    return "", 204  # answer quickly
```

Node

```node
// npm install standardwebhooks
import express from "express";
import { Webhook } from "standardwebhooks";

const webhook = new Webhook(process.env.VINDEX_WEBHOOK_SECRET.replace(/^whsec_/, ""));
const app = express();

// The raw body, exactly as received: parsing and re-serialising breaks the signature.
app.post("/webhooks/vindex", express.raw({ type: "application/json" }), (req, res) => {
  let event;
  try {
    event = webhook.verify(req.body, req.headers);
  } catch {
    return res.sendStatus(400); // wrong signature or too old: ignore it
  }
  queue.add(event); // do the work later
  res.sendStatus(204); // answer quickly
});
```

Without a library

The same check by hand. Compare signatures in constant time, and use the raw body.

Python

```python
import base64
import hashlib
import hmac
import time

def verify(secret: str, headers, body: bytes, tolerance: int = 300) -> None:
    """Raise unless the delivery is signed with this secret and fresh."""
    message_id = headers["webhook-id"]
    timestamp = headers["webhook-timestamp"]
    if abs(time.time() - int(timestamp)) > tolerance:
        raise ValueError("timestamp outside the tolerance")
    key = base64.b64decode(secret.removeprefix("whsec_"))
    signed = f"{message_id}.{timestamp}.".encode() + body
    expected = base64.b64encode(hmac.new(key, signed, hashlib.sha256).digest()).decode()
    for signature in headers["webhook-signature"].split(" "):
        version, _, value = signature.partition(",")
        if version == "v1" and hmac.compare_digest(value, expected):
            return
    raise ValueError("no valid signature")
```

Node

```node
import crypto from "node:crypto";

export function verify(secret, headers, rawBody, toleranceSeconds = 300) {
  const id = headers["webhook-id"];
  const timestamp = headers["webhook-timestamp"];
  if (Math.abs(Date.now() / 1000 - Number(timestamp)) > toleranceSeconds) {
    throw new Error("Timestamp outside the tolerance");
  }
  const key = Buffer.from(secret.replace(/^whsec_/, ""), "base64");
  const expected = crypto
    .createHmac("sha256", key)
    .update(`${id}.${timestamp}.`)
    .update(rawBody)
    .digest("base64");
  const valid = headers["webhook-signature"].split(" ").some((part) => {
    const [version, value] = part.split(",");
    return (
      version === "v1" &&
      value.length === expected.length &&
      crypto.timingSafeEqual(Buffer.from(value), Buffer.from(expected))
    );
  });
  if (!valid) throw new Error("No valid signature");
}
```

Retries and failures

A delivery succeeds when your endpoint answers **2xx within 15 seconds**. Anything else (an error, a timeout, a redirect) counts as a failure and is retried:

1	Immediately
2	5 seconds later
3	5 minutes later
4	30 minutes later
5	2 hours later
6	5 hours later
7	10 hours later
8	10 hours later
9	24 hours later
10	24 hours later

In **live mode** that is ten attempts over about three days. Waits vary by up to 20% so retries don't arrive in a burst. In **test mode** only the first four are made.

Redirects are not followed: give the final URL. If your endpoint answers `429` or `503` with `Retry-After`, Vindex waits at least that long. `410 Gone` turns the endpoint off at once.

Failing for an hour	Vindex emails your team's owners, admins and developers, and marks the endpoint in the console.
Failing for 3 days	A second email.
No success for 5 days	The endpoint is turned off, and you get an email saying so.
Recovering	Fix the endpoint, turn it back on, and use **Resend failed events** in the console, or read what you missed from `GET /v1/events` (kept 30 days).

Handle events safely

- **Answer first, work later.** Verify, queue the event, return `204`. Slow handlers time out and get retried.
- **Expect duplicates.** Delivery is at least once. Skip a `webhook-id` you have already handled.
- **Expect any order.** Events can arrive out of order. Read the object with your key and act on its current status, not on the event alone.
- **No personal data.** Events are thin: ids, statuses and your references. Fetch details with your key when you need them.
- **Check the mode.** `livemode` tells test events from live ones.

Rotate the secret

Roll secret in the console gives you a new secret. For the overlap you choose (none, an hour, or up to 24 hours) Vindex signs every delivery with both, so you can update your servers without missing anything. Libraries accept a delivery if any signature matches. Every reveal and roll is written to your account's audit log.

Event catalogue

Every event has the same envelope; `data` describes the object. Full examples for each type are in the API reference.

verification.decided

```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "verification.decided",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
    "object": "verification",
    "type": "individual",
    "status": "decided",
    "stage": "decided",
    "outcome": "approved",
    "risk_rating": "low",
    "external_reference": "crm-42"
  }
}
```

Verifications

`verification.created`	A verification was created and its hosted link issued.
`verification.started`	The customer opened the hosted link and started.
`verification.submitted`	The customer submitted; Vindex review has started.
`verification.action_required`	Vindex needs more from the customer. Send them their hosted link again.
`verification.resubmitted`	The customer answered Vindex's request; review continues.
`verification.decided`	Vindex reached its outcome: approved, approved with conditions, or rejected.
`verification.expired`	The hosted link expired before the customer submitted. Held credits were released.
`verification.canceled`	The verification was canceled. Held credits were released.
`report.ready`	A report PDF (customer report or outcome letter) can be downloaded.

Screenings

`screening_batch.completed`	Every subject in a batch screening has been screened.

Billing

`billing.credits_purchased`	Credits were added to your live balance after a payment.
`billing.balance_low`	Your live balance dropped to or below your alert threshold.
`billing.credits_exhausted`	Your live balance reached zero; paid calls now return 402.
`billing.credits_expiring`	Some credits expire within the next 30 days.
`billing.auto_recharge_failed`	An automatic top-up failed; auto-recharge is paused.

Account

`api_key.expiring`	An API key expires within 7 days.

See every delivery

The console's webhook page shows each attempt with the status your endpoint answered, how long it took and the first part of its reply, and lets you retry one now. Calling `https://api.vindexlegalrisk.com/v1/events` with your key lists what was sent.

API reference →
Every endpoint, field, error and webhook event, with code in curl, Node and Python.

Verifications →
Send a customer through Vindex's hosted flow, follow it to an outcome, and fetch the reports.

CHAPTER 7

Test mode

Everything in the API works in test mode, for free. Nothing reaches Vindex's reviewers or World-Check, outcomes are simulated, and you choose them, so you can test every path your integration needs.

What test mode does

Your test key	Starts `vdx_test_`. Everything it creates is test data, kept apart from live: `livemode` is false on every object and event.
Verifications	The hosted link opens a test version of the flow: your customer (or you) sees the notice and the steps, then presses **Complete as test customer**. About 20 seconds after submission, Vindex decides.
Screenings	Simulated from the name: test names below give matches; anything else is `no_match`.
Reports	Sample PDFs, marked as test samples.
Webhooks	Sent to your test endpoints, signed with their own secrets, retried up to four times. **Send test event** in the console sends any event type on demand.

Test credits

Test mode spends test credits exactly like live mode spends real ones, so your numbers and your `credits` fields look real. You start with 1,000 free test credits and add more in the console. By default they **refill themselves**: a test call never fails for lack of credits.

> **Testing what happens when credits run out**
> Turn off automatic refills of test credits under **Developers → Settings**, spend your test balance, and your next paid call returns `402 insufficient_credits`, just as live would.

Choose an outcome

Pass `test_scenario` when you create a verification, or give the customer a last name (a company name for companies) ending in one of the words below. With neither, the verification is approved.

TEST_SCENARIO	OR LAST NAME	RESULT
`approve`	`…Approved`	approved, risk low
`approve_with_conditions`	`…Conditional`	approved_with_conditions, risk medium, two sample conditions

TEST_SCENARIO	OR LAST NAME	RESULT
reject	…Rejected	rejected, risk high
action_required	…ActionRequired	action_required first; once the customer answers, approved

A verification that will be rejected

```
{
  "type": "individual",
  "individual": { "first_name": "Riya", "last_name": "Rao" },
  "test_scenario": "reject"
}
```

Screening test names

Put one of these words anywhere in the name being screened:

`Sanctioned`	potential_match, category sanctions
`Pep`	potential_match, category pep (the whole word)
`Adverse`	potential_match, category adverse_media
`Multimatch`	potential_match, three matches in all three categories
`Unavailable`	503 provider_unavailable, nothing charged

For example `"Maria Sanctioned"` or `"Unavailable Trading Ltd"`. The names are made up, so they never collide with real list entries.

Test helpers

Move a test verification along without waiting, and without the hosted page. Each call sends the same webhooks a real change would. With a live key these endpoints don't exist (`404`).

`POST …/submit`	The customer finishes the hosted flow (status submitted; the price is charged).
`POST …/request_action`	Vindex asks the customer for more (action_required).
`POST …/respond`	The customer answers (back to submitted).
`POST …/decide`	Vindex decides. Body: { "outcome": "approved" | "approved_with_conditions" | "rejected" }.
`POST …/expire`	The hosted link expires now (expired; the hold is released).

Decide a test verification now

```
curl https://api.vindexlegalrisk.com/v1/test_helpers/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/decide \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{ "outcome": "approved_with_conditions" }'
```

Using a test helper on a verification stops its automatic decision, so the path you script is the path it takes. Full details are in the API reference.

The Vindex API →
Verify your customers and screen names from your own product. Start with the 10-minute quickstart.

Go live →
Apply for live access, the checklist before you switch, and what stays your responsibility.

CHAPTER 8

Credits and billing

The API runs on prepaid credits. You buy them by card, each paid call spends some, and the console shows where every credit went.

How credits work

Value	One credit is worth US$1 at list price. Bigger packs come with bonus credits.
Prices	Each paid operation costs a set number of credits: a verification (a full Vindex case with human review) costs much more than a screening. Your price list is under **Developers → Credits & billing**; negotiated prices show there too.
Free	Reading objects, listing, downloading reports, events, the balance, and everything in test mode.
Two balances	Live credits are bought. Test credits are free and only work with test keys. `GET /v1/balance` returns the balance of the key's mode.

GET /v1/balance

```
{
  "object": "balance",
  "livemode": true,
  "available": 1204,
  "held": 140,
  "frozen": 0,
  "low_balance_threshold": 140,
  "expiring_soon": [{ "credits": 25, "expires_at": "2026-11-28T09:00:00Z" }]
}
```

Holds and charges

Work that finishes later **holds** credits first: they leave your available balance but aren't spent. When the work is done they are **charged**; if it doesn't happen they are **released**. You never pay for a verification your customer didn't submit, or a screening that failed.

A verification

1. Created: the price is held
2. Customer submits: the hold is charged
3. Link expires or you cancel first: the hold is released

A screening

1. Held while it runs
2. Charged when it completes
3. Provider down: released, nothing charged

A batch

1. The whole batch is held up front
2. Each subject is charged as it is screened
3. What's left is released at the end

Credits are spent soonest-expiring first, then bonus before paid, then oldest first. Every movement is recorded in an append-only ledger you can read in the console.

Running out

A paid call your balance can't cover is refused with `402 insufficient_credits`; nothing is held or charged. The error says how many credits were needed and how many you have:

402

```
HTTP/1.1 402 Payment Required
Content-Type: application/problem+json

{
  "type": "https://vindexlegalrisk.com/docs/api/errors/#insufficient_credits",
  "title": "Not enough credits",
  "status": 402,
  "detail": "Creating this verification needs 70 credits; your live balance has 12. Buy credits or turn on auto-recharge in the Developers console.",
  "code": "insufficient_credits",
  "required": 70,
  "available": 12,
  "request_id": "req_0vZ3pQ8rS1tU2vW3xY4zA5"
}
```

Low-balance alert	When your live balance drops to your alert level, your owners, admins and billing team get an email and you get a `billing.balance_low` webhook. At zero: `billing.credits_exhausted`. Set the level under **Developers → Settings**.
Monthly usage limit	Optional, per mode. Calls that would take this month's spending over it are refused with `403 usage_limit_reached`, a guard against runaway loops.

Buying credits

Owners, admins and the billing team buy credits under **Developers → Credits & billing**, once your account is approved for live use. Choose a pack, or an amount of your own, and pay by card on Stripe's secure checkout. Credits appear as soon as the payment succeeds, with a `billing.credits_purchased` webhook, and a paid invoice is ready to download.

Taxes
Your invoice shows any tax that applies to where your company is based. Enter your company's tax id at checkout so it appears on the invoice.

Auto-recharge

Never run dry: when your balance falls to a level you choose, Vindex tops it back up to a target, charging the card you saved at your last purchase. You set the trigger, the target and a monthly cap. If a charge fails, auto-recharge pauses and your billing team is told by email and with a `billing.auto_recharge_failed` webhook.

> ⚠ **Cards issued in India**
> The Reserve Bank of India's e-mandate rules don't allow these automatic, variable charges on Indian-issued cards. With an Indian card, rely on low-balance alerts and one-click top-ups instead; Vindex tells you when it's time.

Expiry, refunds and disputes

Expiry	Bought and bonus credits expire 12 months after purchase. You are reminded 30 and 7 days before, by email and with a `billing.credits_expiring` webhook. Test credits never expire.
Use	Credits are for Vindex services only. They can't be transferred to another account or exchanged for cash.
Refunds	Unspent paid credits can be refunded to the original card, for example after a duplicate purchase. Contact Vindex; a matching share of the pack's bonus is removed with them. Spent and bonus credits aren't refundable.
Disputes	If a card payment is disputed, that purchase's unspent credits are frozen until the dispute closes. If it is lost, they are removed.

See every purchase, invoice and ledger entry under **Developers → Credits & billing**, and the credit errors.

Go live →
Apply for live access, the checklist before you switch, and what stays your responsibility.

Errors →
The error format, validation errors, and every error code with what to do about it.

CHAPTER 9

Errors

Every error is a problem-details object (RFC 9457) with a stable code you can branch on, a sentence you can show people, and a request id for support.

The error shape

An error

```
HTTP/1.1 402 Payment Required
Content-Type: application/problem+json
Request-Id: req_0vZ3pQ8rS1tU2vW3xY4zA5

{
  "type": "https://vindexlegalrisk.com/docs/api/errors/#insufficient_credits",
  "title": "Not enough credits",
  "status": 402,
  "detail": "Creating this verification needs 70 credits; your live balance has 12. …",
  "code": "insufficient_credits",
  "doc_url": "https://vindexlegalrisk.com/docs/api/errors/#insufficient_credits",
  "request_id": "req_0vZ3pQ8rS1tU2vW3xY4zA5",
  "required": 70,
  "available": 12
}
```

- `code` is stable: branch on it. `title` and `detail` are for people and may be reworded.
- `type` and `doc_url` link to the code's entry on this page.
- `request_id` (also the `Request-Id` header) finds the request in your log; quote it to support.
- Some codes add fields: `required` and `available` on 402, `required_scope` on 403.
- The content type is `application/problem+json`; a normal JSON parser reads it.

Which errors to retry

Retry `429`, `409 idempotency_key_in_use`, `500` and `503`, with backoff and the same `Idempotency-Key`. Don't retry other 4xx errors unchanged: fix the request first.

Validation errors

When fields are wrong, `errors[]` lists each one with a dotted `param`, a JSON Pointer (RFC 6901) to the field, and what is wrong. `param` at the top level is the first of them.

422


```
{
  "type": "https://vindexlegalrisk.com/docs/api/errors/#validation_failed",
  "title": "Some fields are not valid",
  "status": 422,
  "detail": "One or more fields failed validation; see errors[].",
  "code": "validation_failed",
  "param": "individual.last_name",
  "errors": [
    {
      "param": "individual.last_name",
      "pointer": "#/individual/last_name",
      "detail": "Field required",
      "code": "missing"
    }
  ],
  "request_id": "req_0vZ3qR9sT2uV3wX4yZ5aB6"
}
```


Error codes

400 BAD REQUEST

invalid_request The request is not valid	The body or a query parameter can't be understood. **What to do:** Check the request against the reference: a body that isn't JSON, or a parameter of the wrong kind.

401 NOT AUTHENTICATED

authentication_required Authentication required	No API key was sent. Send Authorization: Bearer <key>. **What to do:** Send your key: Authorization: Bearer vdx_test_… or vdx_live_….
invalid_api_key Invalid API key	The key is malformed, unknown, or belongs to another environment. **What to do:** Check you copied the whole key, and that it is a test key for test mode and a live key for live mode.
api_key_revoked API key revoked	The key was revoked. Create or roll a key in the Developers console. **What to do:** Use another key, or create one in the Developers console.
api_key_expired API key expired	The key passed its expiry date. **What to do:** Roll the key, or create a new one, in the Developers console.

402 PAYMENT REQUIRED

insufficient_credits Not enough credits	The balance can't cover this operation. Buy credits or enable auto-recharge.

	What to do: Buy credits (or turn on auto-recharge), then retry. required and available say how many you need.

403 FORBIDDEN

`ip_not_allowed` Address not allowed	The key has an IP allowlist and this address is not on it. **What to do:** Call from an address on the key's allowlist, or add this address to it.
`secret_key_in_browser` Secret key used from a browser	API keys are secret and must only be used from your server, never from web pages or apps. **What to do:** Move the call to your server. Keys must never be in web pages or apps.
`insufficient_scope` Key not permitted	The key's permissions don't include this operation. **What to do:** Use a key with the permission in required_scope, or widen this key's permissions.
`live_mode_not_enabled` Live mode not enabled	Your account can use test keys until Vindex approves live access. **What to do:** Keep building with your test key; apply for live access in the console.
`account_suspended` Account suspended	API access is suspended for this account. Contact Vindex. **What to do:** Contact Vindex.
`feature_not_enabled` Not available yet	This operation is not enabled for your account. **What to do:** The detail says what isn't available yet. It works in test mode.
`usage_limit_reached` Monthly usage limit reached	Your own monthly usage limit stops further paid calls. **What to do:** An owner or admin can raise your monthly usage limit in the console.

404 NOT FOUND

`not_found` Not found	No such object exists in this environment. **What to do:** Check the id, and that you are using a key of the same mode the object was created in.

409 CONFLICT

`report_not_ready` Report not ready	This report exists only after a decision, or is still being generated. **What to do:** Wait for the report.ready webhook, then download.
`invalid_state` Not possible now	The object is in a state that doesn't allow this. **What to do:** Read the object: the detail says why this isn't possible in its current status.
`idempotency_key_in_use` Request already in progress	Another request with this Idempotency-Key is still running. **What to do:** Wait a second (Retry-After) and retry with the same key.

422 UNPROCESSABLE

`validation_failed` Some fields are not valid	One or more fields failed validation; see errors[]. **What to do:** Fix the fields listed in errors[]; each has a pointer to the field and what is wrong.
`idempotency_key_reused` Idempotency-Key reused	This Idempotency-Key was used with different parameters. **What to do:** Use a new Idempotency-Key for a new request.

429 TOO MANY REQUESTS

`rate_limited` Too many requests	Slow down; retry after the number of seconds in Retry-After. **What to do:** Wait the number of seconds in Retry-After, then retry. Spread bursts out.

500 SERVER ERROR

`internal_error` Something went wrong	An unexpected error. Retry with the same Idempotency-Key. **What to do:** Retry with the same Idempotency-Key. If it keeps happening, send Vindex the request_id.

503 SERVICE UNAVAILABLE

`provider_unavailable` Service temporarily unavailable	A screening provider is unavailable; nothing was charged. **What to do:** Nothing was charged. Retry later with the same Idempotency-Key.

Keys, limits and conventions →
API keys, test and live mode, versions, request ids, rate limits, safe retries, paging and metadata.

API reference →
Every endpoint, field, error and webhook event, with code in curl, Node and Python.

CHAPTER 10

Go live

When your integration works in test mode, apply for live access. Vindex checks your company, and once approved you create live keys, buy credits and switch.

Apply for live access

1. In the Developers console, open **Settings** and choose **Go live**.
2. Tell Vindex about your company: legal name, registration number, country, website, what you will use Vindex for, and the volumes you expect. Say whether you are regulated, and by whom.
3. Give the contacts your customers will see on Vindex's consent screen: your **privacy notice** and your **grievance contact**.
4. Accept the partner terms and the **data processing agreement**, and confirm you will use Vindex only for lawful customer-due-diligence checks.

Vindex reviews the application and emails your owners and admins with the result. If something is missing you'll get a note saying what; fix it and apply again.

Before you switch

- **A live key, kept secret**
  Created by an owner or admin, stored in your secrets manager, with an IP allowlist for your servers.
- **A live webhook endpoint**
  https on port 443, verifying every signature, answering 2xx quickly and queueing the work.
- **Duplicates and order handled**
  Skip a webhook-id you've seen; act on the object's current status, not the event's order.
- **Idempotency keys on every POST**
  So a retried create never makes a second verification or a second charge.
- **402 and 429 handled**
  Out of credits: alert your team, don't loop. Rate limited: wait for Retry-After.
- **Credits bought**
  Enough for your first weeks, low-balance alerts set, and auto-recharge if your card allows it.
- **Your brand and notice details**
  Logo, colour, support email, privacy notice and grievance contact: your customers see them.
- **Someone watching**
  Request log and webhook health in the console; Request-Id logged on your side for support.

Then swap your test key for the live one in your server's configuration. Test data stays in test mode: create live verifications afresh.

Your responsibilities

The decision is yours	Vindex's outcome and risk rating are a recommendation from its review. Whether to onboard a customer stays your decision, and your regulator's rules still apply to you. Record it with `partner_decision` so your audit trail shows who decided.
Data protection	You collect your customers' data and decide why; Vindex processes it for you under the data processing agreement (in India's Digital Personal Data Protection Act terms, you are the data fiduciary and Vindex your processor). Keep reports as securely as your other customer records.
Screening results	Use them only for compliance checks, keep them within your compliance team, and never tell the person screened that they matched a list.
Your keys	Anyone with a live key can spend your credits. Revoke a key the moment you think it has leaked.

At launch

Two features open after approvals Vindex is waiting for: **live screening through the API** (from LSEG, the screening data provider) and **DigiLocker in the hosted flow** (from NeGD, which runs DigiLocker). Until then live screening returns `403 feature_not_enabled`, and customers upload their documents instead of fetching them from DigiLocker. Verifications work fully, screening included.

Credits and billing →
How prepaid credits work: prices, holds and charges, buying, auto-recharge, expiry and refunds.

Webhooks →
Get told when things happen: signed deliveries, retries, safe handling and every event.

CHAPTER 11

API reference

Every endpoint of the Vindex API, with its fields, what it returns, the errors it can give, and code you can paste. Signed in? Try any of them in test mode.

BASE URL	AUTHENTICATION	VERSION
`https://api.vindexlegalrisk.com/v1`	`Authorization: Bearer vdx_test_…`	`2026-11-05`

ACCOUNT

- `GET` /me

REPORTS

- `GET` /verifications/{id}/report

BALANCE

- `GET` /balance

TEST HELPERS

- `POST` /test_helpers/verifications/{id}/submit
- `POST` /test_helpers/verifications/{id}/request_action
- `POST` /test_helpers/verifications/{id}/respond
- `POST` /test_helpers/verifications/{id}/decide
- `POST` /test_helpers/verifications/{id}/expire

VERIFICATIONS

- `POST` /verifications
- `GET` /verifications
- `GET` /verifications/{id}
- `POST` /verifications/{id}/cancel
- `POST` /verifications/{id}/hosted_link
- `POST` /verifications/{id}/partner_decision

SCREENINGS

- `POST` /screenings
- `GET` /screenings
- `GET` /screenings/{id}
- `POST` /screening_batches
- `GET` /screening_batches/{id}

EVENTS

- `GET` /events
- `GET` /events/{id}

Errors are **problem details** with a stable `code`; there are 22 of them. Every POST takes an **Idempotency-Key**. Keys, limits and paging are explained in **Keys, limits and conventions**.

Account

Who you are calling as.

GET /v1/me

Check your key

Which partner, environment and key a request authenticates as. Free; handy for testing your setup.

Read limit

RETURNS

200 A **key info** object.

Show its fields

api_version	string	REQUIRED
key	object	REQUIRED
livemode	boolean	REQUIRED
object	string	REQUIRED
partner	object	REQUIRED

ERRORS

400 Bad request · 401 Not authenticated · 402 Not enough credits · 403 Not allowed · 404 Not found · 409 Conflict · 422 Invalid fields · 429 Too many requests

curl

```
curl https://api.vindexlegalrisk.com/v1/me \
  -H "Authorization: Bearer $VINDEX_API_KEY"
```

Node

```
const response = await fetch("https://api.vindexlegalrisk.com/v1/me", {
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
  },
});
const me = await response.json();
```

Python

```python
import os
import requests

response = requests.get(
    "https://api.vindexlegalrisk.com/v1/me",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
    },
    timeout=30,
)
response.raise_for_status()
me = response.json()
```

Response


```
{
  "object": "api_key_info",
  "livemode": false,
  "api_version": "2026-11-05",
  "partner": {
    "id": "ptn_0vZ3aB4cD5eF6gH7iJ8kL9",
    "name": "Acme Lending",
    "status": "live"
  },
  "key": {
    "id": "key_0vZ3aB4cD5eF6gH7iJ8kM1",
    "name": "Production server",
    "scopes": [
      "*"
    ]
  }
}
```


Verifications

Verify a person (KYC) or a company (KYB) through Vindex's hosted flow.

`POST` `/v1/verifications`

Create a verification

Start verifying a person or a company. Returns the verification with a **hosted link** to send to your customer; they complete Vindex's flow (documents, consent, face check) there.

The verification's price is **held** from your credits now and **charged when the customer submits**. If the link expires or you cancel first, the hold is released and nothing is charged.

In test mode nothing reaches Vindex's reviewers. The outcome comes from `test_scenario`, or from the last name (or company name) ending in `Approved`, `Conditional`, `Rejected` or `ActionRequired`; anything else is approved.

Key permission `verifications:write` | Write limit

BODY

type enum REQUIRED

Verify a person (KYC) or a company (KYB).

One of `individual` `company`

company object NULLABLE

Required when type is company.

country string REQUIRED

Country of incorporation (ISO alpha-2)

two-letter country code

legal_name string REQUIRED

1–200 characters

address object NULLABLE

city string NULLABLE

up to 100 characters

country string NULLABLE

ISO 3166-1 alpha-2, e.g. IN

two-letter country code

line1 string NULLABLE

up to 200 characters

line2 string NULLABLE

up to 200 characters

postal_code string NULLABLE

up to 20 characters

state string NULLABLE

up to 100 characters

contact object NULLABLE

The person who will complete the hosted flow

designation string NULLABLE

up to 100 characters

email string NULLABLE

up to 254 characters

name string NULLABLE

up to 200 characters

phone string NULLABLE

up to 32 characters

registration_number string NULLABLE

CIN, company number or equivalent

up to 64 characters

website string NULLABLE

up to 300 characters

external_reference string NULLABLE

Your id for this customer. Searchable; not unique.

up to 255 characters

individual object NULLABLE

Required when type is individual.

first_name string REQUIRED

1–100 characters

last_name string REQUIRED

1–100 characters

address object NULLABLE

city string NULLABLE

up to 100 characters

country string NULLABLE

ISO 3166-1 alpha-2, e.g. IN

two-letter country code

line1 string NULLABLE

up to 200 characters

line2 string NULLABLE

up to 200 characters

postal_code string NULLABLE

up to 20 characters

state string NULLABLE

up to 100 characters

company_name string NULLABLE

Employer or company they act for

up to 200 characters

country string NULLABLE

Country of residence (ISO alpha-2)

two-letter country code

date_of_birth date NULLABLE

designation string NULLABLE

Their role at that company

up to 100 characters

email string NULLABLE

up to 254 characters

middle_name string NULLABLE

up to 100 characters

nationality string NULLABLE

two-letter country code

phone string NULLABLE

E.164 if possible, e.g. +919876543210

up to 32 characters

link_expires_in integer NULLABLE

Seconds until the hosted link expires. Default 7 days.

3,600 to 2,592,000

locale string NULLABLE

format ^[a-z]{2}(-[A-Z]{2})?$

metadata map of string NULLABLE

Up to 50 keys of your own. No personal data.

redirect_url string NULLABLE

Where the customer goes when they finish (https).

up to 2048 characters

test_scenario enum NULLABLE

Test mode only: the outcome to simulate after the customer submits.

One of `approve` `approve_with_conditions` `reject` `action_required`

RETURNS

201 A **verification** object.

Show its fields

action_required_at string REQUIRED NULLABLE

canceled_at string REQUIRED NULLABLE

case_reference string REQUIRED NULLABLE

Vindex's case reference; quote it to Vindex support

conditions array of string REQUIRED

Conditions attached to an approval with conditions

created_at string REQUIRED NULLABLE

credits object REQUIRED

charged integer REQUIRED

held integer REQUIRED

Reserved now; charged when the customer submits

price integer REQUIRED

What the verification costs, in credits

decided_at string REQUIRED NULLABLE

expired_at string REQUIRED NULLABLE

external_reference string REQUIRED NULLABLE

hosted_link object REQUIRED

active boolean REQUIRED

expires_at string REQUIRED NULLABLE

url string REQUIRED NULLABLE

Only returned when a link is created. Send it to your customer; don't store it.

id string REQUIRED

livemode boolean REQUIRED

locale string REQUIRED NULLABLE

metadata map of string REQUIRED

object string REQUIRED

outcome enum REQUIRED NULLABLE

Vindex's recommendation once decided

One of `approved` `approved_with_conditions` `rejected`

partner_decision object REQUIRED NULLABLE

decided_at string REQUIRED

decision enum REQUIRED

One of `accepted` `rejected`

decided_by string NULLABLE

note string NULLABLE

redirect_url string REQUIRED NULLABLE

reports array of objects REQUIRED

available boolean REQUIRED

type enum REQUIRED

One of `outcome_letter` `customer_report`

generated_at string NULLABLE

risk_rating string REQUIRED NULLABLE

low, medium, high (or Vindex's rating)

stage string REQUIRED

Finer progress for display: awaiting_customer, customer_in_progress, screening, under_review, final_review, awaiting_customer_response, appeal_review, decided

started_at string REQUIRED NULLABLE

status enum REQUIRED

One of `created` `in_progress` `submitted` `action_required` `decided` `expired` `canceled`

submitted_at string REQUIRED NULLABLE

type enum REQUIRED

One of `individual` `company`

updated_at string REQUIRED NULLABLE

ERRORS

400 Bad request | 401 Not authenticated | 402 Not enough credits | 403 Not allowed | 404 Not found | 409 Conflict | 422 Invalid fields | 429 Too many requests

curl

```
curl https://api.vindexlegalrisk.com/v1/verifications \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -H "Idempotency-Key: $(uuidgen)" \
  -H "Content-Type: application/json" \
  -d '{
    "type": "individual",
    "external_reference": "crm-42",
    "individual": {
      "first_name": "Aarav",
      "last_name": "Sharma",
      "email": "aarav.sharma@example.com",
      "country": "IN",
      "date_of_birth": "1990-04-12"
    },
    "redirect_url": "https://acme.example/kyc/done",
    "metadata": {
      "plan": "gold"
    }
  }'
```

Node

```
const response = await fetch("https://api.vindexlegalrisk.com/v1/verifications", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
    "Idempotency-Key": crypto.randomUUID(),
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    type: "individual",
    external_reference: "crm-42",
    individual: {
      first_name: "Aarav",
      last_name: "Sharma",
      email: "aarav.sharma@example.com",
      country: "IN",
      date_of_birth: "1990-04-12",
    },
    redirect_url: "https://acme.example/kyc/done",
    metadata: {
      plan: "gold",
    },
  }),
});
const verification = await response.json();
```

Python

```
import os
import uuid
import requests

response = requests.post(
    "https://api.vindexlegalrisk.com/v1/verifications",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
        "Idempotency-Key": str(uuid.uuid4()),
    },
    json={
        "type": "individual",
        "external_reference": "crm-42",
        "individual": {
            "first_name": "Aarav",
            "last_name": "Sharma",
            "email": "aarav.sharma@example.com",
            "country": "IN",
            "date_of_birth": "1990-04-12",
        },
        "redirect_url": "https://acme.example/kyc/done",
        "metadata": {
            "plan": "gold",
        },
    },
    timeout=30,
)
```

```
response.raise_for_status()
verification = response.json()
```

Response


```
{
  "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "verification",
  "livemode": false,
  "type": "individual",
  "status": "created",
  "stage": "awaiting_customer",
  "outcome": null,
  "risk_rating": null,
  "conditions": [],
  "external_reference": "crm-42",
  "metadata": {
    "plan": "gold"
  },
  "case_reference": null,
  "hosted_link": {
    "url": "https://vindexlegalrisk.com/verify/#vlt_x8Qm2Lr0Hc5Wk9Ts3Ny7Vb1Pd4Zf6Ga",
    "expires_at": "2026-11-12T10:24:51Z",
    "active": true
  },
  "redirect_url": "https://acme.example/kyc/done",
  "locale": "en",
  "credits": {
    "price": 70,
    "held": 70,
    "charged": 0
  },
  "reports": [
    {
      "type": "outcome_letter",
      "available": false,
      "generated_at": null
    },
    {
      "type": "customer_report",
      "available": false,
      "generated_at": null
    }
  ],
  "partner_decision": null,
  "created_at": "2026-11-05T10:24:51Z",
  "started_at": null,
  "submitted_at": null,
  "action_required_at": null,
  "decided_at": null,
  "expired_at": null,
  "canceled_at": null,
  "updated_at": "2026-11-05T10:24:51Z"
}
```

`GET` `/v1/verifications`

List verifications

Newest first. Page with `starting_after` set to the last id you received while `has_more` is true.

Key permission `verifications:read` Read limit

QUERY PARAMETERS

`limit` `integer`

1 to 100 · default 10

`starting_after` `string` NULLABLE

Cursor: the last id of the previous page

`status` `enum` NULLABLE

Only this status

One of `created` `in_progress` `submitted` `action_required` `decided` `expired` `canceled`

`external_reference` `string` NULLABLE

`created_gte` `date-time` NULLABLE

Created at or after (ISO 8601)

`created_lt` `date-time` NULLABLE

Created before (ISO 8601)

RETURNS

`200` A **verification list** object.

Show its fields

`data` `array of objects` REQUIRED

`action_required_at` `string` REQUIRED NULLABLE

`canceled_at` `string` REQUIRED NULLABLE

`case_reference` `string` REQUIRED NULLABLE

Vindex's case reference; quote it to Vindex support

`conditions` `array of string` REQUIRED

Conditions attached to an approval with conditions

`created_at` `string` REQUIRED NULLABLE

credits object REQUIRED

charged integer REQUIRED

held integer REQUIRED

Reserved now; charged when the customer submits

price integer REQUIRED

What the verification costs, in credits

decided_at string REQUIRED NULLABLE

expired_at string REQUIRED NULLABLE

external_reference string REQUIRED NULLABLE

hosted_link object REQUIRED

active boolean REQUIRED

expires_at string REQUIRED NULLABLE

url string REQUIRED NULLABLE

Only returned when a link is created. Send it to your customer; don't store it.

id string REQUIRED

livemode boolean REQUIRED

locale string REQUIRED NULLABLE

metadata map of string REQUIRED

object string REQUIRED

outcome enum REQUIRED NULLABLE

Vindex's recommendation once decided

One of `approved` `approved_with_conditions` `rejected`

partner_decision object REQUIRED NULLABLE

decided_at string REQUIRED

decision enum REQUIRED

One of `accepted` `rejected`

decided_by string NULLABLE

note string NULLABLE

redirect_url string REQUIRED NULLABLE

reports array of objects REQUIRED

available boolean REQUIRED

type enum REQUIRED

One of `outcome_letter` `customer_report`

generated_at string NULLABLE

risk_rating string REQUIRED NULLABLE

low, medium, high (or Vindex's rating)

stage string REQUIRED

Finer progress for display: awaiting_customer, customer_in_progress, screening, under_review, final_review, awaiting_customer_response, appeal_review, decided

started_at string REQUIRED NULLABLE

status enum REQUIRED

One of `created` `in_progress` `submitted` `action_required` `decided` `expired` `canceled`

submitted_at string REQUIRED NULLABLE

type enum REQUIRED

One of `individual` `company`

updated_at string REQUIRED NULLABLE

has_more boolean REQUIRED

object string REQUIRED

url string REQUIRED

ERRORS

`400` Bad request `401` Not authenticated `402` Not enough credits `403` Not allowed `404` Not found `409` Conflict

422 Invalid fields 429 Too many requests

curl

```
curl -G https://api.vindexlegalrisk.com/v1/verifications \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -d limit=10 \
  -d status=decided
```

Node

```
const response = await fetch("https://api.vindexlegalrisk.com/v1/verifications?limit=10&status=decided", {
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
  },
});
const page = await response.json();
```

Python

```
import os
import requests

response = requests.get(
    "https://api.vindexlegalrisk.com/v1/verifications",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
    },
    params={
        "limit": "10",
        "status": "decided",
    },
    timeout=30,
)
response.raise_for_status()
page = response.json()
```

Response


```
{
  "object": "list",
  "data": [
    {
      "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
      "object": "verification",
      "livemode": false,
      "type": "individual",
      "status": "decided",
      "stage": "decided",
      "outcome": "approved",
      "risk_rating": "low",
      "conditions": [],
      "external_reference": "crm-42",
      "metadata": {
        "plan": "gold"
      },
```

```
      "case_reference": "KYC-7F3A9C21",
      "hosted_link": {
        "url": null,
        "expires_at": null,
        "active": false
      },
      "redirect_url": "https://acme.example/kyc/done",
      "locale": "en",
      "credits": {
        "price": 70,
        "held": 0,
        "charged": 70
      },
      "reports": [
        {
          "type": "outcome_letter",
          "available": true,
          "generated_at": "2026-11-07T15:02:11Z"
        },
        {
          "type": "customer_report",
          "available": true,
          "generated_at": "2026-11-07T15:02:09Z"
        }
      ],
      "partner_decision": null,
      "created_at": "2026-11-05T10:24:51Z",
      "started_at": "2026-11-05T11:02:40Z",
      "submitted_at": "2026-11-05T11:19:03Z",
      "action_required_at": null,
      "decided_at": "2026-11-07T15:01:58Z",
      "expired_at": null,
      "canceled_at": null,
      "updated_at": "2026-11-07T15:02:11Z"
    }
  ],
  "has_more": false,
  "url": "/v1/verifications"
}
```


GET `/v1/verifications/{verification_id}`

Retrieve a verification

The current status, stage, outcome and risk rating.

Key permission `verifications:read` | Read limit

PATH PARAMETERS

`verification_id` `string` REQUIRED

The object's id

RETURNS

`200` A **verification** object.

Its fields are listed under Create a verification.

ERRORS

`400` Bad request | `401` Not authenticated | `402` Not enough credits | `403` Not allowed | `404` Not found | `409` Conflict | `422` Invalid fields | `429` Too many requests

curl

```
curl https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy \
  -H "Authorization: Bearer $VINDEX_API_KEY"
```

Node

```
const response = await fetch("https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
{
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
  },
});
const verification = await response.json();
```

Python

```
import os
import requests

response = requests.get(
    "https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
    },
    timeout=30,
)
response.raise_for_status()
verification = response.json()
```

Response


```
{
  "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "verification",
  "livemode": false,
  "type": "individual",
  "status": "decided",
  "stage": "decided",
  "outcome": "approved",
  "risk_rating": "low",
  "conditions": [],
  "external_reference": "crm-42",
  "metadata": {
```

```
    "plan": "gold"
  },
  "case_reference": "KYC-7F3A9C21",
  "hosted_link": {
    "url": null,
    "expires_at": null,
    "active": false
  },
  "redirect_url": "https://acme.example/kyc/done",
  "locale": "en",
  "credits": {
    "price": 70,
    "held": 0,
    "charged": 70
  },
  "reports": [
    {
      "type": "outcome_letter",
      "available": true,
      "generated_at": "2026-11-07T15:02:11Z"
    },
    {
      "type": "customer_report",
      "available": true,
      "generated_at": "2026-11-07T15:02:09Z"
    }
  ],
  "partner_decision": null,
  "created_at": "2026-11-05T10:24:51Z",
  "started_at": "2026-11-05T11:02:40Z",
  "submitted_at": "2026-11-05T11:19:03Z",
  "action_required_at": null,
  "decided_at": "2026-11-07T15:01:58Z",
  "expired_at": null,
  "canceled_at": null,
  "updated_at": "2026-11-07T15:02:11Z"
}
```


POST `/v1/verifications/{verification_id}/cancel`

Cancel a verification

Cancel before the customer submits. The link stops working and held credits are released.

Key permission `verifications:write` | Write limit

PATH PARAMETERS

`verification_id` `string` REQUIRED

The object's id

RETURNS

`200` A **verification** object.

Its fields are listed under Create a verification.

ERRORS

`400` Bad request | `401` Not authenticated | `402` Not enough credits | `403` Not allowed | `404` Not found | `409` Conflict | `422` Invalid fields | `429` Too many requests

curl

```
curl https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/cancel \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -H "Idempotency-Key: $(uuidgen)"
```

Node

```
const response = await
fetch("https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/cancel", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
    "Idempotency-Key": crypto.randomUUID(),
  },
});
const verification = await response.json();
```

Python

```
import os
import uuid
import requests

response = requests.post(
    "https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/cancel",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
        "Idempotency-Key": str(uuid.uuid4()),
    },
    timeout=30,
)
response.raise_for_status()
verification = response.json()
```

Response

```
{
  "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "verification",
  "livemode": false,
  "type": "individual",
  "status": "canceled",
  "stage": "canceled",
  "outcome": null,
```

```
  "risk_rating": null,
  "conditions": [],
  "external_reference": "crm-42",
  "metadata": {
    "plan": "gold"
  },
  "case_reference": null,
  "hosted_link": {
    "url": null,
    "expires_at": null,
    "active": false
  },
  "redirect_url": "https://acme.example/kyc/done",
  "locale": "en",
  "credits": {
    "price": 70,
    "held": 0,
    "charged": 0
  },
  "reports": [
    {
      "type": "outcome_letter",
      "available": false,
      "generated_at": null
    },
    {
      "type": "customer_report",
      "available": false,
      "generated_at": null
    }
  ],
  "partner_decision": null,
  "created_at": "2026-11-05T10:24:51Z",
  "started_at": null,
  "submitted_at": null,
  "action_required_at": null,
  "decided_at": null,
  "expired_at": null,
  "canceled_at": "2026-11-05T12:00:00Z",
  "updated_at": "2026-11-05T10:24:51Z"
}
```


`POST` **`/v1/verifications/{verification_id}/hosted_link`**

Create a new hosted link

Issue a fresh link (the previous one stops working). Use it when the link expired, or when Vindex needs more from the customer (status `action_required`).

Key permission `verifications:write` | Write limit

PATH PARAMETERS

verification_id `string` REQUIRED

The object's id

BODY

link_expires_in `integer` NULLABLE

3,600 to 2,592,000

RETURNS

200 A **verification** object.

Its fields are listed under Create a verification.

ERRORS

400 Bad request | 401 Not authenticated | 402 Not enough credits | 403 Not allowed | 404 Not found | 409 Conflict | 422 Invalid fields | 429 Too many requests

curl

```
curl https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/hosted_link \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -H "Idempotency-Key: $(uuidgen)" \
  -H "Content-Type: application/json" \
  -d '{
    "link_expires_in": 259200
  }'
```

Node

```
const response = await
fetch("https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/hosted_link", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
    "Idempotency-Key": crypto.randomUUID(),
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    link_expires_in: 259200,
  }),
});
const verification = await response.json();
```

Python

```python
import os
import uuid
import requests

response = requests.post(
    "https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/hosted_link",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
        "Idempotency-Key": str(uuid.uuid4()),
    },
    json={
        "link_expires_in": 259200,
    },
    timeout=30,
)
response.raise_for_status()
verification = response.json()
```

Response


```
{
  "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "verification",
  "livemode": false,
  "type": "individual",
  "status": "created",
  "stage": "awaiting_customer",
  "outcome": null,
  "risk_rating": null,
  "conditions": [],
  "external_reference": "crm-42",
  "metadata": {
    "plan": "gold"
  },
  "case_reference": null,
  "hosted_link": {
    "url": "https://vindexlegalrisk.com/verify/#vlt_x8Qm2Lr0Hc5Wk9Ts3Ny7Vb1Pd4Zf6Ga",
    "expires_at": "2026-11-12T10:24:51Z",
    "active": true
  },
  "redirect_url": "https://acme.example/kyc/done",
  "locale": "en",
  "credits": {
    "price": 70,
    "held": 70,
    "charged": 0
  },
  "reports": [
    {
      "type": "outcome_letter",
      "available": false,
      "generated_at": null
    },
    {
      "type": "customer_report",
```

```
      "available": false,
      "generated_at": null
    }
  ],
  "partner_decision": null,
  "created_at": "2026-11-05T10:24:51Z",
  "started_at": null,
  "submitted_at": null,
  "action_required_at": null,
  "decided_at": null,
  "expired_at": null,
  "canceled_at": null,
  "updated_at": "2026-11-05T10:24:51Z"
}
```


POST `/v1/verifications/{verification_id}/partner_decision`

Record your decision

Vindex's outcome is a recommendation; the decision to onboard stays yours. Record it here so the audit trail shows who decided what, and when.

Key permission `verifications:write` | Write limit

PATH PARAMETERS

verification_id string REQUIRED

The object's id

BODY

decision enum REQUIRED

What you decided after Vindex's outcome.

One of `accepted` `rejected`

decided_by string NULLABLE

Who decided (name or staff id).

up to 200 characters

note string NULLABLE

up to 2000 characters

RETURNS

200 A **verification** object.

Its fields are listed under Create a verification.

ERRORS

400 Bad request | 401 Not authenticated | 402 Not enough credits | 403 Not allowed | 404 Not found | 409 Conflict | 422 Invalid fields | 429 Too many requests

curl

```
curl https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/partner_decision \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -H "Idempotency-Key: $(uuidgen)" \
  -H "Content-Type: application/json" \
  -d '{
    "decision": "accepted",
    "decided_by": "Priya Nair, onboarding team",
    "note": "Onboard on the standard plan."
  }'
```

Node

```
const response = await
fetch("https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/partner_decision", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
    "Idempotency-Key": crypto.randomUUID(),
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    decision: "accepted",
    decided_by: "Priya Nair, onboarding team",
    note: "Onboard on the standard plan.",
  }),
});
const verification = await response.json();
```

Python

```python
import os
import uuid
import requests

response = requests.post(
    "https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/partner_decision",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
        "Idempotency-Key": str(uuid.uuid4()),
    },
    json={
        "decision": "accepted",
        "decided_by": "Priya Nair, onboarding team",
        "note": "Onboard on the standard plan.",
    },
    timeout=30,
)
response.raise_for_status()
verification = response.json()
```

Response

```
{
  "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "verification",
  "livemode": false,
  "type": "individual",
  "status": "decided",
  "stage": "decided",
  "outcome": "approved",
  "risk_rating": "low",
  "conditions": [],
  "external_reference": "crm-42",
  "metadata": {
    "plan": "gold"
  },
  "case_reference": "KYC-7F3A9C21",
  "hosted_link": {
    "url": null,
    "expires_at": null,
    "active": false
  },
  "redirect_url": "https://acme.example/kyc/done",
  "locale": "en",
  "credits": {
    "price": 70,
    "held": 0,
    "charged": 70
  },
  "reports": [
    {
      "type": "outcome_letter",
      "available": true,
      "generated_at": "2026-11-07T15:02:11Z"
    },
```

```
    {
      "type": "customer_report",
      "available": true,
      "generated_at": "2026-11-07T15:02:09Z"
    }
  ],
  "partner_decision": {
    "decision": "accepted",
    "decided_by": "Priya Nair, onboarding team",
    "note": "Onboard on the standard plan.",
    "decided_at": "2026-11-07T16:10:00Z"
  },
  "created_at": "2026-11-05T10:24:51Z",
  "started_at": "2026-11-05T11:02:40Z",
  "submitted_at": "2026-11-05T11:19:03Z",
  "action_required_at": null,
  "decided_at": "2026-11-07T15:01:58Z",
  "expired_at": null,
  "canceled_at": null,
  "updated_at": "2026-11-07T15:02:11Z"
}
```


Reports

Download the outcome letter and customer report of a decided verification.

`GET` **`/v1/verifications/{verification_id}/report`**

Download a report

The PDF of a decided verification. `outcome_letter` exists for every decision; `customer_report` for approvals. Store it securely: it contains personal data.

Key permission `verifications:read` | Read limit

PATH PARAMETERS

`verification_id` `string` REQUIRED

The object's id

QUERY PARAMETERS

`type` `enum`

Which report: the outcome letter (every decision) or the customer report (approvals)

One of `outcome_letter` `customer_report`

default "outcome_letter"

RETURNS

200 The PDF file (application/pdf).

ERRORS

400 Bad request | 401 Not authenticated | 402 Not enough credits | 403 Not allowed | 404 Not found | 409 Conflict | 422 Invalid fields | 429 Too many requests

curl

```
curl -G https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/report \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -d type=outcome_letter \
  -o outcome-letter.pdf
```

Node

```
const response = await
fetch("https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/report?
type=outcome_letter", {
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
  },
});
const pdf = Buffer.from(await response.arrayBuffer());
```

Python

```
import os
import requests

response = requests.get(
    "https://api.vindexlegalrisk.com/v1/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/report",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
    },
    params={
        "type": "outcome_letter",
    },
    timeout=30,
)
response.raise_for_status()
pdf = response.content
```

Response

```
200 OK
Content-Type: application/pdf
Content-Disposition: attachment; filename="KYC-7F3A9C21 - Outcome letter.pdf"

%PDF-1.7 …
```

Screenings

Screen people and companies against sanctions, PEP and adverse-media lists.

`POST` **`/v1/screenings`**

Screen a name

Screen one person or company and get the result straight away. Charged only when the screen completes.

In test mode, names containing `Sanctioned`, `Pep`, `Adverse` or `Multimatch` return matches and `Unavailable` simulates a provider outage; anything else is `no_match`.

Key permission `screenings:write` | Screening limit

BODY

`subject` `object` REQUIRED

`type` `enum` REQUIRED

One of `person` `company`

`country` `string` NULLABLE

two-letter country code

`date_of_birth` `date` NULLABLE

People only; narrows namesake matches

`first_name` `string` NULLABLE

up to 100 characters

`gender` `enum` NULLABLE

One of `male` `female` `unspecified`

`last_name` `string` NULLABLE

up to 100 characters

`name` `string` NULLABLE

Full name (or use first_name and last_name)

up to 300 characters

`registration_number` `string` NULLABLE

Companies only

up to 64 characters

external_reference string NULLABLE

up to 255 characters

metadata map of string NULLABLE

purpose enum

Why you are screening (kept for audit).

One of `onboarding` `periodic_review` `transaction_monitoring` `other`

default "onboarding"

RETURNS

201 A **screening** object.

Show its fields

batch_id string REQUIRED NULLABLE

completed_at string REQUIRED NULLABLE

created_at string REQUIRED NULLABLE

credits_charged integer REQUIRED

error object REQUIRED NULLABLE

external_reference string REQUIRED NULLABLE

id string REQUIRED

livemode boolean REQUIRED

metadata map of string REQUIRED

object string REQUIRED

purpose string REQUIRED NULLABLE

result object REQUIRED NULLABLE

outcome enum REQUIRED

One of `no_match` `potential_match`

provider enum REQUIRED

One of `world_check` `test`

screened_at string REQUIRED

attribution string NULLABLE

categories array of enum NULLABLE

match_count integer NULLABLE

status enum REQUIRED

One of `pending` `processing` `completed` `failed`

subject object REQUIRED

ERRORS

400 Bad request | 401 Not authenticated | 402 Not enough credits | 403 Not allowed | 404 Not found | 409 Conflict | 422 Invalid fields | 429 Too many requests | 503 Provider unavailable

curl

```
curl https://api.vindexlegalrisk.com/v1/screenings \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -H "Idempotency-Key: $(uuidgen)" \
  -H "Content-Type: application/json" \
  -d '{
    "subject": {
      "type": "person",
      "name": "Aarav Sharma",
      "date_of_birth": "1990-04-12",
      "country": "IN"
    },
    "purpose": "onboarding",
    "external_reference": "crm-42"
  }'
```

Node

```
const response = await fetch("https://api.vindexlegalrisk.com/v1/screenings", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
    "Idempotency-Key": crypto.randomUUID(),
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    subject: {
      type: "person",
      name: "Aarav Sharma",
      date_of_birth: "1990-04-12",
      country: "IN",
    },
    purpose: "onboarding",
    external_reference: "crm-42",
  }),
});
const screening = await response.json();
```

Python

```
import os
import uuid
import requests

response = requests.post(
    "https://api.vindexlegalrisk.com/v1/screenings",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
        "Idempotency-Key": str(uuid.uuid4()),
    },
    json={
        "subject": {
            "type": "person",
            "name": "Aarav Sharma",
            "date_of_birth": "1990-04-12",
            "country": "IN",
        },
        "purpose": "onboarding",
        "external_reference": "crm-42",
    },
    timeout=30,
)
response.raise_for_status()
screening = response.json()
```

Response


```
{
  "id": "scr_0vZ3mT4cQw8nX2Lb9Pk1Ra",
  "object": "screening",
  "livemode": false,
  "batch_id": null,
  "status": "completed",
  "subject": {
    "type": "person",
    "country": "IN"
  },
  "result": {
    "outcome": "no_match",
    "screened_at": "2026-11-05T10:31:12Z",
    "provider": "test",
    "match_count": 0,
    "categories": []
  },
  "error": null,
  "purpose": "onboarding",
  "external_reference": "crm-42",
  "metadata": {},
  "credits_charged": 1,
  "created_at": "2026-11-05T10:31:11Z",
  "completed_at": "2026-11-05T10:31:12Z"
}
```


GET `/v1/screenings`

List screenings

Key permission `screenings:read` Read limit

QUERY PARAMETERS

`limit` integer

1 to 100 · default 10

`starting_after` string NULLABLE

`batch_id` string NULLABLE

RETURNS

200 A **screening list** object.

Show its fields

`data` array of objects REQUIRED

batch_id string REQUIRED NULLABLE

completed_at string REQUIRED NULLABLE

created_at string REQUIRED NULLABLE

credits_charged integer REQUIRED

error object REQUIRED NULLABLE

external_reference string REQUIRED NULLABLE

id string REQUIRED

livemode boolean REQUIRED

metadata map of string REQUIRED

object string REQUIRED

purpose string REQUIRED NULLABLE

result object REQUIRED NULLABLE

outcome enum REQUIRED

One of `no_match` `potential_match`

provider enum REQUIRED

One of `world_check` `test`

screened_at string REQUIRED

attribution string NULLABLE

categories array of enum NULLABLE

match_count integer NULLABLE

status enum REQUIRED

One of `pending` `processing` `completed` `failed`

subject object REQUIRED

has_more boolean REQUIRED

object string REQUIRED

`url` string REQUIRED

ERRORS

400 Bad request | 401 Not authenticated | 402 Not enough credits | 403 Not allowed | 404 Not found | 409 Conflict | 422 Invalid fields | 429 Too many requests

curl

```
curl -G https://api.vindexlegalrisk.com/v1/screenings \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -d limit=10
```

Node

```
const response = await fetch("https://api.vindexlegalrisk.com/v1/screenings?limit=10", {
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
  },
});
const page = await response.json();
```

Python

```
import os
import requests

response = requests.get(
    "https://api.vindexlegalrisk.com/v1/screenings",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
    },
    params={
        "limit": "10",
    },
    timeout=30,
)
response.raise_for_status()
page = response.json()
```

Response


```
{
  "object": "list",
  "data": [
    {
      "id": "scr_0vZ3mT4cQw8nX2Lb9Pk1Ra",
      "object": "screening",
      "livemode": false,
      "batch_id": null,
      "status": "completed",
      "subject": {
        "type": "person",
        "country": "IN"
```

```
      },
      "result": {
        "outcome": "no_match",
        "screened_at": "2026-11-05T10:31:12Z",
        "provider": "test",
        "match_count": 0,
        "categories": []
      },
      "error": null,
      "purpose": "onboarding",
      "external_reference": "crm-42",
      "metadata": {},
      "credits_charged": 1,
      "created_at": "2026-11-05T10:31:11Z",
      "completed_at": "2026-11-05T10:31:12Z"
    }
  ],
  "has_more": false,
  "url": "/v1/screenings"
}
```


GET `/v1/screenings/{screening_id}`

Retrieve a screening

Key permission `screenings:read` | Read limit

PATH PARAMETERS

`screening_id` `string` REQUIRED

The object's id

RETURNS

`200` A **screening** object.

Its fields are listed under Screen a name.

ERRORS

`400` Bad request | `401` Not authenticated | `402` Not enough credits | `403` Not allowed | `404` Not found | `409` Conflict | `422` Invalid fields | `429` Too many requests

curl

```
curl https://api.vindexlegalrisk.com/v1/screenings/scr_0vZ3mT4cQw8nX2Lb9Pk1Ra \
  -H "Authorization: Bearer $VINDEX_API_KEY"
```

Node

```
const response = await fetch("https://api.vindexlegalrisk.com/v1/screenings/scr_0vZ3mT4cQw8nX2Lb9Pk1Ra", {
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
  },
});
const screening = await response.json();
```

Python

```
import os
import requests

response = requests.get(
    "https://api.vindexlegalrisk.com/v1/screenings/scr_0vZ3mT4cQw8nX2Lb9Pk1Ra",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
    },
    timeout=30,
)
response.raise_for_status()
screening = response.json()
```

Response


```
{
  "id": "scr_0vZ3mT4cQw8nX2Lb9Pk1Ra",
  "object": "screening",
  "livemode": false,
  "batch_id": null,
  "status": "completed",
  "subject": {
    "type": "person",
    "country": "IN"
  },
  "result": {
    "outcome": "no_match",
    "screened_at": "2026-11-05T10:31:12Z",
    "provider": "test",
    "match_count": 0,
    "categories": []
  },
  "error": null,
  "purpose": "onboarding",
  "external_reference": "crm-42",
  "metadata": {},
  "credits_charged": 1,
  "created_at": "2026-11-05T10:31:11Z",
  "completed_at": "2026-11-05T10:31:12Z"
}
```

POST `/v1/screening_batches`

Screen up to 100 names

Screen up to 100 subjects in the background. The total price is held up front and charged per subject screened; the rest is released. You get `screening_batch.completed` when it is done.

Key permission `screenings:write` | Screening limit

BODY

subjects `array of objects` REQUIRED

1 to 100 items

subject `object` REQUIRED

type `enum` REQUIRED

One of `person` `company`

country `string` NULLABLE

two-letter country code

date_of_birth `date` NULLABLE

People only; narrows namesake matches

first_name `string` NULLABLE

up to 100 characters

gender `enum` NULLABLE

One of `male` `female` `unspecified`

last_name `string` NULLABLE

up to 100 characters

name `string` NULLABLE

Full name (or use first_name and last_name)

up to 300 characters

registration_number `string` NULLABLE

Companies only

up to 64 characters

external_reference `string` NULLABLE

up to 255 characters

metadata `map of string` NULLABLE

external_reference string NULLABLE

up to 255 characters

metadata map of string NULLABLE

purpose enum

One of `onboarding` `periodic_review` `transaction_monitoring` `other`

default "onboarding"

RETURNS

202 A **screening batch** object.

Show its fields

completed integer REQUIRED

completed_at string REQUIRED NULLABLE

created_at string REQUIRED NULLABLE

credits object REQUIRED

charged integer REQUIRED

held integer REQUIRED

external_reference string REQUIRED NULLABLE

failed integer REQUIRED

id string REQUIRED

livemode boolean REQUIRED

metadata map of string REQUIRED

object string REQUIRED

status enum REQUIRED

One of `processing` `completed` `failed` `canceled`

total integer REQUIRED

with_matches integer REQUIRED

screenings array of objects NULLABLE

batch_id string REQUIRED NULLABLE

completed_at string REQUIRED NULLABLE

created_at string REQUIRED NULLABLE

credits_charged integer REQUIRED

error object REQUIRED NULLABLE

external_reference string REQUIRED NULLABLE

id string REQUIRED

livemode boolean REQUIRED

metadata map of string REQUIRED

object string REQUIRED

purpose string REQUIRED NULLABLE

result object REQUIRED NULLABLE

outcome enum REQUIRED

One of `no_match` `potential_match`

provider enum REQUIRED

One of `world_check` `test`

screened_at string REQUIRED

attribution string NULLABLE

categories array of enum NULLABLE

match_count integer NULLABLE

status enum REQUIRED

One of `pending` `processing` `completed` `failed`

subject object REQUIRED

ERRORS

`400` Bad request `401` Not authenticated `402` Not enough credits `403` Not allowed `404` Not found `409` Conflict

422 Invalid fields 429 Too many requests

curl

```
curl https://api.vindexlegalrisk.com/v1/screening_batches \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -H "Idempotency-Key: $(uuidgen)" \
  -H "Content-Type: application/json" \
  -d '{
    "subjects": [
      {
        "subject": {
          "type": "person",
          "name": "Aarav Sharma",
          "country": "IN"
        },
        "external_reference": "row-1"
      },
      {
        "subject": {
          "type": "company",
          "name": "Bright Widgets Ltd",
          "country": "GB"
        },
        "external_reference": "row-2"
      }
    ],
    "purpose": "periodic_review",
    "external_reference": "review-2026-11"
  }'
```

Node

```
const response = await fetch("https://api.vindexlegalrisk.com/v1/screening_batches", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
    "Idempotency-Key": crypto.randomUUID(),
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    subjects: [
      {
        subject: {
          type: "person",
          name: "Aarav Sharma",
          country: "IN",
        },
        external_reference: "row-1",
      },
      {
        subject: {
          type: "company",
          name: "Bright Widgets Ltd",
          country: "GB",
        },
```

```
        external_reference: "row-2",
      },
    ],
    purpose: "periodic_review",
    external_reference: "review-2026-11",
  }),
});
const batch = await response.json();
```

Python

```
import os
import uuid
import requests

response = requests.post(
    "https://api.vindexlegalrisk.com/v1/screening_batches",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
        "Idempotency-Key": str(uuid.uuid4()),
    },
    json={
        "subjects": [
            {
                "subject": {
                    "type": "person",
                    "name": "Aarav Sharma",
                    "country": "IN",
                },
                "external_reference": "row-1",
            },
            {
                "subject": {
                    "type": "company",
                    "name": "Bright Widgets Ltd",
                    "country": "GB",
                },
                "external_reference": "row-2",
            },
        ],
        "purpose": "periodic_review",
        "external_reference": "review-2026-11",
    },
    timeout=30,
)
response.raise_for_status()
batch = response.json()
```

Response


```
{
  "id": "sbt_0vZ3nY7dRe1pZ5Mc3Qv8Tb",
  "object": "screening_batch",
  "livemode": false,
  "status": "processing",
  "total": 2,
  "completed": 0,
  "failed": 0,
  "with_matches": 0,
  "credits": {
    "held": 2,
    "charged": 0
  },
  "external_reference": "review-2026-11",
  "metadata": {},
  "created_at": "2026-11-05T10:40:00Z",
  "completed_at": null
}
```


GET `/v1/screening_batches/{batch_id}`

Retrieve a batch

The batch with every screening in it.

Key permission `screenings:read` | Read limit

PATH PARAMETERS

batch_id `string` REQUIRED

The object's id

RETURNS

200 A **screening batch** object.

Its fields are listed under Screen up to 100 names.

ERRORS

400 Bad request | 401 Not authenticated | 402 Not enough credits | 403 Not allowed | 404 Not found | 409 Conflict | 422 Invalid fields | 429 Too many requests

curl

```
curl https://api.vindexlegalrisk.com/v1/screening_batches/sbt_0vZ3nY7dRe1pZ5Mc3Qv8Tb \
  -H "Authorization: Bearer $VINDEX_API_KEY"
```

Node

```
const response = await
fetch("https://api.vindexlegalrisk.com/v1/screening_batches/sbt_0vZ3nY7dRe1pZ5Mc3Qv8Tb", {
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
  },
});
const batch = await response.json();
```

Python

```
import os
import requests

response = requests.get(
    "https://api.vindexlegalrisk.com/v1/screening_batches/sbt_0vZ3nY7dRe1pZ5Mc3Qv8Tb",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
    },
    timeout=30,
)
response.raise_for_status()
batch = response.json()
```

Response


```
{
  "id": "sbt_0vZ3nY7dRe1pZ5Mc3Qv8Tb",
  "object": "screening_batch",
  "livemode": false,
  "status": "completed",
  "total": 2,
  "completed": 2,
  "failed": 0,
  "with_matches": 0,
  "credits": {
    "held": 2,
    "charged": 2
  },
  "external_reference": "review-2026-11",
  "metadata": {},
  "created_at": "2026-11-05T10:40:00Z",
  "completed_at": "2026-11-05T10:40:04Z",
  "screenings": [
    {
      "id": "scr_0vZ3mT4cQw8nX2Lb9Pk1Ra",
      "object": "screening",
      "livemode": false,
      "batch_id": null,
      "status": "completed",
      "subject": {
        "type": "person",
        "country": "IN"
      },
      "result": {
        "outcome": "no_match",
```

```
        "screened_at": "2026-11-05T10:31:12Z",
        "provider": "test",
        "match_count": 0,
        "categories": []
      },
      "error": null,
      "purpose": "onboarding",
      "external_reference": "crm-42",
      "metadata": {},
      "credits_charged": 1,
      "created_at": "2026-11-05T10:31:11Z",
      "completed_at": "2026-11-05T10:31:12Z"
    }
  ]
}
```


Balance

Your prepaid credits.

`GET` **`/v1/balance`**

Retrieve your balance

Key permission `balance:read` | Read limit

RETURNS

`200` A **balance** object.

Show its fields

`available` `integer` REQUIRED

Credits you can spend now

`expiring_soon` `array of object` REQUIRED

Credits expiring within 30 days

`frozen` `integer` REQUIRED

Credits locked by a payment dispute

`held` `integer` REQUIRED

Credits reserved for verifications in progress

`livemode` `boolean` REQUIRED

`low_balance_threshold` `integer` REQUIRED NULLABLE

---

object string REQUIRED

ERRORS

400 Bad request | 401 Not authenticated | 402 Not enough credits | 403 Not allowed | 404 Not found | 409 Conflict | 422 Invalid fields | 429 Too many requests

curl

```
curl https://api.vindexlegalrisk.com/v1/balance \
  -H "Authorization: Bearer $VINDEX_API_KEY"
```

Node

```
const response = await fetch("https://api.vindexlegalrisk.com/v1/balance", {
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
  },
});
const balance = await response.json();
```

Python

```
import os
import requests

response = requests.get(
    "https://api.vindexlegalrisk.com/v1/balance",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
    },
    timeout=30,
)
response.raise_for_status()
balance = response.json()
```

Response

```
{
  "object": "balance",
  "livemode": true,
  "available": 1204,
  "held": 140,
  "frozen": 0,
  "low_balance_threshold": 140,
  "expiring_soon": [
    {
      "credits": 25,
      "expires_at": "2026-11-28T09:00:00Z"
    }
  ]
}
```

Events

What happened in your account, also sent to your webhook endpoints.

`GET` `/v1/events`

List events

Events from the last 30 days, newest first. Use this to catch up after your endpoint was down.

Key permission `events:read` | Read limit

QUERY PARAMETERS

limit integer

1 to 100 · default 10

starting_after string NULLABLE

type string NULLABLE

An event type, or a group such as verification.*

object_id string NULLABLE

Events about one object

RETURNS

200 A **event list** object.

Show its fields

data array of objects REQUIRED

created_at string REQUIRED

data object REQUIRED

id string REQUIRED

livemode boolean REQUIRED

object string REQUIRED

type string REQUIRED

has_more boolean REQUIRED

object string REQUIRED

url string REQUIRED

ERRORS

400 Bad request | 401 Not authenticated | 402 Not enough credits | 403 Not allowed | 404 Not found | 409 Conflict | 422 Invalid fields | 429 Too many requests

curl

```
curl -G https://api.vindexlegalrisk.com/v1/events \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -d limit=10 \
  -d type=verification.*
```

Node

```
const response = await fetch("https://api.vindexlegalrisk.com/v1/events?limit=10&type=verification.*", {
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
  },
});
const page = await response.json();
```

Python

```
import os
import requests

response = requests.get(
    "https://api.vindexlegalrisk.com/v1/events",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
    },
    params={
        "limit": "10",
        "type": "verification.*",
    },
    timeout=30,
)
response.raise_for_status()
page = response.json()
```

Response


```
{
  "object": "list",
  "data": [
    {
      "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
      "object": "event",
      "type": "verification.decided",
      "created_at": "2026-11-07T15:01:59Z",
      "livemode": false,
      "data": {
        "object": "verification",
        "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
        "type": "individual",
        "status": "decided",
        "stage": "decided",
        "outcome": "approved",
        "risk_rating": "low",
        "external_reference": "crm-42"
      }
    }
  ],
  "has_more": false,
  "url": "/v1/events"
}
```


GET `/v1/events/{event_id}`

Retrieve an event

Key permission `events:read` Read limit

PATH PARAMETERS

event_id string REQUIRED

The object's id

RETURNS

200 A **event** object.

Show its fields

created_at string REQUIRED

data object REQUIRED

id string REQUIRED

livemode boolean REQUIRED

object string REQUIRED

type string REQUIRED

ERRORS

400 Bad request | 401 Not authenticated | 402 Not enough credits | 403 Not allowed | 404 Not found | 409 Conflict | 422 Invalid fields | 429 Too many requests

curl

```
curl https://api.vindexlegalrisk.com/v1/events/evt_0vZ3kq9XfJ1mB7c2Lr8sTy \
  -H "Authorization: Bearer $VINDEX_API_KEY"
```

Node

```
const response = await fetch("https://api.vindexlegalrisk.com/v1/events/evt_0vZ3kq9XfJ1mB7c2Lr8sTy", {
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
  },
});
const event = await response.json();
```

Python

```
import os
import requests

response = requests.get(
    "https://api.vindexlegalrisk.com/v1/events/evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
    },
    timeout=30,
)
response.raise_for_status()
event = response.json()
```

Response


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "verification.decided",
  "created_at": "2026-11-07T15:01:59Z",
  "livemode": false,
  "data": {
    "object": "verification",
    "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
    "type": "individual",
    "status": "decided",
    "stage": "decided",
    "outcome": "approved",
    "risk_rating": "low",
    "external_reference": "crm-42"
  }
}
```


Test helpers

Test mode only: move a verification through its life cycle.

POST `/v1/test_helpers/verifications/{verification_id}/submit`

Simulate the customer submitting

Key permission `verifications:write` | Write limit | Test keys only

PATH PARAMETERS

`verification_id` `string` REQUIRED

The object's id

RETURNS

`200` A **verification** object.

Its fields are listed under Create a verification.

ERRORS

`400` Bad request | `401` Not authenticated | `402` Not enough credits | `403` Not allowed | `404` Not found | `409` Conflict | `422` Invalid fields | `429` Too many requests

curl

```
curl https://api.vindexlegalrisk.com/v1/test_helpers/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/submit \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -H "Idempotency-Key: $(uuidgen)"
```

Node

```
const response = await
fetch("https://api.vindexlegalrisk.com/v1/test_helpers/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/submit", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
    "Idempotency-Key": crypto.randomUUID(),
  },
});
const verification = await response.json();
```

Python

```
import os
import uuid
import requests

response = requests.post(
    "https://api.vindexlegalrisk.com/v1/test_helpers/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/submit",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
        "Idempotency-Key": str(uuid.uuid4()),
    },
    timeout=30,
)
response.raise_for_status()
verification = response.json()
```

Response


```
{
  "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "verification",
  "livemode": false,
  "type": "individual",
  "status": "submitted",
  "stage": "under_review",
  "outcome": null,
  "risk_rating": null,
  "conditions": [],
  "external_reference": "crm-42",
  "metadata": {
    "plan": "gold"
  },
  "case_reference": null,
  "hosted_link": {
    "url": null,
    "expires_at": "2026-11-12T10:24:51Z",
    "active": true
```

```
  },
  "redirect_url": "https://acme.example/kyc/done",
  "locale": "en",
  "credits": {
    "price": 70,
    "held": 0,
    "charged": 70
  },
  "reports": [
    {
      "type": "outcome_letter",
      "available": false,
      "generated_at": null
    },
    {
      "type": "customer_report",
      "available": false,
      "generated_at": null
    }
  ],
  "partner_decision": null,
  "created_at": "2026-11-05T10:24:51Z",
  "started_at": null,
  "submitted_at": null,
  "action_required_at": null,
  "decided_at": null,
  "expired_at": null,
  "canceled_at": null,
  "updated_at": "2026-11-05T10:24:51Z"
}
```


POST `/v1/test_helpers/verifications/{verification_id}/request_action`

Simulate Vindex asking the customer for more

Key permission `verifications:write` | Write limit | Test keys only

PATH PARAMETERS

`verification_id` `string` REQUIRED

The object's id

RETURNS

`200` A **verification** object.

Its fields are listed under Create a verification.

ERRORS

`400` Bad request | `401` Not authenticated | `402` Not enough credits | `403` Not allowed | `404` Not found | `409` Conflict

422 Invalid fields 429 Too many requests

curl

```
curl https://api.vindexlegalrisk.com/v1/test_helpers/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/request_action
\
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -H "Idempotency-Key: $(uuidgen)"
```

Node

```
const response = await
fetch("https://api.vindexlegalrisk.com/v1/test_helpers/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/request_acti
on", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
    "Idempotency-Key": crypto.randomUUID(),
  },
});
const verification = await response.json();
```

Python

```
import os
import uuid
import requests

response = requests.post(

"https://api.vindexlegalrisk.com/v1/test_helpers/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/request_action",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
        "Idempotency-Key": str(uuid.uuid4()),
    },
    timeout=30,
)
response.raise_for_status()
verification = response.json()
```

Response

```
{
  "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "verification",
  "livemode": false,
  "type": "individual",
  "status": "action_required",
  "stage": "awaiting_customer_response",
  "outcome": null,
  "risk_rating": null,
  "conditions": [],
  "external_reference": "crm-42",
  "metadata": {
    "plan": "gold"
  },
```

```
  "case_reference": null,
  "hosted_link": {
    "url": "https://vindexlegalrisk.com/verify/#vlt_x8Qm2Lr0Hc5Wk9Ts3Ny7Vb1Pd4Zf6Ga",
    "expires_at": "2026-11-12T10:24:51Z",
    "active": true
  },
  "redirect_url": "https://acme.example/kyc/done",
  "locale": "en",
  "credits": {
    "price": 70,
    "held": 70,
    "charged": 0
  },
  "reports": [
    {
      "type": "outcome_letter",
      "available": false,
      "generated_at": null
    },
    {
      "type": "customer_report",
      "available": false,
      "generated_at": null
    }
  ],
  "partner_decision": null,
  "created_at": "2026-11-05T10:24:51Z",
  "started_at": null,
  "submitted_at": null,
  "action_required_at": null,
  "decided_at": null,
  "expired_at": null,
  "canceled_at": null,
  "updated_at": "2026-11-05T10:24:51Z"
}
```


POST `/v1/test_helpers/verifications/{verification_id}/respond`

Simulate the customer answering

Key permission `verifications:write` | Write limit | Test keys only

PATH PARAMETERS

verification_id `string` REQUIRED

The object's id

RETURNS

200 A **verification** object.

Its fields are listed under Create a verification.

ERRORS

400 Bad request | 401 Not authenticated | 402 Not enough credits | 403 Not allowed | 404 Not found | 409 Conflict | 422 Invalid fields | 429 Too many requests

curl

```
curl https://api.vindexlegalrisk.com/v1/test_helpers/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/respond \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -H "Idempotency-Key: $(uuidgen)"
```

Node

```
const response = await
fetch("https://api.vindexlegalrisk.com/v1/test_helpers/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/respond", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
    "Idempotency-Key": crypto.randomUUID(),
  },
});
const verification = await response.json();
```

Python

```
import os
import uuid
import requests

response = requests.post(
    "https://api.vindexlegalrisk.com/v1/test_helpers/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/respond",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
        "Idempotency-Key": str(uuid.uuid4()),
    },
    timeout=30,
)
response.raise_for_status()
verification = response.json()
```

Response


```
{
  "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "verification",
  "livemode": false,
  "type": "individual",
  "status": "submitted",
  "stage": "under_review",
  "outcome": null,
  "risk_rating": null,
  "conditions": [],
  "external_reference": "crm-42",
  "metadata": {
    "plan": "gold"
```

```
  },
  "case_reference": null,
  "hosted_link": {
    "url": "https://vindexlegalrisk.com/verify/#vlt_x8Qm2Lr0Hc5Wk9Ts3Ny7Vb1Pd4Zf6Ga",
    "expires_at": "2026-11-12T10:24:51Z",
    "active": true
  },
  "redirect_url": "https://acme.example/kyc/done",
  "locale": "en",
  "credits": {
    "price": 70,
    "held": 70,
    "charged": 0
  },
  "reports": [
    {
      "type": "outcome_letter",
      "available": false,
      "generated_at": null
    },
    {
      "type": "customer_report",
      "available": false,
      "generated_at": null
    }
  ],
  "partner_decision": null,
  "created_at": "2026-11-05T10:24:51Z",
  "started_at": null,
  "submitted_at": null,
  "action_required_at": null,
  "decided_at": null,
  "expired_at": null,
  "canceled_at": null,
  "updated_at": "2026-11-05T10:24:51Z"
}
```


POST `/v1/test_helpers/verifications/{verification_id}/decide`

Simulate Vindex's decision

Key permission `verifications:write` | Write limit | Test keys only

PATH PARAMETERS

verification_id `string` REQUIRED

The object's id

BODY

outcome enum REQUIRED

One of `approved` `approved_with_conditions` `rejected`

RETURNS

200 A **verification** object.

Its fields are listed under Create a verification.

ERRORS

400 Bad request | 401 Not authenticated | 402 Not enough credits | 403 Not allowed | 404 Not found | 409 Conflict | 422 Invalid fields | 429 Too many requests

curl

```
curl https://api.vindexlegalrisk.com/v1/test_helpers/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/decide \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -H "Idempotency-Key: $(uuidgen)" \
  -H "Content-Type: application/json" \
  -d '{
    "outcome": "approved"
  }'
```

Node

```
const response = await
fetch("https://api.vindexlegalrisk.com/v1/test_helpers/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/decide", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
    "Idempotency-Key": crypto.randomUUID(),
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    outcome: "approved",
  }),
});
const verification = await response.json();
```

Python

```python
import os
import uuid
import requests

response = requests.post(
    "https://api.vindexlegalrisk.com/v1/test_helpers/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/decide",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
        "Idempotency-Key": str(uuid.uuid4()),
    },
    json={
        "outcome": "approved",
    },
    timeout=30,
)
response.raise_for_status()
verification = response.json()
```

Response


```
{
  "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "verification",
  "livemode": false,
  "type": "individual",
  "status": "decided",
  "stage": "decided",
  "outcome": "approved",
  "risk_rating": "low",
  "conditions": [],
  "external_reference": "crm-42",
  "metadata": {
    "plan": "gold"
  },
  "case_reference": "KYC-7F3A9C21",
  "hosted_link": {
    "url": null,
    "expires_at": null,
    "active": false
  },
  "redirect_url": "https://acme.example/kyc/done",
  "locale": "en",
  "credits": {
    "price": 70,
    "held": 0,
    "charged": 70
  },
  "reports": [
    {
      "type": "outcome_letter",
      "available": true,
      "generated_at": "2026-11-07T15:02:11Z"
    },
    {
      "type": "customer_report",
```

```
      "available": true,
      "generated_at": "2026-11-07T15:02:09Z"
    }
  ],
  "partner_decision": null,
  "created_at": "2026-11-05T10:24:51Z",
  "started_at": "2026-11-05T11:02:40Z",
  "submitted_at": "2026-11-05T11:19:03Z",
  "action_required_at": null,
  "decided_at": "2026-11-07T15:01:58Z",
  "expired_at": null,
  "canceled_at": null,
  "updated_at": "2026-11-07T15:02:11Z"
}
```


POST `/v1/test_helpers/verifications/{verification_id}/expire`

Expire the hosted link now

Key permission `verifications:write` | Write limit | Test keys only

PATH PARAMETERS

verification_id `string` REQUIRED

The object's id

RETURNS

200 A **verification** object.

Its fields are listed under Create a verification.

ERRORS

400 Bad request | 401 Not authenticated | 402 Not enough credits | 403 Not allowed | 404 Not found | 409 Conflict | 422 Invalid fields | 429 Too many requests

curl

```
curl https://api.vindexlegalrisk.com/v1/test_helpers/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/expire \
  -H "Authorization: Bearer $VINDEX_API_KEY" \
  -H "Idempotency-Key: $(uuidgen)"
```

Node

```
const response = await
fetch("https://api.vindexlegalrisk.com/v1/test_helpers/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/expire", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${process.env.VINDEX_API_KEY}`,
    "Idempotency-Key": crypto.randomUUID(),
  },
});
const verification = await response.json();
```

Python

```
import os
import uuid
import requests

response = requests.post(
    "https://api.vindexlegalrisk.com/v1/test_helpers/verifications/ver_0vZ3kq9XfJ1mB7c2Lr8sTy/expire",
    headers={
        "Authorization": f"Bearer {os.environ['VINDEX_API_KEY']}",
        "Idempotency-Key": str(uuid.uuid4()),
    },
    timeout=30,
)
response.raise_for_status()
verification = response.json()
```

Response


```
{
  "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "verification",
  "livemode": false,
  "type": "individual",
  "status": "expired",
  "stage": "expired",
  "outcome": null,
  "risk_rating": null,
  "conditions": [],
  "external_reference": "crm-42",
  "metadata": {
    "plan": "gold"
  },
  "case_reference": null,
  "hosted_link": {
    "url": null,
    "expires_at": null,
    "active": false
  },
  "redirect_url": "https://acme.example/kyc/done",
  "locale": "en",
  "credits": {
    "price": 70,
    "held": 0,
    "charged": 0
```

```
  },
  "reports": [
    {
      "type": "outcome_letter",
      "available": false,
      "generated_at": null
    },
    {
      "type": "customer_report",
      "available": false,
      "generated_at": null
    }
  ],
  "partner_decision": null,
  "created_at": "2026-11-05T10:24:51Z",
  "started_at": null,
  "submitted_at": null,
  "action_required_at": null,
  "decided_at": null,
  "expired_at": "2026-11-05T12:00:00Z",
  "canceled_at": null,
  "updated_at": "2026-11-05T10:24:51Z"
}
```


Webhook events

Webhook payloads are **thin**: they say what happened to which object, with its status, and never carry personal data. Fetch the object with your key when you need more. Every payload has the same envelope.

VERIFICATIONS

`verification.created` A verification was created and its hosted link issued.

Payload


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "verification.created",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
    "object": "verification",
    "type": "individual",
    "status": "created",
    "stage": "awaiting_customer",
    "outcome": null,
    "risk_rating": null,
    "external_reference": "crm-42"
  }
}
```

`verification.started` The customer opened the hosted link and started.

Payload


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "verification.started",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
    "object": "verification",
    "type": "individual",
    "status": "in_progress",
    "stage": "customer_in_progress",
    "outcome": null,
    "risk_rating": null,
    "external_reference": "crm-42"
  }
}
```


`verification.submitted` The customer submitted; Vindex review has started.

Payload


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "verification.submitted",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
    "object": "verification",
    "type": "individual",
    "status": "submitted",
    "stage": "under_review",
    "outcome": null,
    "risk_rating": null,
    "external_reference": "crm-42"
  }
}
```


`verification.action_required` Vindex needs more from the customer. Send them their hosted link again.

Payload


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "verification.action_required",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
    "object": "verification",
    "type": "individual",
    "status": "action_required",
    "stage": "awaiting_customer_response",
    "outcome": null,
    "risk_rating": null,
    "external_reference": "crm-42"
  }
}
```


`verification.resubmitted` The customer answered Vindex's request; review continues.

Payload


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "verification.resubmitted",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
    "object": "verification",
    "type": "individual",
    "status": "submitted",
    "stage": "under_review",
    "outcome": null,
    "risk_rating": null,
    "external_reference": "crm-42"
  }
}
```


`verification.decided` Vindex reached its outcome: approved, approved with conditions, or rejected.

Payload


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "verification.decided",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
    "object": "verification",
    "type": "individual",
    "status": "decided",
    "stage": "decided",
    "outcome": "approved",
    "risk_rating": "low",
    "external_reference": "crm-42"
  }
}
```


`verification.expired` The hosted link expired before the customer submitted. Held credits were released.

Payload


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "verification.expired",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
    "object": "verification",
    "type": "individual",
    "status": "expired",
    "stage": "expired",
    "outcome": null,
    "risk_rating": null,
    "external_reference": "crm-42"
  }
}
```


`verification.canceled` The verification was canceled. Held credits were released.

Payload


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "verification.canceled",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
    "object": "verification",
    "type": "individual",
    "status": "canceled",
    "stage": "canceled",
    "outcome": null,
    "risk_rating": null,
    "external_reference": "crm-42"
  }
}
```


`report.ready` A report PDF (customer report or outcome letter) can be downloaded.

Payload


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "report.ready",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "object": "report",
    "external_reference": "crm-42",
    "verification_id": "ver_0vZ3kq9XfJ1mB7c2Lr8sTy",
    "report_type": "outcome_letter"
  }
}
```


SCREENINGS

`screening_batch.completed` Every subject in a batch screening has been screened.

Payload


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "screening_batch.completed",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "id": "sbt_0vZ3kq9XfJ1mB7c2Lr8sTy",
    "object": "screening_batch",
    "status": "completed",
    "total": 25,
    "with_matches": 2
  }
}
```


BILLING

`billing.credits_purchased` Credits were added to your live balance after a payment.

Payload


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "billing.credits_purchased",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "id": "pur_0vZ3kq9XfJ1mB7c2Lr8sTy",
    "object": "credit_purchase",
    "status": "paid",
    "credits": 525
  }
}
```


`billing.balance_low` Your live balance dropped to or below your alert threshold.

Payload


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "billing.balance_low",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "object": "balance",
    "available": 120,
    "held": 70,
    "threshold": 140
  }
}
```


`billing.credits_exhausted` Your live balance reached zero; paid calls now return 402.

Payload


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "billing.credits_exhausted",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "object": "balance",
    "available": 120,
    "held": 70,
    "threshold": 140
  }
}
```


`billing.credits_expiring` Some credits expire within the next 30 days.

Payload


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "billing.credits_expiring",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "object": "balance",
    "available": 120,
    "held": 70,
    "threshold": 140
  }
}
```


`billing.auto_recharge_failed` An automatic top-up failed; auto-recharge is paused.

Payload


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "billing.auto_recharge_failed",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "object": "balance",
    "available": 120,
    "held": 70,
    "threshold": 140
  }
}
```


ACCOUNT

`api_key.expiring` An API key expires within 7 days.

Payload


```
{
  "id": "evt_0vZ3kq9XfJ1mB7c2Lr8sTy",
  "object": "event",
  "type": "api_key.expiring",
  "livemode": true,
  "created_at": "2026-11-05T10:24:51.204Z",
  "data": {
    "id": "key_0vZ3kq9XfJ1mB7c2Lr8sTy",
    "object": "api_key",
    "name": "Production server",
    "expires_at": "2026-12-01T00:00:00Z"
  }
}
```