WatchEye API Documentation

Integration guide and endpoint reference for the WatchEye API

Generated 11 September 2026

Table of Contents

About screenshots and examples

Unless specifically noted as a live example, all names, addresses, phone numbers, email addresses, and other personal details shown in the screenshots and examples throughout this documentation are entirely fictitious and randomly generated for illustrative purposes only. They do not represent real individuals, businesses, or organisations, and no actual personally identifiable information (PII) has been used. Where live examples are included, they use publicly available information - such as government sanctions and watchlist data, or details of public figures sourced from Wikipedia - to demonstrate features like PEP and sanctions screening, and are clearly identified.

API Documentation

WatchEye API System

The WatchEye API allows external programmatic access to the full functionality of the WatchEye system.

For information on managing API keys, see API Keys (login required) in the portal documentation. For the API endpoint reference, see the API Reference.

> **Public Beta.** The WatchEye API is currently in public beta. It is stable and production-capable, and is already being used by integrators to run live workloads. However, some aspects of the API may change as the product evolves and we incorporate feedback from early adopters. Any breaking changes will be managed in line with the API Versioning policy. We actively welcome feedback and suggestions from integrators - please contact your Global Data account manager or the Global Data support team with anything you would like to see improved.

Resource Overview

The API is organised around the following resources. Each resource has list and show endpoints (where applicable), and write endpoints where the resource can be created, updated or deleted via the API. The full request and response schemas, including filters, sort fields and worked examples, live in the API Reference.

Resource	Key endpoints	Purpose
Account	`GET /v1/account`	Confirm credentials, environment, balance and product entitlements
Users	`GET /v1/users`, `GET/PATCH /v1/users/{uuid}`	Read and update portal users on the calling account
Programs	`GET/POST /v1/programs`, `GET/PATCH/DELETE /v1/programs/{uuid}`	Logical containers for entities and monitors
Entities	`GET/POST /v1/programs/{uuid}/entities`, `GET/PATCH/DELETE /v1/entities/{uuid}`, `POST /v1/entities/{uuid}/archive`, `POST /v1/entities/{uuid}/unarchive`	The people, businesses and vessels being monitored
Checks (screening)	`GET /v1/checks`, `GET /v1/checks/{uuid}`, `GET /v1/checks/{uuid}/pdf`, `POST /v1/entities/{uuid}/checks`	Asynchronous PEP, sanction, business, court, adverse media and similar screening tied to an entity
Adhoc checks	`GET/POST /v1/adhoc-checks`, `GET/DELETE /v1/adhoc-checks/{uuid}`, `GET /v1/adhoc-checks/{uuid}/pdf`	One-off screening against a freeform subject with no entity record

Resource	Key endpoints	Purpose
ID Checks	`GET /v1/id-checks`, `GET /v1/id-checks/{uuid}`, `GET /v1/id-checks/{uuid}/pdf`, `POST /v1/entities/{uuid}/id-checks`	Synchronous identity document verification tied to an entity
Adhoc ID Checks	`GET/POST /v1/adhoc-id-checks`, `GET/DELETE /v1/adhoc-id-checks/{uuid}`, `GET /v1/adhoc-id-checks/{uuid}/pdf`	Synchronous identity verification with no entity record
IDPasses	`GET /v1/idpasses`, `GET /v1/idpasses/{uuid}`, `POST /v1/entities/{uuid}/idpasses`, `POST /v1/adhoc-idpasses`, `GET /v1/idpasses/{uuid}/pdf`, `GET /v1/idpasses/{uuid}/images/{type}`	Asynchronous remote identity verification completed by the individual via a hosted link
Monitors	`GET/POST /v1/programs/{uuid}/monitors`, `GET/PATCH/DELETE /v1/monitors/{uuid}`, `POST /v1/monitors/{uuid}/run`	Scheduled or on-demand screening jobs
Events	`GET /v1/events`, `GET/PATCH /v1/events/{uuid}`	The triage queue produced by screening checks
Notes	`GET /v1/notes`, `POST /v1/entities/{uuid}/notes`, `GET/PATCH/DELETE /v1/notes/{uuid}`	Free-text annotations on entities and events
Audits	`GET /v1/audits`, `GET /v1/audits/{uuid}`	Full account audit log, including a row for every API call

Integration security and data handling

Before you build an integration against the WatchEye API, review the responsibilities below. They apply in addition to the authentication, billing, and rate-limit guidance in the rest of this guide.

Protect your API keys

Each API key is a privileged credential for your WatchEye account. A write-enabled key can read and change the records your integration is permitted to reach through the API. A read-only key can read that data without mutating it. In every case, possession of the key and secret is enough to call the API as your account.

Keep API keys and secrets safe: store them in a secrets manager or equivalent, rotate them when access requirements change, and avoid embedding them in source code or shipping them to client-side applications.

Reduce exposure by configuring **IP address or subnet restrictions** on each key in the portal (API Keys (login required)). A key without restrictions can be used from any internet address that has the key and secret.

Personal information in your environment

Using the API has the potential to copy personal information (PII) from WatchEye into your application, databases, logs, backups, and any other systems that store API responses or request payloads. Each integration adds another place where that data may be held.

Personal information stored in WatchEye is covered by Global Data's security policies for the platform. Once your integration copies that information into systems you run (applications, databases, logs, and backups), protecting it becomes **your** responsibility. Global Data's platform policies apply to data in WatchEye, not to copies held in your environment.

Plan your integration with that in mind: retrieve only the fields you need, protect stored API traffic and responses, define retention that matches your compliance requirements, and treat audit logs and application logs that capture API payloads as sensitive.

Access Overview

The API is a RESTful interface with JSON payloads that uses standard HTTP methods and status codes. It is served from https://portal.watcheye.com.au/api/v1 and authenticated with an API key.

API keys are created and managed in the WatchEye portal at https://portal.watcheye.com.au (login required). The portal is also where you view your API usage and access logs, manage users and check your account balance.

Programmatic API Access

The API can be accessed programmatically by authenticating using an API key. You can create multiple API keys and restrict them to particular IP addresses or subnets.

Authentication and Authorization

Overview

To securely access resources via the WatchEye API, clients are required to authenticate using API keys. These tokens serve as credentials, ensuring only authorized entities can interact with the API. This section outlines the process of obtaining, using, and managing these tokens.

Generating an API Key

Visit the WatchEye portal: https://portal.watcheye.com.au

Log in with your registered credentials.

Navigate to the **Account** -> **API Keys** section.

From here, you can generate one or multiple API keys as per your requirement.

> When an API key is created, both the API key and secret will be displayed. You must note down the secret as it will not be displayed again.

Using Your API Key

Once you have your API key and API Secret, include them in the request header when making API calls. The `Authorization` header format is as follows:

```
Authorization: Bearer <your-api-key>|<your-api-secret>
```

Replace `<your-api-key>` with your actual API key and `<your-api-secret>` with your actual API secret. Note that they are separated by a pipe character.

Handling Authentication Failures

If a request is made with an invalid, expired, or missing key or secret, the API will respond with a `401 Unauthorized` status. This indicates that the server understands the request but refuses to authorize it. When you receive such a response:

- Double-check the key and secret you've provided in the `Authorization` header.
- Ensure that any IP address or subnet restrictions you've set up align with the IP address making the request.
- If you suspect the key or secret has been compromised, generate a new one from the WatchEye portal and replace the old one in your application.

Best Practices

- **Store API keys securely**: Treat your API keys and secrets like passwords. Store them securely, and avoid hardcoding them directly in your code.
- **Regularly rotate keys and secrets**: For added security, consider rotating your API keys and secrets periodically.
- **Monitor for unauthorized access**: Monitor your API usage for any unexpected or unauthorized activity.

Following these guidelines will help maintain the integrity and security of your interactions with the WatchEye API.

Quick Start

Once you have an API key and secret, the recommended first call is `GET /v1/account`. It is not billed against any product, works for every active key, and returns the account environment, products and current balance. This lets you confirm three things before going any further: that your credentials work, that you are pointed at the expected environment, and that the products you intend to use are enabled.

1. Make your first call

```
curl -H "Authorization: Bearer <your-api-key>|<your-api-secret>" \
     https://portal.watcheye.com.au/api/v1/account
```

A successful response looks like:

```
{
  "data": {
    "uuid": "7a1f0c8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
    "account_name": "Sample Account",
    "environment": "live",
    "balance": "1234.5678",
    "credit_limit": "100.0000",
    "available_credit": "1334.5678",
    "products": [
      { "name": "pep_sanction_check", "label": "PEP and Sanction Check" }
    ]
  },
  "api_reference": "9a4b1c8e-2f63-4f9a-9f3a-9b1f5a7c2d4f"
```

```
}
```

If you get `401 Unauthorized`, double-check the `Authorization` header format (`Bearer <key>|<secret>`, separated by a pipe character) and that any IP restrictions on the key include the address you are calling from. See Common error responses for the 401 messages.

2. Confirm the environment

`data.environment` is either `live` or `sandbox`. Calls made with a key on a sandbox account are not billed; calls made with a key on a live account are. Sandbox and live are separate accounts with separate keys - see Sandbox for how to use them in your integration test cycle.

3. Confirm your product entitlements

Each entry in `data.products` is a billable operation your account is entitled to call. If you intend to run PEP & Sanction checks, look for `pep_sanction_check`; for business screening, look for `business_check`; for identity verification, look for the per-document-type entries you need. A missing entry typically surfaces later as `403 No access` on the corresponding launch endpoint - resolve it with your Global Data account manager before continuing.

4. Where to go next

Common follow-on calls for a new integration:

- List your programs: `GET /v1/programs`
- Create your first entity: `POST /v1/programs/{program_uuid}/entities`
- Launch a screening check against the entity: `POST /v1/entities/{entity_uuid}/checks`
- Poll the check result: `GET /v1/checks/{check_uuid}`
- Review the audit log of every API call you have made: `GET /v1/audits`

Before you write code against these, skim API Conventions and List Endpoint Conventions - they describe the field-naming, pagination, filter and sort rules that apply uniformly to every endpoint, so each operation's reference only has to describe what is specific to it.

API Conventions

Paths in this guide are written as `/v1/...`. The full URL is `https://portal.watcheye.com.au/api` followed by the path, so `GET /v1/account` is `GET https://portal.watcheye.com.au/api/v1/account`.

Country codes are ISO 3166-1 alpha-2 and are shown in uppercase (`AU`). The API accepts them in either case and always returns them in uppercase.

These conventions apply to every endpoint. Knowing them up front makes the per-operation reference easier to read, because each endpoint only documents what is specific to it.

URLs

Every endpoint is mounted under `/api/v1/`. The version number is part of the URL - see API Versioning.

Path segments are kebab-case: `/v1/id-checks`, `/v1/adhoc-checks/{uuid}`, `/v1/entities/{uuid}/archive`.

Resource identifiers in paths are always UUIDs.

JSON field names and query parameters

Field names in request and response bodies are snake_case: `first_name`, `created_at`, `address_line1`, `is_immutable`.

Query parameter names are snake_case: `per_page`, `created_at_from`, `archived`.

Enum and discriminator string values are snake_case: `"individual"`, `"api_key"`, `"pep_sanction_check"`.

Identifiers

Every resource is identified by a `uuid` (RFC 4122). UUIDs are the only identifiers accepted in URLs and the only ones that should be persisted on your side as a foreign key into WatchEye.

Some resources also carry a short, human-friendly identifier (`program_number`, `entity_number`, `check_number`, etc.) for display in dashboards. These are not stable lookup keys - always use the `uuid`.

Dates and timestamps

Timestamps are ISO 8601 UTC: `2025-01-15T03:00:00Z`.

Calendar dates use `YYYY-MM-DD` unless otherwise noted (Medicare and ASIC/MSIC `card_expiry` accept `YYYY-MM`).

Money

Monetary fields (`balance`, `credit_limit`, `available_credit`) are returned as decimal strings with 4 decimal places, for example `"1234.5678"`. Use a decimal arithmetic library, not floating point, when comparing or accumulating these values.

Response envelopes

Every endpoint returns one of three shapes.

Show, create, update, async accept, and singleton (`/account`) responses wrap the resource under a `data` key:

```
{ "data": { ... }, "api_reference": "9a4b1c8e-2f63-4f9a-9f3a-9b1f5a7c2d4f" }
```

List responses wrap the array under `data` and add a `meta` block describing the pagination:

```
{
  "data": [ ... ],
  "meta": { "current_page": 1, "per_page": 30, "total": 42, "last_page": 2 },
  "api_reference": "9a4b1c8e-2f63-4f9a-9f3a-9b1f5a7c2d4f"
}
```

Error responses carry a `message` and, on validation failures, an `errors` map - see Errors.

`api_reference` is present on every response, success or failure, and is the unique UUID of the audit-log row for the call. Capture it in your logs and quote it when raising support tickets - see Errors for why.

Content type and request bodies

Requests with a body (`POST` , `PATCH`) must send `Content-Type: application/json` and a valid JSON object as the body. Form-encoded bodies are not accepted.

Empty `POST` and `PATCH` bodies should be sent as `{}` rather than no body at all.

Authentication

Every endpoint requires an API key and secret combination sent as a Bearer token. See Authentication and Authorization for the header format. There are no anonymous endpoints.

List Endpoint Conventions

Every list endpoint (`GET /v1/programs` , `GET /v1/programs/{program_uuid}/entities` , `GET /v1/checks` , etc.) shares the same pagination, filtering and sorting query parameters. The per-operation reference only documents the filters and sort fields specific to that resource.

Pagination

`page` selects the page number (1-based; default `1`).

`per_page` sets the page size (default `30` , maximum `500`).

The response envelope includes a `meta` block with `current_page` , `per_page` , `total` (total matching rows across all pages) and `last_page` .

```
GET /v1/programs/9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f/entities?per_page=100&page=2
```

Sending `per_page` larger than 500, or `page` smaller than 1, returns `400 Bad Request` .

Filtering

Filters use the `filter[key]=value` syntax. The supported filter keys are listed per endpoint; unknown keys are ignored.

```
GET /v1/checks?filter[outcome]=warning&filter[program_uuid]=9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f
```

Filters that target a UUID (a program, entity, parent check, user, API key, and so on) return an empty page (`data: []` , `meta.total: 0`) when the UUID is unknown or belongs to a different account - they do not return `404` . This is the same response as a filter that legitimately matches no rows.

Boolean filters such as the top-level `?archived=true` accept any of `true` / `false` , `1` / `0` , or the string forms `"true"` / `"false"` interchangeably.

Sorting

The `sort` query parameter selects the order. Prefix the field name with `-` for descending order:

```
GET /v1/programs?sort=program_name    # A-Z by name
GET /v1/programs?sort=-created_at     # newest first
```

Each endpoint documents the sort fields it supports. The default sort is the one most useful for the resource - normally `-created_at` or `-checked_at`.

Date-range filters

List endpoints that accept a date range use a `..._from` / `..._to` pair (or `checked_after` / `checked_before` on the checks endpoints). Both bounds are inclusive and accept ISO 8601 timestamps.

Synchronous and Asynchronous Endpoints

Write endpoints fall into two categories. The HTTP success status code tells you which is which.

Synchronous (200 OK / 201 Created)

The response body contains the final state of the resource. No follow-up calls are required.

Endpoint family	Success status	Returns
All read endpoints (`GET ...`)	`200`	Resource (or paginated list)
`POST /v1/programs` , `POST /v1/programs/{uuid}/entities` , `POST /v1/programs/{uuid}/monitors` , `POST /v1/entities/{uuid}/notes`	`201`	Created resource
`PATCH` / `DELETE` on any resource	`200`	Updated or soft-deleted resource
`POST /v1/entities/{uuid}/archive` , `POST /v1/entities/{uuid}/unarchive`	`200`	Updated entity
`POST /v1/entities/{uuid}/id-checks` , `POST /v1/adhoc-id-checks`	`200`	Completed ID check (full payload, including the final `outcome`)

Identity verification checks are run synchronously against the upstream provider before the response is returned - the `outcome` (`pass` / `fail`) is final by the time you get the response. The one exception is an ID check completed through an IDPass verification, which reports `pending` until the person finishes (see IDPass); otherwise there is no `pending` state to poll for ID checks.

Asynchronous (202 Accepted)

The response confirms that the work has been queued. The actual screening runs in the background and you must poll the corresponding show endpoint until `status` (or, for monitor runs, `process_status`) reaches a terminal value.

Launch endpoint	Poll	Terminal values
`POST /v1/entities/{uuid}/checks`	`GET /v1/checks/{uuid}`	`status` is `complete` or `failed`
`POST /v1/adhoc-checks`	`GET /v1/adhoc-checks/{uuid}`	`status` is `complete` or `failed`
`POST /v1/monitors/{uuid}/run`	`GET /v1/monitors/{uuid}`	`process_status` is `null` (idle again), `failed` or `paused`

The launch response carries the new resource's `uuid` (and on checks, a `detail_url` you can use directly). A reasonable polling cadence is every 2-5 seconds for the first 30 seconds, backing off to every 15-30 seconds for longer-running operations. Most screening checks complete within a few seconds; a monitor run across thousands of entities can take several minutes.

Idempotency interacts with polling

When you retry an async launch endpoint with the same `Idempotency-Key`, the replayed `202` response carries the same `uuid` as the original call (and an additional `Idempotent-Replay: true` response header). The background work was already queued by the original call - keep polling the original `uuid` rather than launching a new one. See Idempotency for the full contract.

Billing and Account Management

Overview

Some calls to the WatchEye API are chargeable, where each access to a chargeable API endpoint incurs a cost. The specifics of these charges, including rate and any potential discounts, are determined on a per-account basis.

Chargeable API Calls

Every time you make a request to a chargeable API endpoint, the associated fee is immediately billed to your account. Detailed billing rates, as well as any applicable promotions or discounts, will be communicated to you by your Global Data account manager.

Billing Statements

While charges are billed in real-time, you can access detailed billing statements, including a breakdown of API calls and their respective costs, through the WatchEye portal.

Credit and Call Limits

Your API account will have predefined limits based on the credit amount or the number of API calls:

Credit Limit: If your account balance reaches zero or goes into a negative due to accessing chargeable endpoints, your ability to make further chargeable API calls will be temporarily halted.

Call Limit: Some accounts may have a restriction on the number of API calls in a given period (e.g., monthly). Once this limit is reached, additional chargeable API requests will not be processed.

When either of these limits is breached, any subsequent chargeable API requests will return a `402 Payment Required` response.

Managing and Monitoring Your Account

To keep track of your usage and to avoid any unwanted disruptions:

Regularly check your account balance: This ensures you are aware of your remaining credits.

Set up alerts: If possible, configure notifications to alert you when your balance or call count approaches its limit.

Contact your account manager: For any billing discrepancies or to discuss adjustments to your credit or call limits.

Adhoc check data fields

The body of `POST /v1/adhoc-checks` carries the subject to screen in a `data` object. Adhoc checks have no entity record to read from, so every required field has to be supplied in the request itself.

The required keys depend on which operations are listed in `check_types`. When multiple operations are launched together, each operation validates the same shared `data` object independently - every required field for every requested operation must be present.

Validation runs **before** any check rows are created or any credit is charged. A missing or malformed field returns `400 Bad Request` identifying the offending key (`data.first_name` , etc.).

Entity-driven launches (`POST /v1/entities/{uuid}/checks` and monitor runs) read the same fields from the saved entity record, not from the request body, so this section does not apply to them.

Common field formats

These conventions show up across multiple operations:

Field family	Format	Notes
Names (`first_name` , `middle_name` , `last_name` , `business_name` , etc.)	UTF-8 strings	Per-operation length and character-set constraints; see the operation tables below.
`dob`	`YYYY-MM-DD` (ISO 8601) or `DD/MM/YYYY`	Year range varies by operation: `pep_sanction_check` allows 1900-2099, `adc_check` and `adc_custodian_check` allow 1800-2099.
`address_country`	ISO 3166-1 alpha-2 country code	E.g. `AU` , `GB` , `US` .

Field family	Format	Notes
`address_state`	AU state code	For Australian operations: `ACT`, `NSW`, `NT`, `QLD`, `SA`, `TAS`, `VIC`, `WA`.

For check types where multiple subject "kinds" are supported (individual, business, vessel) the per-operation table describes which fields belong to which kind. Mixing fields from two kinds in the same `data` object is rejected (e.g. supplying `business_name` together with `first_name` for `pep_sanction_check`).

Per-operation field reference

pep_sanction_check

Screens an individual, business or vessel against PEP and sanction watchlists. Exactly one of the three subject kinds must be supplied.

Field	Required	Type / format	Notes
`full_name`	required for individual	string, max 255	Use this **or** `first_name` + `last_name`.
`first_name`	required for individual	string, max 255	When supplied, `last_name` must also be supplied (or `full_name` instead).
`middle_name`	optional	string, max 255	Only valid alongside individual fields.
`last_name`	required for individual	string, min 2, max 255	When supplied, `first_name` or `full_name` must also be supplied.
`dob`	optional	date	`YYYY-MM-DD` or `DD/MM/YYYY`, year 1900-2099. Individual subjects only.
`business_name`	required for business	string, max 255	Mutually exclusive with all individual and vessel fields.
`vessel_name`	required for vessel	string, max 255	Mutually exclusive with all individual and business fields.
`address_country`	optional	ISO country code	Used to scope the search.

adverse_media

Screens an individual or business for adverse media coverage. Exactly one subject kind must be supplied.

Field	Required	Type / format	Notes
`first_name`	required for individual	string, min 1, max 50	Letters, hyphens, apostrophes, periods, spaces.
`middle_name`	optional	string, min 1, max 50	Same character set as `first_name`.
`last_name`	required for individual	string, min 1, max 50	Same character set as `first_name`.
`dob`	optional	date	`YYYY-MM-DD` or `DD/MM/YYYY`.

Field	Required	Type / format	Notes
`business_name`	required for business	string, min 1, max 100	Mutually exclusive with the individual fields.
`address_country`	required	ISO country code	Adverse media is scoped per country.

adc_check

Australian death-certificate match. Individual subjects only.

Field	Required	Type / format	Notes
`first_name`	required	string, min 1, max 50	Letters, hyphens, apostrophes, periods, spaces.
`middle_name`	optional	string, min 1, max 50	
`last_name`	required	string, min 1, max 50	
`dob`	required	date	`YYYY-MM-DD` or `DD/MM/YYYY`, year 1800-2099.
`address_state`	optional	AU state code	One of `ACT`, `NT`, `NSW`, `QLD`, `SA`, `TAS`, `VIC`, `WA`.

adc_custodian_check

Custodian variant of the ADC check. Same fields and rules as `adc_check`.

Field	Required	Type / format	Notes
`first_name`	required	string, min 1, max 50	
`middle_name`	optional	string, min 1, max 50	
`last_name`	required	string, min 1, max 50	
`dob`	required	date	`YYYY-MM-DD` or `DD/MM/YYYY`, year 1800-2099.
`address_state`	optional	AU state code	One of `ACT`, `NT`, `NSW`, `QLD`, `SA`, `TAS`, `VIC`, `WA`.

banned_disqualified_persons

ASIC and ATO banned-or-disqualified registers. Individual subjects only.

Field	Required	Type / format	Notes
`first_name`	required	string, min 1, max 50	
`middle_name`	optional	string, min 1, max 50	
`last_name`	required	string, min 1, max 50	
`address_state`	optional	string, max 50	
`address_country`	optional	ISO country code	

court_check

Australian court-record match. Either an individual or a business subject - the two are mutually exclusive.

Field	Required	Type / format	Notes
`first_name`	required for individual	string, min 1, max 50	
`middle_name`	optional	string, min 1, max 50	
`last_name`	required for individual	string, min 1, max 50	
`business_name`	required for business	string, min 1, max 512	Mutually exclusive with the individual fields.
`address_state`	optional	AU state code	One of `ACT`, `NT`, `NSW`, `QLD`, `SA`, `TAS`, `VIC`, `WA`.

email_check

Verifies the deliverability of one to four email addresses.

Field	Required	Type / format	Notes
`email1`	required (when `email2` - `email4` are empty)	RFC-compliant email, max 255	At least one of `email1` - `email4` must be supplied.
`email2`	optional	RFC-compliant email, max 255	
`email3`	optional	RFC-compliant email, max 255	
`email4`	optional	RFC-compliant email, max 255	

Each populated email is billed and verified separately. Submitting all four addresses produces four billable units.

phone_check

Verifies one to four Australian phone numbers.

Field	Required	Type / format	Notes
`phone1`	required (when `phone2` - `phone4` are empty)	AU phone number	E.164 (`+61...`) or local (`0...`); landline area codes `02` / `03` / `07` / `08` or mobile `04`.
`phone2`	optional	AU phone number	
`phone3`	optional	AU phone number	
`phone4`	optional	AU phone number	

Each populated phone number is billed and verified separately.

business_check

Australian business / company lookup, driven by ABN or ACN.

Field	Required	Type / format	Notes
`business_number_type`	required	`au_abn` or `au_acn`	
`business_number`	required	digits	11 digits when `business_number_type` is `au_abn`, 9 digits when `au_acn`. ABN/ACN modulo checksums are also enforced at launch time.
`business_name`	optional	string, max 255	Returned alongside the lookup result for traceability.

uk_business_check

UK Companies House lookup.

Field	Required	Type / format	Notes
`business_number_type`	required	`uk_crn`	
`business_number`	required	8-character alphanumeric	Case-insensitive. Uppercase prefixes such as `SC` (Scotland) are accepted.
`business_name`	optional	string, max 255	

realestate_check

Australian property history lookup, driven by address.

Field	Required	Type / format	Notes
`address_country`	required	`AU`	The real-estate dataset is Australia-only.
`address_line1`	required	string, min 4, max 100	
`address_line2`	optional	string, max 100	
`address_suburb`	required	string, min 3, max 60	
`address_state`	required	AU state code	One of `ACT`, `NT`, `NSW`, `QLD`, `SA`, `TAS`, `VIC`, `WA`.
`address_postcode`	required	3 or 4 digits	

Worked example

A multi-check launch combining a PEP/sanction screen, a court check and a phone check on the same individual:

```
curl -X POST https://portal.watcheye.com.au/api/v1/adhoc-checks \
  -H "Authorization: Bearer <key>|<secret>" \
```

```
  -H "Content-Type: application/json" \
  -d '{
    "check_types": ["pep_sanction_check", "court_check", "phone_check"],
    "config": {
      "court_check": { "listing": "all" }
    },
    "data": {
      "first_name": "John",
      "last_name": "Smith",
      "dob": "1980-06-23",
      "phone1": "+61412345678",
      "address_state": "VIC",
      "address_country": "AU"
    }
  }'
```

Because three check types were requested, the response is `202 Accepted` with a parent group adhoc-check UUID and one child entry per operation. Poll each child via its `detail_url` until `status` is `complete` or `failed`, then read the typed `details` block via the show adhoc check endpoint.

Rate Limiting

Overview

To ensure equitable access and maintain performance for all users, the WatchEye API enforces rate limiting on all endpoints. This section provides details on how rate limiting works, its default configurations, and how to manage and respond to rate limit constraints.

Default Rate Limit Configuration

By default, the rate limit for the API is **600 requests per minute**. The limit is enforced with a 60-second fixed window: each request consumes one slot from the current window, and the counter resets once the window expires. Pacing requests evenly over the minute is the safest way to stay below the limit.

Adjusting Your Rate Limit

If you need a higher request rate, adjustments can be made on a case-by-case basis. To initiate a review or request a change in your rate limit:

Contact Global Data support.

Provide a clear reason for the requested change, along with any data or justification that supports your use case.

Once your request is reviewed, the support team will contact you with a decision or request further information.

Exceeding Rate Limit & Response Handling

When the rate limit is exceeded, the API responds with a `429 Too Many Requests` status. Applications should implement a back-off strategy and wait before retrying.

Every response (both successful responses and `429`s) includes headers describing the current rate-limit state:

- **`X-RateLimit-Limit`** - the maximum number of requests allowed in the current 60-second window.
- **`X-RateLimit-Remaining`** - the number of requests still available before the limit is hit.

On a `429` response, two additional headers tell you exactly how long to wait:

- **`Retry-After`** - the number of seconds to wait before the next request will be accepted. This is the value to use to schedule your retry.
- **`X-RateLimit-Reset`** - the Unix timestamp (seconds since the epoch) at which the current window will reset.

For example, a successful response when the limit is 600 per minute and 1 call has been made:

```
X-RateLimit-Limit: 600
X-RateLimit-Remaining: 599
```

A `429` response when the limit has just been hit and the window resets in 42 seconds:

```
HTTP/1.1 429 Too Many Requests
X-RateLimit-Limit: 600
X-RateLimit-Remaining: 0
Retry-After: 42
X-RateLimit-Reset: 1740000042
```

HTTP header names are case-insensitive; the headers above may appear in lowercase depending on your HTTP client.

Best Practices

Honour `Retry-After`: On a `429` response, schedule your retry for `Retry-After` seconds in the future rather than using a fixed sleep. This keeps your back-off correctly sized for the actual reset window.

Watch `X-RateLimit-Remaining`: Slow down pre-emptively when the remaining count drops near zero rather than waiting for a `429`. Many integrations add a small dynamic pause once the remaining count falls below a threshold.

Spread out batch work: If you need to process a large batch (for example, creating many entities in one go), distribute the requests evenly over time instead of bursting.

Centralise rate-limit handling: Implement the `429` / `Retry-After` handling once in your HTTP client wrapper so every call site benefits, rather than spreading retry logic through individual endpoints.

Idempotency

Overview

Network calls can fail in ambiguous ways. A connection can be dropped after your request has been received and processed by WatchEye but before the response reaches you. Without help from the server, retrying the request would risk creating duplicate records, double-billing your account, or sending the same downstream request twice.

To make retries safe, the WatchEye API supports the **`Idempotency-Key`** request header. When you supply this header on a write request, WatchEye guarantees that the same key sent more than once will only do the underlying work once. Any subsequent request with the same key returns the original response, byte-for-byte, with no new database changes and no new billable charges.

Sending the header is **always optional**. If you do not send it, each request is treated as a brand new request.

Generating a Key

The `Idempotency-Key` value is a string of your choosing, up to 255 characters. It must be unique per logical operation - typically you generate a fresh UUID per request from your application:

```
Idempotency-Key: 8c4d3a18-2f63-4f9a-9f3a-9b1f5a7c2d4f
```

Two different operations should never share the same key. Reusing a key for a different request body is treated as a client error (see below).

Which Endpoints Honour the Header

The header is honoured on **`POST` requests only**. `GET`, `PATCH` and `DELETE` are conventionally idempotent at the HTTP level already (re-sending the same `PATCH` or `DELETE` produces the same end state), so the header is silently ignored on those methods.

If you send the header on a method where it has no effect, the request is processed normally and no error is returned.

Behaviour

For a `POST` request that includes the header, one of the following will happen:

First request - WatchEye records the key, runs the request, charges any applicable fees, returns the response, and remembers the response for **24 hours**.

Retry with the same key and same body - WatchEye returns the **original response, verbatim** (same status code, same body, same `api_reference`). No new work is done and no new charge is applied. The replay is identifiable by an additional response header:

```
Idempotent-Replay: true
```

Retry with the same key but a different body - WatchEye returns `422 Unprocessable Entity` with the message:

```
{ "message": "Idempotency-Key was reused with a different request body." }
```

This is a programming error in your client - either generate a new key for the new request, or send the same body as the original.

Concurrent retry while the first request is still in flight - WatchEye returns `409 Conflict` with the message:

```
{ "message": "A request with this Idempotency-Key is already being processed." }
```

Wait for the first request to complete (a few seconds at most for normal calls), then retry.

Original request failed (non-2xx) - The key is released. You can re-send with the same key, and the request will run fresh. This is intentional: if validation failed the first time and you have fixed your payload, your retry should run as a normal new request.

Detecting a Replay

Every replayed response includes the header:

```
Idempotent-Replay: true
```

Fresh responses do not include this header (the header is absent rather than `false`). If you ever need to know whether you are looking at the original outcome or a cached replay - for example, when debugging "why didn't this request seem to do anything?" - check for this header in the response.

This is especially useful for asynchronous endpoints such as `POST /v1/monitors/{uuid}/run`, where the response body alone does not tell you whether a new background job was actually queued or whether you are looking at the cached body from an earlier call.

Storage Window

A successful (2xx) response is remembered for **24 hours**. After that the entry is discarded; a request with the same key after the window will be treated as a brand new request. In practice this is far longer than any reasonable client-side retry window.

Worked Example

A typical "create entity then retry on network error" flow:

```
POST /api/v1/programs/9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f/entities HTTP/1.1
Authorization: Bearer wch_xxxx|secret
Idempotency-Key: 8c4d3a18-2f63-4f9a-9f3a-9b1f5a7c2d4f
Content-Type: application/json

{ "entity_type": "individual", "first_name": "Jane", "last_name": "Doe", "risk_level": "low" }
```

If the response never reaches you, replaying the **identical** request returns the originally-created entity, not a duplicate:

```
HTTP/1.1 201 Created
Idempotent-Replay: true
Content-Type: application/json

{ "data": { "uuid": "...", ... }, "api_reference": "..." }
```

Best Practices

Generate keys per logical operation, not per HTTP request: if your retry loop wraps the HTTP call, the same key must be used on every attempt of that loop. A new key per attempt defeats the entire mechanism.

Use UUIDs: They are easy to generate, globally unique by construction, and have no information leakage.

Treat the key as opaque: WatchEye stores a hash of the key, not the key itself. Do not depend on parsing the key on either side.

Do not reuse a key for a different operation: Always generate a fresh key when the request body changes.

Always retry on network failures with the same key: This is what the header is for. You cannot accidentally double-charge yourself.

Testing with API Clients (Postman, Insomnia, etc.)

Many GUI HTTP clients automatically attach a sticky `Idempotency-Key` to saved POST requests. The key is generated once when you first send the request and is then re-used on every subsequent click of **Send** until you regenerate or remove it. Combined with the 24-hour replay window, this means a single "test" request can keep replaying the same cached response for the rest of the day, even though it looks to you like you are submitting a fresh request each time.

The symptoms are easy to mistake for a server-side problem:

The response body keeps showing the same `pending` / `created` state from your first call.

The `api_reference` UUID is identical across every response (it should be different on every real request).

No new database records appear and, for async endpoints, no new background job is queued.

The `Idempotent-Replay: true` response header is present.

If you see those symptoms while testing:

Open the request's **Headers** tab in your client.

Either remove the `Idempotency-Key` header entirely (it is optional) or regenerate its value before each Send.

Confirm by checking that `api_reference` changes between calls and that `Idempotent-Replay` is no longer in the response.

For production traffic the sticky key is exactly what you want - it makes "retry on network error" automatically safe. It is only the interactive "click Send again" testing workflow where it surprises people.

API Versioning

Overview

Versioning is a critical aspect of API design, ensuring that changes to the API can be rolled out without negatively impacting existing applications. The WatchEye API employs a straightforward versioning approach by incorporating the version number in the request URL. This section provides an understanding of the versioning scheme and how changes are managed.

Version Number in URL

The version number is embedded directly in the API endpoint URL. For instance:

```
https://portal.watcheye.com.au/api/v1/resource-endpoint
```

In the above example, `v1` indicates that the API endpoint belongs to version 1.

Changes & Backward Compatibility

Additions: New resources and methods can be introduced to an existing version without changing the version number, provided they don't affect backward compatibility. This ensures seamless integration for developers relying on the current version.

Modifications: If a method undergoes changes that alter its functionality in a way that disrupts backward compatibility, a new version of the API will be introduced.

This documentation pertains to **version 1**. Check the version in the endpoint URL to be sure you are accessing the correct version.

Version Support Lifecycle

Every version of the WatchEye API receives full support for a minimum of **12 months** from the introduction of its subsequent version. This provides ample time for developers to adapt to newer versions without abrupt disruptions.

For example, when version 2 is launched, version 1 will continue to be supported for at least another 12 months.

Transitioning Between Versions

It's advisable for developers to:

Regularly check for announcements on new API versions.

Test their applications against the new version during the support overlap period.

Migrate to the newer version before the end of the previous version's support period.

Best Practices

Stay Updated: Monitor official channels for announcements related to version releases or changes.

Maintain Flexibility: Design your application to be adaptable, making it easier to accommodate changes in the API.

By adhering to these guidelines and understanding the versioning strategy, developers can ensure smooth interactions with the WatchEye API and easily navigate between versions.

Maintenance Windows

To ensure that the WatchEye API performs optimally, periodic maintenance is necessary. During these maintenance windows, you may experience brief interruptions, delayed responses, or reduced system capacities. We've scheduled these windows to have the least possible impact on the majority of our users.

Time Zone: All mentioned times are in Australia/Melbourne local time and may differ in their offset from GMT based on the local daylight saving applied. (AEST: GMT+10 or AEDT: GMT+11)

Window Start	Window End	System Impact
Sunday 04:00	Sunday 04:30	Reduced capacity. Query response times may be impacted.

What to Expect During Maintenance

Reduced Capacity: While the system remains operational, it may operate at a lower capacity than usual, leading to potentially slower response times.

Potential Downtime: Rarely, the API might be temporarily unavailable. However, this is not common during the maintenance windows outlined above.

Updates & Improvements: Maintenance allows us to deploy upgrades and optimizations, ensuring a robust and efficient API service.

Recommendations for API Users

Plan Accordingly: If possible, try to schedule your heavy API tasks outside of these maintenance windows.

Error Handling: Make sure your integration has adequate error handling for times when the API might be slow or temporarily unreachable.

Stay Informed: While the above table mentions our regular maintenance windows, there might be rare occasions where we need additional or emergency maintenance. We recommend subscribing to our status updates or notifications to stay informed.

Questions or Concerns

If you have queries about these maintenance windows or need further clarification, please don't hesitate to reach out to the Global Data support team. We're here to assist and ensure a seamless experience with our API.

Errors

When a request fails the API responds with a non-2xx HTTP status code and a JSON body using the same envelope for every endpoint.

Error response shape

```
{
  "message": "A human-readable description of what went wrong.",
  "api_reference": "9a4b1c8e-2f63-4f9a-9f3a-9b1f5a7c2d4f",
  "errors": {
    "data.passport_number": ["The passport number format is invalid."]
  }
}
```

Field	Always present	Description
`message`	Yes	A short, human-readable message. Suitable for logging or surfacing to operators - not always suitable for end users.
`api_reference`	Yes	A unique UUID for this exact request. The same value is recorded against the call's audit trail entry on WatchEye's side.
`errors`	On `400` only	Field-keyed map of per-field validation messages. Present when the failure is a validation error (e.g. missing or malformed `data.*` fields). Each value is an array of one or more messages.

The same envelope is used across all error status codes (`400` , `401` , `402` , `403` , `404` , `409` , `422` , `429` , `5xx`). Successful responses also include `api_reference` so you can correlate any response - success or failure - against your own logs.

Using api_reference when raising support tickets

Every API call - whether it succeeded, returned a validation error, or hit an internal error - is written to WatchEye's audit log, keyed by the `api_reference` UUID. If you need to raise a support ticket about a specific failed call:

Capture the `api_reference` value from the response body. (Many integrations already log every response payload - if yours does not, log at minimum the status code, the `message` , and the `api_reference` .)

Include the `api_reference` in your support request, along with the endpoint you called and the rough time of the call.

Support can use the `api_reference` to look up the exact request body, status code, response body and (where applicable) upstream provider response that we stored for that call.

This makes intermittent failures much easier to diagnose than a textual description of the symptoms alone, especially for endpoints that proxy out to third-party providers (`business_check` , ID checks, etc.).

> **Note:** For idempotent replays (see the Idempotency section), the `api_reference` returned by the replay is the same UUID as the original successful call, not a new value. That is intentional - the replay is identical to the original response in every respect.

Soft Deletion

WatchEye preserves a record of every entity, program, monitor, note and related artefact for compliance and audit purposes. To make that possible, all `DELETE` endpoints in the API are **soft deletes** rather than physical deletes.

What "soft-deleted" means

A soft-deleted record is hidden from both the API and the portal but remains in the database:

- The record is no longer returned by `GET` list endpoints.
- The record's individual `GET /<resource>/{uuid}` endpoint returns `404 Not Found`.
- `POST` / `PATCH` calls that reference the record by UUID return `404 Not Found`.
- Where a record has dependents (e.g. deleting a program also soft-deletes its entities, checks, ID checks, reports, events and notes), the dependent records are hidden in the same way.
- The original `DELETE` call returns the deleted record's representation once as confirmation. Subsequent calls return `404`.

Recovery window

Soft-deleted records can be restored within **30 days** of deletion. Restoration is currently a portal-only action: a user with the account manager role can navigate to the relevant list (e.g. **Programs**, **Entities**, **Monitors**), apply the "Show deleted" filter, open the record and choose **Restore**. There is no API equivalent of the restore action.

After the 30-day window, soft-deleted records are eligible for permanent removal and may be purged from the database without notice. Once purged, the record and its UUID cannot be recovered.

If you need to retain a record beyond 30 days but no longer want it surfaced in your normal workflows, archive it instead of deleting it (where the resource supports archiving) or use the appropriate `status` / `flags` fields to mark it as inactive.

What is and isn't soft-deleted

Endpoint	Behaviour
`DELETE /v1/programs/{uuid}`	Soft-deletes the program and cascades to every dependent record
`DELETE /v1/entities/{uuid}`	Soft-deletes the entity and its dependent records
`DELETE /v1/monitors/{uuid}`	Soft-deletes the monitor (no dependent records)
`DELETE /v1/notes/{uuid}`	Soft-deletes the note (no dependent records)
`DELETE /v1/adhoc-checks/{uuid}`	Soft-deletes the adhoc check; for a group check, its child checks are deleted alongside it
`DELETE /v1/adhoc-id-checks/{uuid}`	Soft-deletes the adhoc ID check (no dependent records)

`PATCH` calls that change a `status` field (e.g. `program.status = "paused"`, `monitor.status = "paused"`) are not deletions; the affected record remains fully visible via the API.

Prefer archiving over deletion for compliance

If you only need to take a record out of day-to-day workflows but must keep it accessible for compliance, audit or historical reporting, **archive it instead of deleting it**. Archived records are not subject to the 30-day recovery window - they are retained indefinitely.

The differences:

	Soft delete	Archive
Returned by list / show endpoints	No - `404` on the UUID	Yes (filterable via the `archived` query parameter)
PDF / export still possible	No	Yes
Counts towards data-retention quotas	No	Yes
Recovery window	30 days, then eligible for permanent removal	Indefinite - the record is never auto-purged
Intended use case	The record was created in error or is no longer relevant	The record should be retained but is no longer in active use

Archive support per resource:

Resource	How to archive via API	API surfaces archive state?	Filter by archive state via API?
Entity	`POST /v1/entities/{uuid}/archive` (and `POST /v1/entities/{uuid}/unarchive` to restore)	Yes (`archived_at` field)	Yes (`?archived=true` / `?archived=false`)
Check	Inherited from the parent entity - archive the entity to archive its checks alongside it	Yes (`archived_at` field)	Yes (`?archived=true` / `?archived=false`)
Event	Inherited from the parent entity - archive the entity to archive its events alongside it	Yes (`archived_at` field)	Yes (`?archived=true` / `?archived=false`)
Note	Inherited from the parent entity - archive the entity to archive its notes alongside it	Yes (`archived_at` field)	Yes (`?archived=true` / `?archived=false`)

Archiving an entity also cascades to its identification checks and reports. Checks, events and notes do not have a per-record archive action - their archive state always follows the entity they belong to. Programs do not support archiving; if you need to take a program out of day-to-day workflows you can set its `status` to `paused` (scheduled monitor runs and event report emails will skip paused programs).

Reference numbers and soft-deleted entities

Entity `reference_number` values must be unique among **non-deleted** entities in the same program. Soft-deleted entities are excluded from that check.

Example: you `DELETE` an entity with `reference_number = "00-000-001"`. While that record remains soft-deleted (including throughout the 30-day recovery window), you may `POST` a new entity in the **same program** with `reference_number = "00-000-001"`. The API accepts the create because the deleted row no longer counts toward uniqueness.

The same `reference_number` may also be used on a **different program** while the soft-deleted copy still exists, because uniqueness is scoped per program.

After the recovery window, the soft-deleted record may be purged. Whether or not it has been purged, creating a new entity with that `reference_number` in the same program remains valid as long as no other non-deleted entity in that program already uses it.

Implications for integrators

Treat a `404` on a UUID you previously saw as "the record may have been deleted" rather than "the UUID was never valid". The 30-day recovery window gives you time to coordinate with the account owner if the deletion was unintended.

For how `reference_number` interacts with soft deletes, see Reference numbers and soft-deleted entities above.

If you need a confirmed-permanent removal (for example to satisfy a customer's data-erasure request), contact the Global Data support team. Permanent removal is a manual operation handled outside the API.

Monitor Configuration

A monitor's `config` object is per-operation: the shape depends on which screening operation the monitor's `monitor_check_type` refers to. This section lists the configuration keys for every operation type currently exposed by the API.

The same configuration is set by the corresponding portal screen at **Programs -> Program -> Monitors -> New / Edit**; integrators may find it useful to create one monitor in the portal first and `GET /v1/monitors/{uuid}` to see exactly what the API returns for an operation type before constructing a `POST` or `PATCH` payload of their own.

Common notes

On `POST /v1/programs/{program_uuid}/monitors`, every `required` key must be supplied. Missing keys produce a `400` with the relevant validation errors.

On `PATCH /v1/monitors/{monitor_uuid}`, the `config` object is merged with the existing config key-by-key. Only the keys you supply are validated and changed; keys you do not supply are preserved as-is.

Country codes are ISO 3166-1 alpha-2 (`AU`, `GB`), plus the synthetic values `ZZ` (Global / unattributed) and `EU` (European Union). The full list is in the API reference.

Some operations (e.g. PEP / sanction, banned & disqualified persons) accept an empty array to mean "all". This is documented per operation below.

pep_sanction_check - PEP & Sanction Screening

Key	Type	Required	Allowed values	Default
`search_type`	string	yes	`broad_search`, `medium_search`, `narrow_search`	`medium_search`
`similarity_threshold`	string	yes	`10`, `20`, `30`, `40`, `50`, `60`, `70`, `80`, `90`, `100`	`80`
`pep_countries`	array	no	Array of ISO 3166-1 alpha-2 country codes; empty array = global	`[]` (global)
`sanction_countries`	array	no	Array of ISO 3166-1 alpha-2 country codes; empty array = global	`["AU", "CA", "NZ", "GB", "US"]`
`max_results`	string	no	`10`, `15`, `20`, `25`, `50`, `100`	`25`
`pep_sanction_extended_result`	boolean	no	`true`, `false`	`true`

`search_type` controls the leniency of the name-matching algorithm. `narrow_search` returns fewer, higher-confidence results; `broad_search` returns more candidates including phonetic and partial matches.

`similarity_threshold` is expressed as a percentage string (not a number). Anything below the threshold is dropped before being returned.

`pep_sanction_extended_result` adds biographical and source metadata to each match. Disable to keep responses lean.

banned_disqualified_persons - Banned & Disqualified Persons Screening

Key	Type	Required	Allowed values	Default
`search_type`	string	yes	`broad_search`, `medium_search`, `narrow_search`	`medium_search`
`similarity_threshold`	string	yes	`10`, `20`, `30`, `40`, `50`, `60`, `70`, `80`, `90`, `100`	`80`
`banned_types`	array	no	See list below; empty array = all	`[]` (all)

`banned_types` may include any of:

Value	Meaning
`afs_banned_disqualified`	AFS-licensed financial service providers banned or disqualified under s. 922A(2) of the Corporations Act 2001
`banned_futures`	Banned futures representatives register (pre-AFS licences)
`banned_securities`	Banned securities representatives register (pre-AFS licences)
`credit_banned_disqualified`	Persons subject to a banning order or disqualification order under Part 2-4 of the NCCP Act
`disqualified_director`	Company directors and other office holders disqualified under s. 1274AA of the Corporations Act 2001
`disqualified_smsf`	Disqualified SMSF auditors per s. 130F of the SIS Act
`ato_disqualified_trustee`	Persons disqualified by the ATO from acting as the trustee of a self-managed superannuation fund

business_check - Australian Business Screening (ABN / ACN)

Key	Type	Required	Allowed values	Default
`name_match`	string	yes	`exact`, `similar`, `no`	`similar`
`abn_active`	string	yes	`yes` (alert if not active), `on_change`, `no`	`yes`
`acn_active`	string	yes	`yes` (alert if not active), `on_change`, `no`	`yes`
`gst_registered`	string	yes	`yes` (alert if not registered), `on_change`, `no`	`yes`
`recent_documents`	string	yes	`no`, `1`, `2`, `3`, `6`, `12` (months)	`3`
`recent_business_names`	string	yes	`no`, `1`, `2`, `3`, `6`, `12` (months)	`3`

The "months" keys are strings, not integers (`"3"` , not `3`). `no` means "do not alert on this signal".

`on_change` means alert whenever the value transitions between runs (e.g. ABN becomes inactive, or business address changes).

uk_business_check - UK Companies House Screening

Key	Type	Required	Allowed values	Default
`name_match`	string	yes	`exact`, `similar`, `no`	`similar`
`company_status`	string	yes	`yes` (alert if not active), `on_change`, `no`	`yes`
`recent_filings`	string	yes	`no`, `1`, `2`, `3`, `6`, `12` (months)	`3`
`officer_changes`	string	yes	`no`, `1`, `2`, `3`, `6`, `12` (months)	`3`
`registered_address_change`	string	yes	`on_change`, `no`	`on_change`
`sic_codes_change`	string	yes	`on_change`, `no`	`on_change`

court_check - Court Records Screening

Key	Type	Required	Allowed values	Default
`listing`	string	yes	`all`, `civil`, `criminal`	`criminal`
`search_party`	string	yes	`any`, `plaintiff`, `defendant`	`defendant`

realestate_check - Real Estate Screening

Key	Type	Required	Allowed values	Default
`date_from`	string	no	ISO 8601 date (`YYYY-MM-DD`), or `null` to consider all available history	`null`

adc_check - ADC Deceased Screening

This operation has no configuration keys. Send `"config": {}` (or omit the field entirely on create).

adc_custodian_check - Custodian Deceased Screening

This operation has no configuration keys. Send `"config": {}` (or omit the field entirely on create).

adverse_media - Adverse Media Screening

This operation has no configuration keys. Send `"config": {}` (or omit the field entirely on create).

email_check - Email Screening

This operation has no configuration keys. Send `"config": {}` (or omit the field entirely on create).

phone_check - Phone Screening

This operation has no configuration keys. Send `"config": {}` (or omit the field entirely on create).

ID Checks

Identification (ID) checks verify identity documents and related records against the Document Verification Service (DVS) and non-DVS providers. Unlike screening checks, ID checks run **synchronously** - launch endpoints return **200 OK** with the full completed check payload inline. There is no `status` field to poll.

Consent

Identification checks verify personal information against trusted sources, including the document issuer (DVS), superannuation and employment records, and other reliable sources. Australian privacy and AML/CTF obligations require that this only happens with the **express, demonstrable consent** of the individual being verified.

When you launch an ID check via the API, **your system - not WatchEye - is the system that interacts with the end user**. WatchEye therefore cannot collect or display consent on your behalf. Instead, every launch request must include `"consent_obtained": true`, which is your assertion that:

- You have presented consent wording to the individual being verified before submitting the check.
- The individual has accepted that wording.
- You have retained the exact wording shown, together with the time and identity of the individual, in your own system as evidence that consent was obtained.
- You can produce that record on request for audit and compliance purposes.

WatchEye records the launch with a generic consent attestation against the resulting check (visible on the portal and in the certificate PDF) noting that consent was obtained before the check was launched via the API and that the exact wording is retained by the account holder. The authoritative record of the wording shown to the individual lives in your system.

Your obligations vary by check type

`consent_obtained: true` is a single boolean on the wire, but the obligation it represents is substantial. The wording you must show and the records you must keep depend on which checks you run, on whom, and under which regulatory regime. **You** - not WatchEye - are the consent collector, the record-keeper, and the party accountable to regulators and to the data providers behind each check if the consent you obtained is ever audited.

The specific consent your integration needs depends on:

- **The check types you run.** DVS checks (`drivers_licence` , `passport` , `medicare` , `visa` , `asic_msic` , `aec` , `centrelink` , `immicard` , `citizenship_certificate` , `birth_certificate` , `death_certificate` , `marriage_certificate` , `name_change_certificate` , `registration_by_descent`) are governed by the **DVS Agreement** your account holds with the Australian Government. That agreement prescribes specific disclosure and record-

keeping requirements - your consent wording must align with it. Non-DVS checks (`globaldata`, `asic_id_check`, `payroll_super`) reach different data sources, each with their own provider-imposed consent requirements that you should review independently.

Your regulatory regime and jurisdiction. The Australian Privacy Principles, the AML/CTF Act and Rules, and any sector-specific obligations (AFSL or ACL holders, real estate agents under tranche 2 reforms, accountants, conveyancers, and so on) impose additional disclosure, record-keeping and consent-withdrawal requirements that go well beyond the API boolean.

Who is being verified and why. A primary customer verified for onboarding has different consent requirements from a beneficial owner verified against a customer's information, from an employee verified for payroll purposes, or from a deceased individual verified against a death certificate. Multi-purpose use of the verified data generally requires broader, more specific consent than a single-purpose check.

Your business case. If verification feeds into broader AML monitoring, employment decisioning, fraud-risk scoring or any other downstream use, the consent must disclose those uses up front - consent obtained for one purpose does not implicitly extend to another.

If you have any doubt about what wording or records your specific use case requires, **consult your DVS Agreement, your other data-provider agreements, and your own legal counsel before going live**.

Illustrative wording

The following text is provided as a **starting point only**. It is broadly suitable for a simple flow that runs DVS-based identity checks against an individual who is themselves the data subject, for identity verification, KYC and AML/CTF purposes. Using it verbatim does not, on its own, discharge your obligations under the DVS Agreement, the Privacy Act, the AML/CTF Act, or any other regime that applies to your business.

Review it against the check types you actually intend to run, your DVS Agreement, your other data-provider agreements, and your jurisdiction's privacy and AML obligations, and adapt or extend it accordingly before deploying it.

> This identity verification check requires your consent and can only proceed with your express permission. By providing your consent, you authorise our organisation and our identity verification provider to access and verify the personal information you have supplied against trusted sources, including the document issuer (DVS), superannuation and employment records, and other reliable sources. This information will be used for identity verification, KYC, AML and CTF purposes, in compliance with relevant laws and regulations.

If you intend to run non-DVS checks, or to verify individuals who are not themselves your primary customer, extend the wording to name the additional data sources (for example Global Data identity verification, ASIC ID search, payroll and superannuation records) and any additional purposes or recipient categories your business case requires.

Whatever you settle on, **store the exact text you showed**, not just a flag - regulators and DVS auditors examine the wording, not the boolean.

Submitting consent_obtained

`consent_obtained` is a required boolean on every ID check launch and must be `true`. A request with `consent_obtained: false` (or omitted) is rejected with `400`. Sending `true` without having obtained, recorded and retained consent that meets the obligations described above is a breach of the API terms and may also breach your DVS Agreement and other data-provider agreements.

Entity-bound vs adhoc

`GET/POST /v1/id-checks` and `POST /v1/entities/{uuid}/id-checks` - checks tied to a saved entity (include `program` and `entity` on list/show).

`GET/POST /v1/adhoc-id-checks` - quick checks with no entity record (include `user` or `api_key` attribution instead).

PDF certificates

Completed checks (`outcome` not `pending`) can be downloaded as verification certificate PDFs via `GET /v1/id-checks/{uuid}/pdf` and `GET /v1/adhoc-id-checks/{uuid}/pdf` .

Supported id_type values

All single-type checks are supported: DVS types (`drivers_licence` , `passport` , `medicare` , `visa` , `birth_certificate` , `centrelink` , `immicard` , `aec` , `asic_msic` , `citizenship_certificate` , `death_certificate` , `marriage_certificate` , `name_change_certificate` , `registration_by_descent`) and non-DVS types (`globaldata` , `asic_id_check` , `nz_drivers_licence` , `nz_passport` , `payroll_super`). MultiCheck (multiple ID types in a single launch) is not available via the API in v1.

Common fields and formats

The subsections below list the fields you submit inside the launch request's `data` object. The following conventions apply across all id types:

Names (`first_name` , `middle_name` , `last_name` , `full_name`): up to 50 characters by default (80 for Centrelink `full_name` , 100 for ASIC/MSIC `full_name`). Letters, spaces, apostrophes, hyphens and full stops only.

`birth_date` : `YYYY-MM-DD` .

`state` : one of `ACT` , `NSW` , `NT` , `QLD` , `SA` , `TAS` , `VIC` , `WA` . Lowercase values are accepted and uppercased automatically.

Other date fields (`card_expiry` , `acquisition_date` , `event_date` , `registration_date`): supply in `YYYY-MM-DD` format. Some fields such as Medicare and ASIC/MSIC `card_expiry` accept month-only values; submit those as `YYYY-MM` .

For checks that match against an entity, the `birth_date` you send must equal the entity's stored date of birth - otherwise the request is rejected.

For deeper background on each check (verification rules, supported document variants, sample images), follow the **Portal documentation** link in each subsection (portal login required).

aec

AEC (Australian Electoral Commission) enrolment verification (DVS). Portal documentation (login required).

Field	Type	Required	Notes
`first_name`	string	Yes	First name as recorded on the enrolment.
`middle_name`	string	No	If recorded on the enrolment.
`last_name`	string	Yes	

Field	Type	Required	Notes
`birth_date`	date `YYYY-MM-DD`	Yes	
`street_address`	string	Yes	Up to 255 characters, single line.
`suburb`	string	Yes	Up to 100 characters, single line.
`postcode`	string	Yes	4 digits.
`state`	string	Yes	AU state/territory code.

Requires the AEC feature on the calling account.

asic_msic

ASIC or MSIC photo identification card (DVS). Portal documentation (login required).

Field	Type	Required	Notes
`full_name`	string	Yes	Up to 100 characters.
`birth_date`	date `YYYY-MM-DD`	Yes	
`card_number`	string	Yes	6-20 alphanumeric characters.
`card_expiry`	date `YYYY-MM`	Yes	Month and year only.
`card_type`	string	Yes	`ASIC` or `MSIC`.

birth_certificate

Australian birth certificate (DVS). Portal documentation (login required).

Field	Type	Required	Notes
`first_name`	string	Conditional	Provide at least a first or last name.
`middle_name`	string	No	Requires `first_name` if supplied.
`last_name`	string	Conditional	Provide at least a first or last name.
`birth_date`	date `YYYY-MM-DD`	Yes	
`certificate_number`	string	Conditional	Provide one of `certificate_number`, `registration_number` or `registration_date`. 4-32 characters (`A-Z` , digits, `-`).
`registration_number`	string	Conditional	Same character rules as `certificate_number`.
`registration_date`	date `YYYY-MM-DD`	Conditional	
`state`	string	Yes	State of issue.

centrelink

Centrelink concession or healthcare card (DVS). Portal documentation (login required).

Field	Type	Required	Notes
`full_name`	string	Yes	Up to 80 characters.
`birth_date`	date `YYYY-MM-DD`	Yes	
`crn`	string	Yes	9 digits followed by 1 letter (hyphens optional).
`card_expiry`	date `YYYY-MM-DD`	Yes	
`card_type`	string	Yes	`HCC` (Health Care), `PCC` (Pensioner Concession) or `SHC` (Senior Health Care).

citizenship_certificate

Australian citizenship certificate (DVS). Portal documentation (login required).

Field	Type	Required	Notes
`first_name`	string	No	Required if `middle_name` is supplied.
`middle_name`	string	No	
`last_name`	string	Yes	
`birth_date`	date `YYYY-MM-DD`	Yes	
`acquisition_date`	date `YYYY-MM-DD`	Yes	Date citizenship was acquired.
`stock_number`	string	Yes	4-32 characters. Letters, digits, `/` and `-`.

death_certificate

Australian death certificate (DVS). Portal documentation (login required).

Field	Type	Required	Notes
`first_name`	string	Conditional	Provide at least a first or last name.
`middle_name`	string	No	Requires `first_name` if supplied.
`last_name`	string	Conditional	Provide at least a first or last name.
`event_date`	date `YYYY-MM-DD`	Yes	Date of death.
`certificate_number`	string	Conditional	Provide one of `certificate_number`, `registration_number` or `registration_date`. 4-32 characters (`A-Z`, digits, `-`).
`registration_number`	string	Conditional	Same character rules as `certificate_number`.
`registration_date`	date `YYYY-MM-DD`	Conditional	

Field	Type	Required	Notes
`state`	string	Yes	State of registration.

This check does not use `birth_date`; the deceased's date of death (`event_date`) is the relevant date.

drivers_licence

Australian driver's licence (DVS). Portal documentation (login required).

Field	Type	Required	Notes
`first_name`	string	Conditional	Provide at least a first or last name. Required if `middle_name` is supplied.
`middle_name`	string	No	
`last_name`	string	Conditional	Provide at least a first or last name.
`birth_date`	date `YYYY-MM-DD`	Yes	
`licence_number`	string	Yes	6-16 characters. Letters, digits, `-`.
`card_number`	string	Yes	6-16 characters. Letters, digits, `-`.
`state`	string	Yes	State of issue.

immicard

Department of Home Affairs ImmiCard (DVS). Portal documentation (login required).

Field	Type	Required	Notes
`first_name`	string	No	Required if `middle_name` is supplied.
`middle_name`	string	No	
`last_name`	string	Yes	
`birth_date`	date `YYYY-MM-DD`	Yes	
`card_number`	string	Yes	3 letters followed by 6 digits (for example `ABC123456`).

marriage_certificate

Australian marriage certificate (DVS). Portal documentation (login required).

Field	Type	Required	Notes
`first_name1`	string	Conditional	First spouse - provide at least a first or last name.
`middle_name1`	string	No	
`last_name1`	string	Conditional	First spouse - provide at least a first or last name.

Field	Type	Required	Notes
`first_name2`	string	Conditional	Second spouse - provide at least a first or last name.
`middle_name2`	string	No	
`last_name2`	string	Conditional	Second spouse - provide at least a first or last name.
`event_date`	date `YYYY-MM-DD`	Yes	Date of marriage.
`certificate_number`	string	Conditional	Provide one of `certificate_number`, `registration_number` or `registration_date`. 4-32 characters (`A-Z` , digits, `-`).
`registration_number`	string	Conditional	Same character rules as `certificate_number`.
`registration_date`	date `YYYY-MM-DD`	Conditional	
`state`	string	Yes	State of registration.

This check does not use `birth_date`. The two-party fields are suffixed `1` and `2`.

medicare

Medicare card (DVS). Portal documentation (login required).

Field	Type	Required	Notes
`card_type`	string	Yes	`G` (Green), `B` (Blue) or `Y` (Yellow).
`individual_ref_number`	integer	Yes	1-9 (the IRN printed beside the cardholder's name).
`full_name`	string	Yes	As printed on the card.
`birth_date`	date `YYYY-MM-DD`	Yes	
`medicare_number`	string	Yes	10 digits, optional spaces.
`card_expiry`	date `YYYY-MM`	Yes	Use `YYYY-MM-DD` for Blue and Yellow cards which carry a full expiry date.

name_change_certificate

Australian change-of-name certificate (DVS). Portal documentation (login required).

Field	Type	Required	Notes
`first_name`	string	Conditional	Provide at least a first or last name.
`middle_name`	string	No	Requires `first_name` if supplied.
`last_name`	string	Conditional	Provide at least a first or last name.

Field	Type	Required	Notes
`birth_date`	date `YYYY-MM-DD`	Yes	
`certificate_number`	string	Conditional	Provide one of `certificate_number`, `registration_number` or `registration_date`. 4-32 characters (`A-Z`, digits, `-`).
`registration_number`	string	Conditional	Same character rules as `certificate_number`.
`registration_date`	date `YYYY-MM-DD`	Conditional	
`state`	string	Yes	State of registration.

passport

Australian passport (DVS). Portal documentation (login required).

Field	Type	Required	Notes
`first_name`	string	No	Required if `middle_name` is supplied.
`middle_name`	string	No	
`last_name`	string	Yes	
`birth_date`	date `YYYY-MM-DD`	Yes	
`gender`	string	No	`M`, `F` or `X`.
`passport_number`	string	Yes	1-2 letters followed by 7 digits (for example `PA1234567`).

registration_by_descent

Australian Registration by Descent certificate (DVS). Portal documentation (login required).

Field	Type	Required	Notes
`first_name`	string	No	Required if `middle_name` is supplied.
`middle_name`	string	No	
`last_name`	string	Yes	
`birth_date`	date `YYYY-MM-DD`	Yes	
`acquisition_date`	date `YYYY-MM-DD`	Yes	Date the citizenship by descent was acquired.
`stock_number`	string	Yes	4-32 characters. Letters, digits, `/` and `-`.

visa

Foreign passport / visa entry (DVS). Portal documentation (login required).

Field	Type	Required	Notes
`first_name`	string	No	Required if `middle_name` is supplied.
`middle_name`	string	No	
`last_name`	string	Yes	
`birth_date`	date `YYYY-MM-DD`	Yes	
`passport_number`	string	Yes	6-9 alphanumeric characters.

asic_id_check

ASIC ID search against the ASIC database (non-DVS). Portal documentation (login required).

Field	Type	Required	Notes
`first_name`	string	Yes	
`middle_name`	string	No	
`last_name`	string	Yes	
`birth_date`	date `YYYY-MM-DD`	Yes	

globaldata

Global Data identity verification against multiple reliable data sources (non-DVS). Portal documentation (login required).

Field	Type	Required	Notes
`first_name`	string	Yes	
`middle_name`	string	No	
`last_name`	string	Yes	
`birth_date`	date `YYYY-MM-DD`	Yes	
`street_address`	string	No	Optional. If any address field is supplied, **all four** of `street_address`, `suburb`, `postcode` and `state` are required.
`suburb`	string	No	See `street_address`.
`postcode`	string	No	4 digits. See `street_address`.
`state`	string	No	AU state/territory code. See `street_address`.
`phone_number`	string	No	10-digit Australian phone number starting with `0`.
`email_address`	string	No	Valid email address.

nz_drivers_licence

NZ Drivers Licence verification against the New Zealand NZTA Driver Licence database (non-DVS). Portal documentation (login required).

Field	Type	Required	Notes
`first_name`	string	Yes	
`middle_name`	string	No	
`last_name`	string	Yes	
`birth_date`	date `YYYY-MM-DD`	Yes	
`licence_number`	string	Yes	2 letters followed by 6 digits (e.g. `DB123456`). Lowercase values are accepted and uppercased automatically. The number must have a valid check digit.
`licence_version`	string	Yes	The 3-digit version number from the front of the licence (e.g. `001`).

nz_passport

NZ Passport verification against the New Zealand Department of Internal Affairs (DIA) Passport database (non-DVS). Portal documentation (login required).

Field	Type	Required	Notes
`first_name`	string	Yes	
`middle_name`	string	No	
`last_name`	string	Yes	
`birth_date`	date `YYYY-MM-DD`	Yes	
`passport_number`	string	Yes	2 letters followed by 6 digits (e.g. `LA123456`); 1 letter followed by 7 digits is also accepted. Lowercase values are accepted and uppercased automatically.
`expiry_date`	date `YYYY-MM-DD`	Yes	The passport expiry date.

payroll_super

Payroll and superannuation match (non-DVS). Portal documentation (login required).

Field	Type	Required	Notes
`first_name`	string	Yes	
`middle_name`	string	No	
`last_name`	string	Yes	

Field	Type	Required	Notes
`birth_date`	date `YYYY-MM-DD`	Yes	
`street_address`	string	No	Optional. If any address field is supplied, **all four** of `street_address`, `suburb`, `postcode` and `state` are required.
`suburb`	string	No	See `street_address`.
`postcode`	string	No	4 digits. See `street_address`.
`state`	string	No	AU state/territory code. See `street_address`.
`phone_number`	string	No	10-digit Australian phone number starting with `0`.
`email_address`	string	No	Valid email address.
`employer_abn`	string	No	11 digits.

IDPass

An **IDPass** is a remote identity verification. Rather than you submitting document details for a synchronous DVS check (as the ID check endpoints do), an IDPass produces a secure hosted link that the individual opens themselves to photograph their documents, complete a liveness check and consent to the verification. Because a person has to interact with the hosted link, an IDPass is **asynchronous**: the launch call returns immediately and you poll for the result.

IDPasses can be launched in two ways:

- **Entity-bound** (`POST /v1/entities/{uuid}/idpasses`) - tied to a saved entity. The supplied `dob` must match the entity's date of birth.
- **Adhoc** (`POST /v1/adhoc-idpasses`) - a one-off verification with no entity record.

Both forms share the same configuration, the same response shape and the same polling, PDF and image endpoints described below.

Endpoints

Endpoint	Method	Purpose
`/v1/idpasses`	`GET`	List IDPasses on the account. Filter by `source` (`entity` / `adhoc`) and `status`.
`/v1/idpasses/{uuid}`	`GET`	Show a single IDPass with full configuration, status, document/validation summaries, activity log and image metadata. Poll this for the result.
`/v1/entities/{uuid}/idpasses`	`POST`	Launch an entity-bound IDPass. Returns `202 Accepted`.
`/v1/adhoc-idpasses`	`POST`	Launch an adhoc IDPass. Returns `202 Accepted`.
`/v1/idpasses/{uuid}/pdf`	`GET`	Download the verification certificate PDF (only once the IDPass is terminal).

Endpoint	Method	Purpose
`/v1/idpasses/{uuid}/images/{type}`	`GET`	Download a single verification image as raw bytes (only when `return_verification_images` was set).

Both launch endpoints are chargeable and accept the `Idempotency-Key` header. The product billed depends on how many document verification steps are configured. The PDF and image endpoints are not billed - they read from the on-record IDPass.

Manual correction of an IDPass result and cancellation of an in-flight IDPass remain **portal-only** functions; they are not exposed through the API.

Discoverability from ID checks

When an ID check was satisfied by an IDPass, the ID check show/list payloads (`/v1/id-checks/...` and `/v1/adhoc-id-checks/...`) embed a lightweight `idpass` shell - its `uuid` , `status` , `verification_status` and `idpass_link` - alongside an `idpass_url` pointing at the full IDPass resource. Follow `idpass_url` to `GET /v1/idpasses/{uuid}` for the complete record. Conversely, the IDPass detail payload carries an `id_check` block linking back to the originating check.

Lifecycle and status

The launch response is `202 Accepted` and carries the IDPass `uuid` , its `detail_url` , and the hosted `idpass_link` . Poll `GET /v1/idpasses/{uuid}` until `status` reaches a terminal value:

`status`	Meaning
`new`	Created; the individual has not opened the link yet.
`opened`	The individual has opened the hosted link.
`in_progress`	Documents are being submitted/processed.
`complete`	**Terminal.** Verification finished - read `verification_status` (`passed` , `review` or `failed`) for the outcome.
`expired`	**Terminal.** The link validity window elapsed before completion.
`failed`	**Terminal.** The verification could not be completed.
`cancelled`	**Terminal.** Cancelled in the portal.

`completed` is `true` once any terminal status is reached. A `review` outcome means the identity was verified but has been flagged for review in the portal - treat it as not yet passed until it has been looked at. Because completion depends on a person, polling can run for minutes to hours - poll every few seconds initially, then back off to a longer interval (e.g. every 30-60 seconds), and rely on `link_validity_days` for the upper bound.

Delivery

`config.delivery_method` controls how the hosted link reaches the individual:

- `manual` - the link is returned to you as `idpass_link` and you deliver it yourself.

`sms` - WatchEye sends the link by SMS to `config.delivery_phone` (an Australian mobile, `04XXXXXXXX`). The link is still returned as `idpass_link` as well.

Worked example

The example below launches an **adhoc** IDPass, polls for the result, then downloads the certificate and a verification image.

1. Launch the IDPass

```
curl -X POST https://portal.watcheye.com.au/api/v1/adhoc-idpasses \
  -H "Authorization: Bearer <key>|<secret>" \
  -H "Content-Type: application/json" \
  -H "Idempotency-Key: 7c3f9a1e-2b44-4f6a-9d2e-8a1b6c5d4e3f" \
  -d '{
    "data": {
      "first_name": "John",
      "last_name": "Smith",
      "dob": "1980-06-23"
    },
    "config": {
      "link_validity_days": 7,
      "check_liveness": true,
      "document_1_allowed_types": ["licence"],
      "document_2_allowed_types": ["medicare"],
      "require_id_photo": false,
      "return_verification_images": true,
      "delivery_method": "sms",
      "delivery_phone": "0412345678"
    }
  }'
```

The response is `202 Accepted`. The `data` block holds what exists at launch: the IDPass's identifier, configuration, delivery details, the hosted link and the ID check it belongs to. The verification outcome, summaries, log and images are not part of it - they come from `GET /v1/idpasses/{uuid}` once the individual has completed the link. Abridged, note the `uuid`, the `detail_url` to poll, and the hosted `idpass_link`:

```
{
  "data": {
    "uuid": "5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a",
    "source": "adhoc",
    "delivery_method": "sms",
    "idpass_link": "https://idpass.globaldata.net.au/idpass?token=abcd123456",
    "detail_url": "/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a"
  },
  "api_reference": "9f8e7d6c-..."
}
```

Because `delivery_method` is `sms`, the individual receives the `idpass_link` by text message. Treat `idpass_link` as opaque: pass it on exactly as returned and do not parse or rebuild it. (See the sandbox note below for how this behaves in sandbox.)

2. Poll for the result

```
curl -H "Authorization: Bearer <key>|<secret>" \
     https://portal.watcheye.com.au/api/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a
```

Keep polling until `status` is terminal. A completed pass looks like:


```
{
  "data": {
    "uuid": "5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a",
    "status": "complete",
    "completed": true,
    "verification_status": "passed",
    "return_verification_images": true,
    "images": [
      { "type": "id_photo", "title": "ID Photo", "mime_type": "image/jpeg", "url": "/v1/idpasses/5b1c7c8e-0b6e-4f
      { "type": "licence_front", "title": "Drivers Licence Front", "mime_type": "image/jpeg", "url": "/v1/idpasse
    ],
    "pdf_url": "/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a/pdf"
  },
  "api_reference": "..."
}
```


3. Download the certificate PDF

Once `status` is terminal, fetch the certificate. The PDF streams as raw bytes with a `Content-Disposition` filename, so write the body straight to a file:

```
curl -H "Authorization: Bearer <key>|<secret>" \
     https://portal.watcheye.com.au/api/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a/pdf \
     -o idpass-certificate.pdf
```

Requesting the PDF before the IDPass is terminal returns `409 Conflict` - keep polling and retry.

4. Download a verification image

When the IDPass was launched with `return_verification_images: true`, each entry in the `images` array can be fetched by its `type`. Images also stream as raw bytes (`image/jpeg`):

```
curl -H "Authorization: Bearer <key>|<secret>" \
     https://portal.watcheye.com.au/api/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a/images/id_photo \
     -o id-photo.jpg
```

Requesting a type that was not captured, or any image when `return_verification_images` was `false`, returns `404`.

Sandbox behaviour

IDPasses work end-to-end in sandbox using the same endpoints and response shapes as live, against the sandbox environment - so you can launch, poll and download without consuming live credit.

To exercise the hosted verification flow you open the `idpass_link` yourself and upload a synthetic identity document. The sample licences and passports to use, and the result each one produces, are listed under IDPass Test Documents (login required).

SMS delivery in sandbox

In sandbox, **no real SMS is sent.** When you launch a `sms`-delivery IDPass on a sandbox account, the message is recorded as sent but is not handed to the SMS gateway, so no text reaches the `delivery_phone`. The `idpass_link` is still returned in the launch response, so use that value to open the hosted flow during testing. On a live account the SMS is delivered normally.

Sandbox

WatchEye provides separate **sandbox** accounts for testing integrations without consuming live credit or touching live screening data. Sandbox is the recommended environment for all development work and continuous-integration runs.

What's different

Separate account. Sandbox is a distinct account with its own UUID and its own API keys. The environment (`live` or `sandbox`) is fixed at the account level - you cannot switch an existing key between environments with a query parameter, header or environment variable.

No billing. Calls made with a key on a sandbox account are not charged.

Test data only. Screening operations return canned or test-provider data, not real PEP, sanction or business records. The response shape is identical to live so that your integration code does not need to branch on environment - only the data behind it changes.

Same endpoints, same URL. Sandbox and live share `https://portal.watcheye.com.au/api/v1`. Which environment you are operating in is determined entirely by the account behind the API key you authenticate with.

How to tell which environment you are pointed at

`GET /v1/account` returns an `environment` field of either `live` or `sandbox`. This is the authoritative answer.

Getting a sandbox account

Sandbox accounts are provisioned by your Global Data account manager. Once your sandbox account exists, API keys on it are created and managed in the API Keys (login required) section of the sandbox portal in exactly the same way as keys on a live account.

Recommended workflow

Build and exercise your integration against a key on your sandbox account. Every endpoint, every response shape, every error envelope, idempotency and rate-limit behaviour mirrors live.

Run your happy-path and failure-path test suites against sandbox - as often as you like, with no billing impact.

Switch to a key on your live account for production. The only change in your code should be the credentials.

Sample sandbox test data

In sandbox mode the underlying data providers (DVS, ASIC, Companies House, etc.) are replaced with simulators that return deterministic results from a fixed set of test inputs. You can use any entity details you like to exercise the request and response plumbing, but to drive a particular outcome - a match, an explicit not-matched, a system error - you need to use one of the values listed below.

All of these values only produce results on keys belonging to a sandbox account. The same inputs sent to a live key reach the real upstream providers and return real (or no) data.

Screening check inputs

Sample data for screening checks is matched against the **entity** record (or, for adhoc checks, against the input payload). To exercise a particular outcome, create an entity with the matching details and then launch the relevant screening check against it.

pep_sanction_check

Names that produce a match in the sandbox PEP / sanction dataset:

Name	Birth date	Result
Ed Virgil Leuschke	1993-03-23	Sanctioned (AU DFAT)
Alford McGlynn	1994-03-13	Sanctioned (AU DFAT)
Justus Tanner Beatty	1950-06-04	PEP (QLD politician)
Julien Kovacek	2002-08-14	PEP (NSW senator)

Business / organisation matches:

Name	Sanction source
Southern Cross Commodities Pty Ltd	AU DFAT + US Trade CSL
Baltic Petrochem Logistics OU	EU + UK HMT
Jade Meridian Holdings Pte Ltd	US OFAC SDN + US Trade CSL
Andes Orbital Freight SAC	UK HMT
Sahel Mineral Ventures SARL	UN SC + AU DFAT

Any other name returns an empty result set.

banned_disqualified_persons

The banned and disqualified persons check covers two registers: the ASIC banned and disqualified persons registers (the six `banned_type` values shown below) and the ATO disqualified SMSF trustees register (`ato_disqualified_trustee`). Each of the eight sandbox names below has one record on the ASIC side and one record on the ATO side, so an unfiltered call typically returns two matches per name. Set `config.banned_disqualified_persons.banned_types` to one or more `banned_type` values (including `ato_disqualified_trustee`) to scope the search to specific registers.

Name	ASIC `banned_type`	Address
John Michael Smith	`afs_banned_disqualified`	Shepparton VIC
John Robert Smith	`banned_futures`	Sydney NSW
Jane Penny Sally Roberts	`banned_securities`	Camberwell VIC
Jane Roberts	`banned_securities`	Camberwell VIC
J S Roberts	`credit_banned_disqualified`	Melbourne VIC
P Zao	`disqualified_director`	Woolongong NSW
Simone N Zao	`disqualified_smsf`	(no address)
Adam Darren Halliday	`disqualified_director`	Illawong NSW

All three `search_type` values (`narrow_search` , `medium_search` , `broad_search`) are supported by the sandbox dataset. Any other name returns an empty result set.

business_check

The Australian business check is driven by the entity's ABN or ACN. The following sample businesses produce results in sandbox:

ABN	ACN	Name	Status	Notes
32111111114	111111114	Tech Innovators Ltd	Registered	Immediate result
11123456780	123456780	Green Energy Solutions	Deregistered	Historical extract
54222222228	222222228	Urban Development Corp	Registered	Relational extract
33234567894	234567894	Creative Media Agency	Registered	Delayed ~5s, exercises async polling
94782610486	782610486	Copper Finch Construction Pty Ltd	Registered	Risk: court case + insolvency notice
84655375652	655375652	Marble Kookaburra Interiors Pty Ltd	Registered	Risk: court case + adjudication (delayed)

Any other ABN / ACN returns an empty extract.

uk_business_check

Driven by the entity's UK company number:

Company number	Name	Status	Notes
12345678	TEST COMPANY LTD	active	Immediate
87654321	SAMPLE HOLDINGS PLC	active	Immediate
11223344	DORMANT SERVICES LTD	dormant	Immediate
55667788	DELAYED REPORT LTD	active	~2s delay, exercises async polling
99887766	DISSOLVED EXAMPLE LTD	dissolved	Immediate

Company number	Name	Status	Notes
SC654321	SCOTTISH ENTERPRISE SC	active	Scottish prefix

Any other company number returns "company not found".

court_check

Sandbox only returns matches for one person and one company:

Search input	Result
first name `Michael`, last name `Raymond` (optionally state `VIC`)	2 civil + 1 criminal listing in VIC
business name `Samplemart` (optionally state `VIC`)	1 civil + 1 criminal company listing in VIC

Any other name returns zero records.

realestate_check

Driven by the entity's address. The sandbox real-estate dataset is keyed by G-NAF ID; matching addresses must resolve to one of the IDs below.

G-NAF ID	Address	Listings
GAVIC421647320	20 Hardy St, Lilydale VIC 3140	3 records (sale 2018, rent 2016, sale 2012)
GANSW716615055	35 Yaralla St, Concord West NSW 2138	2 records (sale 2019, rent 2015)
GANSW704115400	17 Napier St, Binnaway NSW 2395	3 records (sales 2010, 2008, 2005)
GANSW704224437	375 Argent St, Broken Hill NSW 2880	2 records (sale 2018, rent 2016)

For a negative case, use G-NAF ID `GAVIC419608268` - the address resolves but has zero listings, which exercises the "address found, no listings" branch.

adc_check

The ADC (Australian deaths) check returns matches for these names. Lookups use first name + middle name + last name + birth date.

First	Middle	Last	Birth date	Result
John	(none)	Smith	1998-03-21	1 match - NSW, died 2023-07-24
John	(none)	Doe	1971-01-23	2 matches - NSW (died 2003-08-22) and WA (died 2011-11-03)
John	Andrew	Doe	1971-01-23	1 exact match - WA, died 2011-11-03
Mary	(none)	Smith	1965-02-17	1 match - NSW, date-of-death range late 2018 / early 2019

Any other input returns no matches.

adc_custodian_check

The custodian variant of the ADC check returns a simple deceased / not-deceased flag. Same backend data as `adc_check`, so the names above also match here. The search type adjusts based on what you supply: birth date + multi-word first name forces an exact-name match; birth date + single-word first name allows a partial-name match (useful for matching records that include a middle name); death date only matches name + date of death.

First	Last	Birth date	Death date	Result
John	Smith	1998-03-21	(omitted)	Deceased
John	Smith	1998-03-21	2023-07-24	Deceased (date of death matches)
John	Smith	1998-03-21	1999-01-01	Not deceased (date of death does not match)
John	Doe	1971-01-23	(omitted)	Deceased
John Andrew	Doe	1971-01-23	2011-11-03	Deceased (exact name + date of death match)
Mary	Smith	1965-02-17	(omitted)	Deceased

Any other name returns "not deceased".

adverse_media

Adverse media requires the entity's country to match the seeded record:

People:

First	Middle	Last	Birth date	Country
John	James	Doe	1990-01-01	AU
Jane	Karen	Smith	1995-02-18	AU
Liam	Raphael	O'Connor	1980-02-01	AU
Olivia	Rose	Smith	1973-06-06	GB

Businesses:

Name	Country
Acme Systems	AU
Sample Software Services	NZ

Any other input returns `has_adverse_media: false`.

email_check

The email-check sandbox is driven entirely by the email suffix - the local part can be anything:

Email suffix	Result
`.asn.au`	Simulated upstream failure (no result)
`.com.au`	Deliverable (after a ~20s simulated delay)
`.org.au`	Undeliverable - mailbox not found (after ~20s)
`.net.au`	Undetermined - greylisted (after ~20s)
`.edu.au`	Spamtrap (after ~20s)
`.org`	Undeliverable - mailbox not found
`.net`	Undetermined - greylisted
`.edu`	Spamtrap
anything else	Deliverable

Example: `test@example.org` returns Undeliverable; `slow@example.com.au` returns Deliverable after a ~20 second delay (useful for exercising your timeout handling).

phone_check

The phone-check sandbox uses a small seeded dataset. Phone numbers must be Australian and can be submitted in E.164 (`+61...`) or local (`0...`) form. Sample numbers that return data:

Phone number	Notes
`0427519643`	Person 1 - Mr John Andrew Smith
`0436991031`	Person 2 - Mr Robert Patrick Brown
`0755687356`	Person 3 - Ms Mary Sally Jones

These same people are also useful for global-data ID checks and other lookups backed by the same sandbox dataset - see the globaldata section below.

ID check inputs

ID checks against DVS providers all follow the same convention: number ranges drive the broad result, individual magic numbers drive specific not-matched reasons, and out-of-range numbers return a random result on each request.

DVS document number ranges

For every DVS document type listed below, the document / card / certificate number is checked against a numeric range first:

Number range	Outcome
`1...` (starts with 1)	MATCH
`2...` (starts with 2)	INVALID, with a random error reason

Number range	Outcome
`3...` (starts with 3)	NOT_MATCHED, with a random error reason

The per-document tables below list the specific ranges (for example `10000000` - `19999999` for driver licences) and the magic numbers within them that produce deterministic not-matched reasons.

> **Important:** if you submit a document number that falls outside any documented range and does not match a magic number, the sandbox returns a **random** result on each request. To get a deterministic failure use one of the specific magic numbers in the tables below; to get a deterministic match use any number in the `1...` range.

Last-name overrides

The following surname prefixes override the number-based outcome for any DVS check. They are matched case-insensitively against the surname field for the document type (`last_name` , `full_name` , or - for marriage certificates - either `last_name1` or `last_name2`).

Last-name prefix	Outcome
starts with `error...`	Simulated upstream system error
starts with `locked...`	"Document Temporarily Locked"

The prefix overrides take precedence over the document-number ranges, so `last_name = Errorsmith` returns a system error regardless of which document number you send.

drivers_licence

The `state` must be `VIC` for the sandbox to engage. Use the licence number as both `licence_number` and `card_number` .

Card number	Outcome
`10000000` - `19999999`	MATCH
`20000000` - `29999999`	INVALID (random reason)
`30000000` - `39999999`	NOT_MATCHED (random reason)
`77777770`	MATCH but "card number has not been checked"
`88888880`	NOT_MATCHED - given name does not match
`88888881`	NOT_MATCHED - middle name does not match
`88888882`	NOT_MATCHED - surname does not match
`88888883`	NOT_MATCHED - date of birth does not match
`88888884`	NOT_MATCHED - licence number does not match
`88888885`	NOT_MATCHED - card number not matched
`99999990`	INVALID - document invalid

passport

Passport number	Outcome
`M1000000` - `M1999999`	MATCH
`M2000000` - `M2999999`	INVALID (random reason)
`M3000000` - `M3999999`	NOT_MATCHED (random reason)
`M9999990`	INVALID - document invalid
`M9999991`	INVALID - document number does not match
`M9999992`	NOT_MATCHED - gender does not match
`M9999993`	NOT_MATCHED - date of birth does not match
`M9999994`	NOT_MATCHED - family name does not match
`M9999995`	NOT_MATCHED - given name(s) does not match

visa

The `passport_number` is the foreign passport number.

Passport number	Outcome
`M1000000` - `M1999999`	MATCH
`M2000000` - `M2999999`	INVALID (random reason)
`M3000000` - `M3999999`	NOT_MATCHED (random reason)
`M9999990`	NOT_MATCHED - travel document could not be found
`M9999991`	NOT_MATCHED - date of birth does not match
`M9999992`	NOT_MATCHED - family name does not match
`M9999993`	NOT_MATCHED - given and family name do not match

medicare

Medicare number	Outcome
`2000000020` - `2999999939`	MATCH
`3000000030` - `3999999949`	INVALID (random reason)
`4000000040` - `4999999959`	NOT_MATCHED (random reason)
`6000000060`	NOT_MATCHED - name does not match
`6000000061`	NOT_MATCHED - date of birth does not match
`6000000062`	NOT_MATCHED - IRN / card number does not match

Medicare number	Outcome
`6000000063`	NOT_MATCHED - expiry date does not match
`6000000064`	NOT_MATCHED - Medicare card number does not match

asic_msic

Card number	Outcome
`IC1000000` - `IC1999999`	MATCH
`IC2000000` - `IC2999999`	INVALID (random reason)
`IC3000000` - `IC3999999`	NOT_MATCHED (random reason)
`IC9999990`	INVALID - document invalid
`IC8888880`	NOT_MATCHED - name on card does not match
`IC8888881`	NOT_MATCHED - date of birth does not match
`IC8888882`	NOT_MATCHED - card number does not match
`IC8888883`	NOT_MATCHED - expiry date does not match

citizenship_certificate and registration_by_descent

Both use the same `stock_number` ranges.

Stock number	Outcome
`10000000000` - `19999999999`	MATCH
`20000000000` - `29999999999`	INVALID (random reason)
`30000000000` - `39999999999`	NOT_MATCHED (random reason)
`99999999990`	NOT_MATCHED - name or date of birth does not match
`99999999991`	NOT_MATCHED - document not found

immicard

ImmiCard number	Outcome
`ABC100000` - `ABC199999`	MATCH
`ABC200000` - `ABC299999`	INVALID (random reason)
`ABC300000` - `ABC399999`	NOT_MATCHED (random reason)
`ABC888880`	NOT_MATCHED - travel document could not be found
`ABC888881`	NOT_MATCHED - date of birth does not match

ImmiCard number	Outcome
`ABC888882`	NOT_MATCHED - family name does not match

centrelink

CRN matching uses the leading digit for the range and the trailing letter for magic-number outcomes.

CRN	Outcome
`100000000A` - `199999999Z`	MATCH
`200000000A` - `299999999Z`	INVALID (random reason)
`300000000A` - `399999999Z`	NOT_MATCHED (random reason)
`999999999A`	NOT_MATCHED - name does not match
`999999999B`	NOT_MATCHED - date of birth does not match
`999999999C`	NOT_MATCHED - CRN is invalid
`999999999D`	NOT_MATCHED - expiry date does not match
`999999999E`	NOT_MATCHED - card type does not match
`999999999F`	NOT_MATCHED - input card type not matched
`999999999G`	NOT_MATCHED - input name mismatch

aec

The AEC sandbox is driven by the leading house number of `street_address` and by `suburb` instead of a document number; the rest of the address can be anything.

Input	Outcome
`street_address` numeric prefix 101 - 200	MATCH
`street_address` numeric prefix 201 - 300	INVALID (random reason)
`street_address` numeric prefix 301 - 400	NOT_MATCHED (random reason)
`suburb` = `Sampleville`	NOT_MATCHED - address not known or invalid
`suburb` = `Testville`	NOT_MATCHED - name and/or date of birth not found at address

birth_certificate, marriage_certificate, death_certificate, name_change_certificate

The four BDM (Births, Deaths and Marriages) certificate checks share a common scheme but the certificate-number ranges depend on the issuing `state` and on which certificate it is.

Range plan by state:

State	Certificate number length	MATCH range	INVALID range	NOT_MATCHED range
ACT, NSW, NT, SA, VIC	7 digits	`1000000` - `1999999`	`2000000` - `2999999`	`3000000` - `3999999`
QLD	10 digits	`1000000000` - `1999999999`	`2000000000` - `2999999999`	`3000000000` - `3999999999`
TAS, WA	11 digits	`10000000000` - `19999999999`	`20000000000` - `29999999999`	`30000000000` - `39999999999`

Deterministic NOT_MATCHED magic numbers (return a fixed reason per state):

Birth, marriage and name-change certificates:

ACT, NSW, NT, SA, VIC: `9999990` - `9999994` (7 digits)

QLD: `9999999990` - `9999999994` (10 digits)

TAS, WA: `99999999990` - `99999999995` (11 digits)

Death certificates:

ACT, NSW, NT, SA: `9999990` - `9999994` (7 digits)

VIC: `99999990` - `99999993` (8 digits) - intentional outlier, follows the live VIC death-certificate format

QLD: `9999999990` - `9999999994` (10 digits)

TAS: `99999999990` - `99999999991` (11 digits)

WA: `99999999990` - `99999999995` (11 digits)

For VIC death certificates specifically: `99999990` = document not found; `99999991` = certificate number does not match; `99999992` = registration number does not match; `99999993` = given name does not match.

For marriage certificates the document records both parties but the request still carries a single `certificate_number`; the magic numbers apply to that one field regardless of which spouse's name is involved.

asic_id_check

The non-DVS ASIC ID check returns matches for these people:

Name	Birth date	State / suburb
John Michael Doe	1980-05-15	NSW / Sydney
Jane Alice Smith	1987-03-22	QLD / Brisbane
Lucas Benjamin Carter	1992-08-09	WA / Perth (delayed, exercises async polling)
Maria Scott	1971-11-02	VIC / Melbourne
Patrick O'Connor	1965-07-19	QLD / Brisbane
Mary-Jane Parker	1983-02-28	SA / Adelaide

Any other name returns no match.

globaldata

The Global Data identity check searches the same seeded sandbox dataset as the phone check, so any of the seeded people work as inputs. The sandbox outcome is keyed on `last_name`; supply the other fields the check requires (see ID Checks).

First	Middle	Last	Birth date	Result
John	Andrew	Smith	1998-03-21	High-confidence match (person 1)
Robert	Patrick	Brown	1971-03-18	Match (duplicates exist, useful for fuzzy-ranking tests)
Mary	Sally	Jones	1989-08-12	Match
Frederick	Brian	Oag	1983-10-12	Match (rare-surname test)
Pauline	Sandra	Payne	1969-09-06	Single match
Matthew	(none)	Smith	1999-10-21	Match
John	(none)	James	1962-06-11	Match

`last_name` of `Notarealsurname` (or any other unseeded surname) returns an empty match set.

nz_drivers_licence

The NZ Drivers Licence check verifies against a simulated NZTA database. Licence numbers must pass the check-digit validation regardless of the scenario being exercised.

Scenario	Inputs	Result
Verified match	Dana Mitchell, born 1965-09-13, licence `DB512036` version `001`	Pass - `match_status` `Match`, `document_verified` `true`
No match	Any other valid inputs	Fail - `match_status` `NoMatch`
No match with explanation	Licence `DB111112` (any name)	Fail - `NoMatch` with `additional_information` of `driversLicenceVersion does not match driversLicenceNo`
Source outage	Licence `DB000000` (any name)	Fail - `NoMatch`, document not verified
Validation error	Licence version `999`	`400` validation error response

nz_passport

The NZ Passport check verifies against a simulated DIA Passport database. Passport numbers must be in the valid NZ format regardless of the scenario being exercised.

Scenario	Inputs	Result
Verified match	Southland Stags, born 1980-12-12, passport `LZ115041` expiring 2017-10-10	Pass - `match_status` `Match`, `document_verified` `true`
No match	Any other valid inputs	Fail - `match_status` `NoMatch`

Scenario	Inputs	Result
No match with explanation	Passport `ZZ111111` (any name)	Fail - `NoMatch` with `additional_information` of `passportExpiry does not match passportNo`
Source outage	Passport `ZZ000000` (any name)	Fail - `NoMatch`, document not verified
Validation error	Expiry date `1900-01-01`	`400` validation error response

payroll_super

The non-DVS payroll & superannuation check returns matches for these people. Address, email, phone and employer-ABN inputs are also matched against the seeded record - matching values come back as `MATCHED`, anything else as `UNMATCHED`.

First	Middle	Last	Birth date	Super	Payroll
John	James	Doe	1990-01-01	MATCHED	MATCHED
Jane	Karen	Smith	1995-02-18	UNMATCHED	MATCHED
Fiona	(none)	Cheng	1982-06-28	MATCHED	UNMATCHED

Reference values for the matched paths (any field that does not match the value below comes back as `UNMATCHED`):

```
John James Doe (1990-01-01)
  4/123 Fake Street, Fakeville VIC 3987 AU
  john@example.com / 0400123456 / employer_abn 12345678901

Jane Karen Smith (1995-02-18)
  456 Wrong Avenue, Sampleville NSW 2785 AU
  jane@example.com / 0412345678 / employer_abn 98765432109

Fiona Cheng (1982-06-28)
  3 Right Avenue, Hiddenville WA 6824 AU
  fiona@example.com / 0400987654 / employer_abn 91237856482
```

To force a `500` upstream-failure response, submit `first_name` = `Service` and `last_name` = `Unavailable`.

ID Pass verifications are exercised with document images rather than field values: see IDPass Test Documents (login required) for the sample licences and passports and the result each one returns.

The WatchEye API gives programmatic access to everything in WatchEye: PEP and sanctions screening, identity checks, IDPass verifications, monitoring programs, events and reports. Every request is authenticated with an API key and secret sent as a bearer token. The API guide covers the conventions, idempotency, rate limits, versioning and the sandbox.

Servers

URL	Description
`https://portal.watcheye.com.au/api/v1`	Base URL for live and sandbox accounts

Authentication

BearerAuth: http (bearer) , format `api-key|api-secret`

Account

The calling account - its environment, balance and enabled products.

Show the calling account

GET `/account`

Returns the account that this API key belongs to, including the environment, the products the account can use, and the current billing balance.

This is the recommended first call for any new integration: it lets you confirm that your credentials work, that you are pointed at the expected environment (`live` or `sandbox`), and that the products you intend to use are enabled on the account.

Sub accounts

For a sub account API key:

- `parent_account_uuid` is populated with the parent (reseller) account's UUID.
- `balance` , `credit_limit` and `available_credit` are sourced from the parent account because that is the account charges are billed against.
- `products` is the intersection of the parent's products and the sub account's own products - that is, the products this sub account can actually use.

Responses

200 **Account response** (application/json)

Field	Description
`data`	**object**

Field	Description
data. uuid	**string (uuid)** The account's UUID **Example:** `7a1f0c8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. account_name	**string** The display name of the account **Example:** `Sample Account`
data. company_name	**string** The legal/company name on file for the account **Example:** `Sample Company Pty Ltd`
data. environment	**string** Which environment this account (and any keys on it) operate in: `live` - real screening data; calls bill against the account's balance `sandbox` - sandbox/test data; calls do not bill **Enum:** `live`, `sandbox` **Example:** `live`
data. is_reseller	**boolean** Whether this account is a reseller (i.e. has sub accounts under it). The Reseller functionality itself (managing sub accounts) is portal-only - this flag is exposed so integrators can detect that they are operating against a reseller account. **Example:** `false`
data. parent_account_uuid	**string (uuid) or null** The UUID of the parent account when this account is a sub account of a reseller. `null` for normal (top-level) accounts.
data. expires_at	**string (date-time) or null** ISO 8601 timestamp at which the account is scheduled to expire. `null` means no expiry. After the expiry date the API will reject calls with `401 Account Expired`. **Example:** `2026-12-31T00:00:00Z`
data. data_retention_days	**integer or null** Account-level data retention in days. `null` means the system default applies. Note: per-program retention is configured separately via `data_retention_months` on each program. **Example:** `365`

Field	Description
data. balance	**string** Current cash balance for the billing account, as a decimal string with 4 decimal places. For sub accounts this is the parent's balance because charges are billed to the parent. Negative values indicate the account has been billed beyond its cash balance and is consuming credit limit. **Example:** `1234.5678`
data. credit_limit	**string** Credit limit for the billing account, as a decimal string with 4 decimal places. Calls remain billable while `balance + credit_limit >= cost_of_call` ; once that sum drops below the cost, billable endpoints return `402 Insufficient Credit` . **Example:** `100.0000`
data. available_credit	**string** Convenience field equal to `balance + credit_limit` , expressed as a decimal string with 4 decimal places. This is the actual amount available to spend on billable calls. **Example:** `1334.5678`
data. products	**array of objects** The list of products this account can actually use. Each entry is a product the account is entitled to call billable endpoints against. For sub accounts, this is the intersection of the parent's products and the sub account's own product entitlements (a sub account cannot use a product the parent does not have, and the reseller can further restrict what the sub account can use).
data. products[]. name	**string** Stable machine-readable product key. Use this in code paths that decide which endpoints to call. **Example:** `pep_sanction_check`
data. products[]. label	**string** Human-readable product name. Suitable for display in dashboards. **Example:** `PEP and Sanction Check`
data. created_at	**string (date-time)** ISO 8601 timestamp at which the account was created **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** ISO 8601 timestamp at which the account was last updated **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "7a1f0c8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "account_name": "Sample Account",
        "company_name": "Sample Company Pty Ltd",
        "environment": "live",
        "is_reseller": false,
        "parent_account_uuid": null,
        "expires_at": "2026-12-31T00:00:00Z",
        "data_retention_days": 365,
        "balance": "1234.5678",
        "credit_limit": "100.0000",
        "available_credit": "1334.5678",
        "products": [
            {
                "name": "pep_sanction_check",
                "label": "PEP and Sanction Check"
            }
        ],
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 401, 403, 429, 5XX (see Common error responses).

Adhoc Checks

Screening checks run on supplied details, outside a program.

List adhoc checks

GET `/adhoc-checks`

Returns adhoc ("quick") checks for the calling account. Results are paginated.

An adhoc check is a one-off screening run performed from the portal by an account user against a free-form subject - they do not belong to a program or entity record. They use the same underlying screening operations as entity checks (PEP and Sanction, phone, business, etc.) so the typed `details` payload uses the same per-operation schemas.

The list endpoint returns a compact at-a-glance envelope per check; use the show adhoc check endpoint to read the typed `details` block and the uniform `results_overview` array for a specific record.

Group checks

A `check_type` value of `group` represents a parent record that aggregates one or more child adhoc checks (for example a quick check that runs a PEP screen and a phone check on the same subject in one go). Group checks appear in the list as first-class

records; use the `filter[parent_uuid]` filter to find their children, or fetch the group via show adhoc check to read the `children[]` summary.

Filters

The following filters can be applied via the `filter[...]` query parameters:

- `check_type` - one of the operation types (e.g. `pep_sanction_check`, `phone_check`, `business_check`) or the special value `group` for aggregate parent records
- `outcome` - `pass` or `warning`
- `created_by` - `user` (only checks launched by a portal user) or `api_key` (only checks launched by any API key on this account)
- `created_by_user_uuid` - only adhoc checks launched by the given portal user
- `created_by_api_key_uuid` - only adhoc checks launched by the given API key
- `parent_uuid` - only adhoc checks whose parent is the given check (use this with a group's uuid to list its children)
- `checked_after` - inclusive lower bound on `checked_at` (ISO 8601)
- `checked_before` - inclusive upper bound on `checked_at` (ISO 8601)

Filters that target a uuid (user, api key, parent) return an empty page when the target uuid does not exist on the calling account, identical to the behaviour for a uuid that does not exist at all.

Soft-deleted adhoc checks are excluded.

Adhoc checks do not support archiving - there is no `archived` query parameter and no `archived_at` field on the response.

Sorting

The `sort` query parameter accepts `checked_at` (default: `-checked_at`, newest first), `check_type` or `outcome`. Prefix with `-` for descending order.

Query parameters

Field	Description
`page`	**integer** The page of results to return, starting at 1. **Example:** `1`
`per_page`	**integer** The number of adhoc checks per page (defaults to 30, max 500) **Example:** `30`
`filter[check_type]`	**string** Only records of the given check type. The endpoint description lists the values. **Example:** `pep_sanction_check`

Field	Description
`filter[outcome]`	**string** Only records with the given outcome. The endpoint description lists the values. **Enum:** `pass`, `warning`
`filter[created_by]`	**string** Only records created by the given kind of actor: `user` (a portal user) or `api_key`. **Enum:** `user`, `api_key`
`filter[created_by_user_uuid]`	**string (uuid)** Only records created by the given portal user.
`filter[created_by_api_key_uuid]`	**string (uuid)** Only records created by the given API key.
`filter[parent_uuid]`	**string (uuid)** Only records whose parent is the given record. Use a group's uuid to list its children.
`filter[checked_after]`	**string (date-time)** Inclusive lower bound on `checked_at` (ISO 8601). **Example:** `2025-01-01T00:00:00Z`
`filter[checked_before]`	**string (date-time)** Inclusive upper bound on `checked_at` (ISO 8601). **Example:** `2025-12-31T23:59:59Z`
`sort`	**string** Field to sort by; prefix with `-` for descending order **Enum:** `checked_at`, `-checked_at`, `check_type`, `-check_type`, `outcome`, `-outcome` **Example:** `-checked_at`

Responses

200 **Adhoc checks list response** (application/json)

Field	Description
`data`	**array of objects** An array of adhoc checks
`data[].` `uuid`	**string (uuid)** The adhoc check's UUID **Example:** `6f9e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`

Field	Description
data[]. check_number	**string** A short obfuscated public identifier for the adhoc check (e.g. "Q-A1B2CD") **Example:** `Q-A1B2CD`
data[]. check_type	**string** The type of check that was run. One of the supported screening operations (e.g. `pep_sanction_check`, `phone_check`, `business_check`) or `group` for an aggregate record that owns one or more child checks. **Example:** `pep_sanction_check`
data[]. check_type_name	**string** Display name for the check type (e.g. "PEP and Sanction Screening", "Phone Check", "Group Check"). Suitable for showing in UIs. **Example:** `PEP and Sanction Screening`
data[]. status	**string** Lifecycle status of the adhoc check: `pending` - the check has been created and charged but the upstream call has not started yet `processing` - the upstream provider is being called `complete` - the check finished successfully and `outcome` is populated `failed` - the check could not be completed; `failed_reason` is populated and the credit charged at launch has been refunded For group adhoc checks the status is rolled up from the children: any child still pending or processing keeps the group at `processing`; the group is `failed` when any child failed, otherwise `complete` when every child is complete. **Enum:** `pending`, `processing`, `complete`, `failed` **Example:** `complete`
data[]. failed_reason	**string or null** Short reason why the check failed. Populated only when `status = failed`, otherwise `null`.

Field	Description
data[]. outcome	**string or null** High-level outcome of the check: `pass` - nothing flagged `warning` - one or more matches found that you should review `null` for group checks that have no child results yet, and for checks that have not reached `status = complete`. **Enum:** `pass`, `warning` **Example:** `warning`
data[]. check_summary	**string or null** Short, human-readable summary of the result (e.g. "Found 2 PEP matches", "All phones connected", "Business record exists"). `null` if the operation did not produce a summary. **Example:** `Found 2 PEP matches`
data[]. data_summary	**string or null** Short, human-readable summary of the subject the check ran against (e.g. "John A Smith, 1980-06-23" or "ACME Pty Ltd, ABN: 12 345 678 901"). For adhoc checks the value is captured at run time from the data the user typed into the quick-check form. `null` for group checks and for checks whose operation did not produce a subject summary. **Example:** `John A Smith, 1980-06-23`
data[]. checked_at	**string (date-time) or null** ISO 8601 timestamp at which the check was run. `null` while the check is still `pending` or `processing`. **Example:** `2025-01-01T00:00:00Z`
data[]. created_by	**object or null** Identifies who launched the adhoc check. The `type` discriminator picks the shape of the remaining fields: `{ type: "user", uuid, username }` - a portal user launched the adhoc check. `{ type: "api_key", uuid, label }` - an API key on this account launched the adhoc check. `null` - rare legacy rows from before actor attribution was recorded. If the user has since been deleted from the account the historical record is retained and the original `user` discriminator is still returned.
data[]. created_by. type	**string** Discriminator for the actor type. **Enum:** `user`, `api_key`

Field	Description
data[]. created_by. uuid	**string (uuid)** UUID of the user or API key that launched the check.
data[]. created_by. username	**string** Username of the portal user. Present only when `type` is `user`.
data[]. created_by. label	**string** Label of the API key. Present only when `type` is `api_key`.
data[]. parent_uuid	**string (uuid) or null** UUID of the parent (group) adhoc check this check is a child of. `null` when the check is not part of a group.
meta	**object**
meta. current_page	**integer** **Example:** `1`
meta. per_page	**integer** **Example:** `30`
meta. total	**integer** **Example:** `1`
meta. last_page	**integer** **Example:** `1`
api_reference	**string (uuid)**

Sample response


```
{
    "data": [
        {
            "uuid": "6f9e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
            "check_number": "Q-A1B2CD",
            "check_type": "pep_sanction_check",
            "check_type_name": "PEP and Sanction Screening",
            "status": "complete",
            "failed_reason": null,
            "outcome": "warning",
            "check_summary": "Found 2 PEP matches",
            "data_summary": "John A Smith, 1980-06-23",
            "checked_at": "2025-01-01T00:00:00Z",
            "created_by": {
                "type": "user",
                "uuid": "00000000-0000-0000-0000-000000000000",
                "username": "string",
                "label": "string"
```

```
            },
            "parent_uuid": null
        }
    ],
    "meta": {
        "current_page": 1,
        "per_page": 30,
        "total": 1,
        "last_page": 1
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 403, 429, 5XX (see Common error responses).

Launch adhoc screening checks

`POST` `/adhoc-checks`

Queues one or more "quick check" screening operations against a freeform subject supplied in the request body and returns `202 Accepted` with the parent (or single child) adhoc-check UUID. The screening runs in the background - poll `GET /v1/adhoc-checks/{uuid}` and wait for `status` to be `complete` or `failed` before reading the results.

Each requested check type is charged at launch time. If the upstream provider returns an error, the check is marked as `failed` and the credit for that individual check is refunded.

When a single check type is requested, the response describes that check. When two or more types are requested, a parent group adhoc check is created and returned, with one child per requested type.

Pre-flight checks

The endpoint validates the request before any check rows are created or any credit is charged:

- `400` - the request body failed validation (missing `data`, missing or unknown check type)
- `403 Forbidden` - the calling account is not entitled to one or more of the requested check types (the product is not enabled for the account)
- `402 Payment Required` - the calling account has insufficient credit to cover the total projected cost across all requested check types

Idempotency

This endpoint accepts the `Idempotency-Key` header so that retries on network failure are safe. See the Idempotency section of the API guide for the full contract.

Header parameters

Field	Description

`Idempotency-Key`	**string [max 255 characters]** Optional client-generated key (typically a UUID) that lets you safely retry a write request without risk of duplicate work or duplicate billing. A retry with the same key and body returns the original response verbatim, with the `Idempotent-Replay: true` response header. Honoured on `POST` requests only. See the Idempotency section of the API guide for the full contract. **Example:** `8c4d3a18-2f63-4f9a-9f3a-9b1f5a7c2d4f`

Request body

The adhoc check launch parameters

Field	Description
`check_types` REQUIRED	**array of strings [min 1 items]** One or more operation names to launch. See the operation enum in the AdhocCheck schema for the supported set.
`config`	**object** Per-operation configuration map keyed by operation name. Each listed operation has defaults that apply when its key is omitted, so you only need to send a block when you want to override a default. The five operations not listed below (`adverse_media` , `phone_check` , `email_check` , `adc_check` , `adc_custodian_check`) take no configuration.
`config.` `pep_sanction_check`	**object** Per-operation configuration for a `pep_sanction_check` launch. Every property is optional - any key you omit is filled with its default value before the check runs, and the resolved values are echoed back on the launch response so you can confirm what the check is using.

`config.`
`pep_sanction_check.`
`search_type`
	string
	Match strictness. `narrow_search` requires close name matches and returns the fewest false positives; `broad_search` returns more candidates for review; `medium_search` is the balanced default. Defaults to `medium_search` .
	Enum: `broad_search` , `medium_search` , `narrow_search`
	Example: `medium_search`

`config.`
`pep_sanction_check.`
`similarity_threshold`
	string
	Minimum name similarity percentage (10..100). Match candidates below this score are dropped. Sent as a string. Defaults to `"80"` .
	Enum: `10` , `20` , `30` , `40` , `50` , `60` , `70` , `80` , `90` , `100`
	Example: `80`

Field	Description
config. pep_sanction_check. pep_countries	**array of strings** Two-letter ISO 3166-1 alpha-2 country codes (either case is accepted; returned in uppercase) to scope the PEP search. An empty array searches all PEP jurisdictions (the default).
config. pep_sanction_check. sanction_countries	**array of strings** Two-letter ISO 3166-1 alpha-2 country codes (either case is accepted; returned in uppercase) to scope the sanction search. Defaults to `["AU", "CA", "NZ", "GB", "US"]`.
config. pep_sanction_check. max_results	**integer** Maximum number of match candidates to return. Sent as an integer. Defaults to `25`. **Enum:** `10`, `15`, `20`, `25`, `50`, `100` **Example:** `25`
config. pep_sanction_check. pep_sanction_extended_result	**boolean** When `true`, includes the extended biographical and source-list detail on each match. Defaults to `true`. **Example:** `true`
config. banned_disqualified_persons	**object** Per-operation configuration for a `banned_disqualified_persons` launch. Every property is optional - any key you omit is filled with its default value before the check runs.
config. banned_disqualified_persons. search_type	**string** Match strictness. `narrow_search` requires close name matches and returns the fewest false positives; `broad_search` returns more candidates for review; `medium_search` is the balanced default. Defaults to `medium_search`. **Enum:** `broad_search`, `medium_search`, `narrow_search` **Example:** `medium_search`
config. banned_disqualified_persons. similarity_threshold	**string** Minimum name similarity percentage (10..100), sent as a string. Defaults to `"80"`. **Enum:** `10`, `20`, `30`, `40`, `50`, `60`, `70`, `80`, `90`, `100` **Example:** `80`
config. banned_disqualified_persons. banned_types	**array of strings** Restrict the search to one or more specific banned/disqualified registers. An empty array (the default) searches every register. **Enum:** `afs_banned_disqualified`, `banned_futures`, `banned_securities`, `credit_banned_disqualified`, `disqualified_director`, `disqualified_smsf`, `ato_disqualified_trustee`

Field	Description
config. court_check	**object** Per-operation configuration for a `court_check` launch. Every property is optional - any key you omit is filled with its default value before the check runs.
config. court_check. listing	**string** Which court listings to search. `criminal` is the default; `civil` searches civil matters only; `all` searches both. **Enum:** `all`, `civil`, `criminal` **Example:** `criminal`
config. court_check. search_party	**string** Whether the subject should be matched as the `defendant`, `plaintiff`, or `any` party to the case. Defaults to `defendant`. **Enum:** `any`, `plaintiff`, `defendant` **Example:** `defendant`
config. business_check	**object** Per-operation configuration for a `business_check` launch. Every property is optional - any key you omit is filled with its default value before the check runs. Each property selects whether the corresponding business-record signal raises an alert (`yes`), is recorded for audit but does not raise an alert (`on_change`), or is skipped entirely (`no`); the recent-window properties select a months-back window instead.
config. business_check. name_match	**string** How strictly the trading name on the entity must match the registered ASIC name. Defaults to `similar`. **Enum:** `exact`, `similar`, `no` **Example:** `similar`
config. business_check. abn_active	**string** Alert when the ABN is not active. Defaults to `yes`. **Enum:** `yes`, `on_change`, `no` **Example:** `yes`
config. business_check. acn_active	**string** Alert when the ACN is not active. Defaults to `yes`. **Enum:** `yes`, `on_change`, `no` **Example:** `yes`

Field	Description
config. business_check. gst_registered	**string** Alert when GST registration changes. Defaults to `yes`. **Enum:** `yes`, `on_change`, `no` **Example:** `yes`
config. business_check. recent_documents	**string** Alert when ASIC documents have been lodged within the last N months. `no` disables the check; `1` .. `12` set the months-back window. Defaults to `"3"`. **Enum:** `no`, `1`, `2`, `3`, `6`, `12` **Example:** `3`
config. business_check. recent_business_names	**string** Alert when registered business names have changed within the last N months. `no` disables the check; `1` .. `12` set the months-back window. Defaults to `"3"`. **Enum:** `no`, `1`, `2`, `3`, `6`, `12` **Example:** `3`
config. uk_business_check	**object** Per-operation configuration for a `uk_business_check` launch. Every property is optional - any key you omit is filled with its default value before the check runs.
config. uk_business_check. name_match	**string** How strictly the trading name must match the Companies House record. Defaults to `similar`. **Enum:** `exact`, `similar`, `no` **Example:** `similar`
config. uk_business_check. company_status	**string** Alert when the company status is not `active`. Defaults to `yes`. **Enum:** `yes`, `on_change`, `no` **Example:** `yes`
config. uk_business_check. recent_filings	**string** Alert when filings have been lodged within the last N months. Defaults to `"3"`. **Enum:** `no`, `1`, `2`, `3`, `6`, `12` **Example:** `3`

Field	Description
config. uk_business_check. officer_changes	**string** Alert when officers have changed within the last N months. Defaults to `"3"`. **Enum:** `no`, `1`, `2`, `3`, `6`, `12` **Example:** `3`
config. uk_business_check. registered_address_change	**string** Whether to record registered address changes. Defaults to `on_change`. **Enum:** `on_change`, `no` **Example:** `on_change`
config. uk_business_check. sic_codes_change	**string** Whether to record SIC code changes. Defaults to `on_change`. **Enum:** `on_change`, `no` **Example:** `on_change`
config. realestate_check	**object** Per-operation configuration for a `realestate_check` launch. Every property is optional.
config. realestate_check. date_from	**string (date) or null** ISO 8601 date (`YYYY-MM-DD`). When set, only listings on or after this date are considered. Omit to search all available history. **Example:** `2024-01-01`
data REQUIRED	**object (dynamic)** The subject fields to screen. Required keys depend on the requested operation - see Adhoc check data fields for the full per-operation reference. Common fields include `first_name`, `last_name`, `dob`, `business_name`, `business_number`, `phone1`, `email1`. When multiple operations are launched together, every required field for every requested operation must be present in this shared `data` object. Validation runs before any check rows are created or any credit is charged; a missing or malformed field returns `400 Bad Request` identifying the offending key (`data.first_name`, etc.).

Sample request

```
{
    "check_types": [
        "pep_sanction_check"
    ],
    "config": {
        "pep_sanction_check": {
            "search_type": "medium_search",
            "similarity_threshold": "80",
```

```
            "pep_countries": [],
            "sanction_countries": [
                "AU",
                "CA",
                "NZ",
                "GB",
                "US"
            ],
            "max_results": 25,
            "pep_sanction_extended_result": true
        },
        "banned_disqualified_persons": {
            "search_type": "medium_search",
            "similarity_threshold": "80",
            "banned_types": []
        },
        "court_check": {
            "listing": "criminal",
            "search_party": "defendant"
        },
        "business_check": {
            "name_match": "similar",
            "abn_active": "yes",
            "acn_active": "yes",
            "gst_registered": "yes",
            "recent_documents": "3",
            "recent_business_names": "3"
        },
        "uk_business_check": {
            "name_match": "similar",
            "company_status": "yes",
            "recent_filings": "3",
            "officer_changes": "3",
            "registered_address_change": "on_change",
            "sic_codes_change": "on_change"
        },
        "realestate_check": {
            "date_from": "2024-01-01"
        }
    }
}
```

Responses

202 **Adhoc check accepted - the parent (or single child) adhoc check has been queued for processing. Poll `GET /v1/adhoc-checks/{uuid}` until `status` is `complete` or `failed`. The response shape depends on whether one check type or multiple were launched: * Single-check launch - the response describes that one check directly, with the resolved `config` block (defaults merged with user overrides) included so you can confirm exactly what the check is running with. `config` is `null` for operations that take no configuration. * Multi-check launch - the response describes the parent group check, with a `children[]` array carrying one entry per launched operation. Each child entry includes its own `uuid`, `status`, resolved `config`, and `detail_url` so you can start polling each child without first fetching the parent.**
(application/json)

Field	Description

`data`	**object**
`data.` `uuid`	**string (uuid)**
`data.` `check_number`	**string**
`data.` `check_type`	**string** The operation name when a single type was launched, or `group` when multiple types were launched and a parent check was created.
`data.` `status`	**string** **Enum:** `pending`, `processing`, `complete`, `failed` **Example:** `pending`
`data.` `config`	**object (dynamic) or null** The resolved configuration for the launched adhoc check (defaults merged with any user-supplied values). Present on single-check launches; `null` for operations that take no configuration. Omitted on group launches - see `children[]` for per-child configs.
`data.` `detail_url`	**string** Path to the adhoc check detail endpoint. **Example:** `/v1/adhoc-checks/3e36e9ec-0c44-4d65-a2f7-37b1b0a4d57e`
`data.` `children`	**array of objects** Returned on group launches (i.e. when `check_type = group`). One entry per launched operation, in the order they were requested. Each child has already been queued and can be polled independently via its `detail_url`.

`data.`
`children[].`
`uuid`
	string (uuid)

`data.`
`children[].`
`check_type`
	string
	Example: `court_check`

`data.`
`children[].`
`status`
	string
	Enum: `pending`, `processing`, `complete`, `failed`
	Example: `pending`

`data.`
`children[].`
`config`
	object (dynamic) or null
	The resolved configuration for this child (defaults merged with any user-supplied values). `null` for operations that take no configuration.

`data.`
`children[].`
`detail_url`
	string
	Example: `/v1/adhoc-checks/4d7e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`

`api_reference`	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "00000000-0000-0000-0000-000000000000",
        "check_number": "string",
        "check_type": "string",
        "status": "pending",
        "config": null,
        "detail_url": "/v1/adhoc-checks/3e36e9ec-0c44-4d65-a2f7-37b1b0a4d57e",
        "children": [
            {
                "uuid": "00000000-0000-0000-0000-000000000000",
                "check_type": "court_check",
                "status": "pending",
                "config": null,
                "detail_url": "/v1/adhoc-checks/4d7e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f"
            }
        ]
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


422 **Idempotency-Key was reused with a different request body** (application/json)

Field	Description
`message`	**string** **Example:** `Idempotency-Key was reused with a different request body.`

Sample response


```
{
    "message": "Idempotency-Key was reused with a different request body."
}
```


Standard error responses: 400, 401, 402, 403, 429, 5XX (see Common error responses).

Show an adhoc check

GET `/adhoc-checks/{adhoc_check_uuid}`

Returns a single adhoc check by its UUID, including:

- The base envelope (`created_by` , outcome, timestamps, etc.) - same shape as the list endpoint
- `results_overview[]` - one row per result, with a uniform `{title, status, alert_ids}` shape across all operation types
- `children[]` - per-child summaries when the check is a group; empty otherwise
- `details` - the typed per-operation details block. The exact shape depends on `check_type` and uses the same per-operation discriminated union as the entity check show endpoint. Group checks (`check_type = group`) always return `details: null`

- fetch each child via this endpoint to read its typed details.

PDF version

A printable PDF of the adhoc check (including subject, configuration and per-match details) is available via the PDF download endpoint.

Path parameters

Field	Description
`adhoc_check_uuid` REQUIRED	**string (uuid)** The adhoc check UUID

Responses

200 **Adhoc check response** (application/json)

Field	Description
`data`	**object** Same as AdhocCheck, plus the uniform `results_overview` and `children` arrays and the typed per-operation `details` block. Returned by the show endpoint so callers do not need a follow-up request to read match-level data.
`data.` `uuid`	**string (uuid)** The adhoc check's UUID **Example:** `6f9e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
`data.` `check_number`	**string** A short obfuscated public identifier for the adhoc check **Example:** `Q-A1B2CD`
`data.` `check_type`	**string** The type of check that was run (see AdhocCheck.check_type) **Example:** `pep_sanction_check`
`data.` `check_type_name`	**string** Display name for the check type **Example:** `PEP and Sanction Screening`
`data.` `status`	**string** Lifecycle status of the adhoc check (see AdhocCheck.status) **Enum:** `pending`, `processing`, `complete`, `failed` **Example:** `complete`

Field	Description
data. failed_reason	**string or null** Short reason why the check failed (see AdhocCheck.failed_reason)
data. outcome	**string or null** High-level outcome of the check (see AdhocCheck.outcome) **Enum:** `pass`, `warning` **Example:** `warning`
data. check_summary	**string or null** Short, human-readable summary of the result **Example:** `Found 2 PEP matches`
data. data_summary	**string or null** Short, human-readable summary of the subject the check ran against **Example:** `John A Smith, 1980-06-23`
data. checked_at	**string (date-time) or null** **Example:** `2025-01-01T00:00:00Z`
data. created_by	**object or null** Actor who launched the adhoc check - see AdhocCheck.created_by
data. created_by. type	**string** **Enum:** `user`, `api_key`
data. created_by. uuid	**string (uuid)**
data. created_by. username	**string** Present only when `type` is `user`.
data. created_by. label	**string** Present only when `type` is `api_key`.
data. parent_uuid	**string (uuid) or null**
data. results_overview	**array of objects** A uniform per-result summary across all operation types. Empty for group checks and for operations that pack everything into a single record (in which case `details.record` carries the full data).

Field	Description
data. results_overview[]. title	**string** Short label for the result, suitable for display in a list (e.g. "Match 1: John Smith - PEP", "0412 345 678 (Connected)", "5 records at 12 Main St, Sydney"). **Example:** `Match 1: John Smith - PEP`
data. results_overview[]. status	**string** Per-result status: `pass` - this individual result is clean `warning` - this individual result needs review (the default for any non-pass row when an operation does not assign a finer-grained status) `fail` - this individual result is a definitive failure (used by email and phone checks for undeliverable / disconnected entries) **Enum:** `pass`, `warning`, `fail` **Example:** `warning`
data. results_overview[]. alert_ids	**array of strings** Provider-side identifiers for the alert(s) this row generated. Use these to match up against `check_result_ids` on a related event from the events endpoint. Empty when the result is `pass`.
data. children	**array of objects** Summaries of the child adhoc checks for a group check. Empty for non-group checks. Fetch each child via the show adhoc check endpoint to get its full payload.
data. children[]. uuid	**string (uuid)** **Example:** `6f9e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. children[]. check_number	**string** **Example:** `Q-A1B2CD`
data. children[]. check_type	**string** **Example:** `pep_sanction_check`
data. children[]. check_type_name	**string** **Example:** `PEP and Sanction Screening`
data. children[]. status	**string** Lifecycle status of the child check (see AdhocCheck.status) **Enum:** `pending`, `processing`, `complete`, `failed` **Example:** `complete`
data. children[]. failed_reason	**string or null** Short reason why the child check failed (see AdhocCheck.failed_reason)

Field	Description
data. children[]. outcome	**string or null** **Enum:** `pass`, `warning` **Example:** `warning`
data. children[]. check_summary	**string or null** Short human-readable summary of the child adhoc check result (see AdhocCheck.check_summary). `null` while the child is pending / processing or when the child has no summary. **Example:** `Found 2 PEP matches`
data. children[]. checked_at	**string (date-time) or null** **Example:** `2025-01-01T00:00:00Z`
data. details	**one of 11 shapes** Typed per-operation details for the check. `null` for group checks. The exact shape depends on the `check_type` and uses the same per-operation schemas as the entity checks endpoint: `pep_sanction_check` - see CheckDetailsPepSanction `adverse_media` - see CheckDetailsAdverseMedia `adc_check` - see CheckDetailsAdc `adc_custodian_check` - see CheckDetailsAdcCustodian `banned_disqualified_persons` - see CheckDetailsBannedDisqualified `court_check` - see CheckDetailsCourt `email_check` - see CheckDetailsEmail `phone_check` - see CheckDetailsPhone `business_check` - see CheckDetailsBusiness `uk_business_check` - see CheckDetailsUkBusiness `realestate_check` - see CheckDetailsRealEstate

Field	Description

OBJECT 1

Field	Description
subject	**object**
subject. full_name	**string or null** **Example:** `John Smith`
subject. first_name	**string or null** **Example:** `John`
subject. middle_name	**string or null** **Example:** `Albert`
subject. last_name	**string or null** **Example:** `Smith`
subject. dob	**string or null** **Example:** `1980-06-23`
subject. business_name	**string or null**
subject. vessel_name	**string or null**
subject. address_country	**string or null** **Example:** `AU`
config	**object**
config. search_type	**string or null** **Enum:** `broad_search` , `medium_search` , `narrow_search` **Example:** `medium_search`
config. similarity_threshold	**integer or null** **Example:** `80`
config. pep_countries	**array of strings**
config. sanction_countries	**array of strings**
config. max_results	**integer or null** **Example:** `25`
config. extended_result	**boolean** **Example:** `true`
results	**array of objects**
results[]. uuid	**string or null** Provider-side identifier for the match. **Example:** `NK-12345`
results[]. name	**string or null** Canonical name on the matched record.

Field	Description
api_reference	string (uuid)

Sample response


```
{
    "data": {
        "uuid": "6f9e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "check_number": "Q-A1B2CD",
        "check_type": "pep_sanction_check",
        "check_type_name": "PEP and Sanction Screening",
        "status": "complete",
        "failed_reason": null,
        "outcome": "warning",
        "check_summary": "Found 2 PEP matches",
        "data_summary": "John A Smith, 1980-06-23",
        "checked_at": "2025-01-01T00:00:00Z",
        "created_by": {
            "type": "user",
            "uuid": "00000000-0000-0000-0000-000000000000",
            "username": "string",
            "label": "string"
        },
        "parent_uuid": "00000000-0000-0000-0000-000000000000",
        "results_overview": [
            {
                "title": "Match 1: John Smith - PEP",
                "status": "warning",
                "alert_ids": [
                    "NK-12345"
                ]
            }
        ],
        "children": [
            {
                "uuid": "6f9e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
                "check_number": "Q-A1B2CD",
                "check_type": "pep_sanction_check",
                "check_type_name": "PEP and Sanction Screening",
                "status": "complete",
                "failed_reason": null,
                "outcome": "warning",
                "check_summary": "Found 2 PEP matches",
                "checked_at": "2025-01-01T00:00:00Z"
            }
        ],
        "details": {
            "subject": {
                "full_name": "John Smith",
                "first_name": "John",
                "middle_name": "Albert",
                "last_name": "Smith",
                "dob": "1980-06-23",
                "business_name": null,
                "vessel_name": null,
                "address_country": "AU"
```

```
        },
        "config": {
            "search_type": "medium_search",
            "similarity_threshold": 80,
            "pep_countries": [],
            "sanction_countries": [
                "au",
                "nz",
                "gb",
                "us",
                "ca"
            ],
            "max_results": 25,
            "extended_result": true
        },
        "results": [
            {
                "uuid": "NK-12345",
                "name": "John Albert Smith",
                "matched_name": "J A Smith",
                "similarity": 0.95,
                "is_pep": true,
                "is_sanctioned": false,
                "pep_matches": [],
                "sanction_matches": [],
                "record": {
                    "id": "Q12345",
                    "caption": "John Albert Smith",
                    "schema": "Person",
                    "first_seen": "2018-01-01T00:00:00Z",
                    "last_seen": "2025-01-01T00:00:00Z",
                    "last_change": "2024-12-01T00:00:00Z"
                },
                "properties": {
                    "birth_date": [
                        "1980-06-23"
                    ],
                    "death_date": [],
                    "gender": [
                        "male"
                    ],
                    "nationality": [
                        "au"
                    ],
                    "citizenship": [
                        "au"
                    ],
                    "birth_place": [
                        "Sydney"
                    ],
                    "aliases": [
                        "J Smith",
                        "Johnny Smith"
                    ],
                    "positions": [
                        "Senator",
                        "Cabinet Minister"
                    ],
                    "phone": [],
```

```
                        "email": [],
                        "address": [
                            "12 Main St, Sydney NSW 2000"
                        ],
                        "education": [],
                        "religion": [],
                        "notes": [],
                        "wikidata_id": [
                            "Q12345"
                        ],
                        "source_urls": [
                            "https://example.com/source"
                        ],
                        "websites": []
                    }
                }
            ]
        }
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

Delete an adhoc check

DELETE `/adhoc-checks/{adhoc_check_uuid}`

Soft-deletes the adhoc check. For a group check (`check_type = group`), its child checks are deleted alongside it. See the Soft Deletion section of the API guide for the full policy (30-day recovery window, portal-only restore).

The records are no longer returned by list/show endpoints but remain in the database for compliance and audit purposes. The deleted adhoc check record is returned in the response as confirmation.

Path parameters

Field	Description
`adhoc_check_uuid` REQUIRED	string (uuid) The adhoc check UUID

Responses

200 **Adhoc check deleted (soft-delete - returned as confirmation)** (application/json)

Field	Description

`data`	**object** A one-off "quick check" run from the portal by an account user against a free-form subject, with no associated program or entity record. The base shape is the same on the list and detail endpoints; the detail endpoint additionally embeds the typed `details` block and the uniform `results_overview` and `children` arrays (see AdhocCheckDetail).
`data.` `uuid`	**string (uuid)** The adhoc check's UUID **Example:** `6f9e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
`data.` `check_number`	**string** A short obfuscated public identifier for the adhoc check (e.g. "Q-A1B2CD") **Example:** `Q-A1B2CD`
`data.` `check_type`	**string** The type of check that was run. One of the supported screening operations (e.g. `pep_sanction_check`, `phone_check`, `business_check`) or `group` for an aggregate record that owns one or more child checks. **Example:** `pep_sanction_check`
`data.` `check_type_name`	**string** Display name for the check type (e.g. "PEP and Sanction Screening", "Phone Check", "Group Check"). Suitable for showing in UIs. **Example:** `PEP and Sanction Screening`
`data.` `status`	**string** Lifecycle status of the adhoc check: `pending` - the check has been created and charged but the upstream call has not started yet `processing` - the upstream provider is being called `complete` - the check finished successfully and `outcome` is populated `failed` - the check could not be completed; `failed_reason` is populated and the credit charged at launch has been refunded For group adhoc checks the status is rolled up from the children: any child still pending or processing keeps the group at `processing`; the group is `failed` when any child failed, otherwise `complete` when every child is complete. **Enum:** `pending`, `processing`, `complete`, `failed` **Example:** `complete`
`data.` `failed_reason`	**string or null** Short reason why the check failed. Populated only when `status = failed`, otherwise `null`.

data. outcome	**string or null** High-level outcome of the check: `pass` - nothing flagged `warning` - one or more matches found that you should review `null` for group checks that have no child results yet, and for checks that have not reached `status = complete`. **Enum:** `pass`, `warning` **Example:** `warning`
data. check_summary	**string or null** Short, human-readable summary of the result (e.g. "Found 2 PEP matches", "All phones connected", "Business record exists"). `null` if the operation did not produce a summary. **Example:** `Found 2 PEP matches`
data. data_summary	**string or null** Short, human-readable summary of the subject the check ran against (e.g. "John A Smith, 1980-06-23" or "ACME Pty Ltd, ABN: 12 345 678 901"). For adhoc checks the value is captured at run time from the data the user typed into the quick-check form. `null` for group checks and for checks whose operation did not produce a subject summary. **Example:** `John A Smith, 1980-06-23`
data. checked_at	**string (date-time) or null** ISO 8601 timestamp at which the check was run. `null` while the check is still `pending` or `processing`. **Example:** `2025-01-01T00:00:00Z`
data. created_by	**object or null** Identifies who launched the adhoc check. The `type` discriminator picks the shape of the remaining fields: `{ type: "user", uuid, username }` - a portal user launched the adhoc check. `{ type: "api_key", uuid, label }` - an API key on this account launched the adhoc check. `null` - rare legacy rows from before actor attribution was recorded. If the user has since been deleted from the account the historical record is retained and the original `user` discriminator is still returned.
data. created_by. type	**string** Discriminator for the actor type. **Enum:** `user`, `api_key`
data. created_by. uuid	**string (uuid)** UUID of the user or API key that launched the check.

data. created_by. username	**string** Username of the portal user. Present only when `type` is `user`.
data. created_by. label	**string** Label of the API key. Present only when `type` is `api_key`.
data. parent_uuid	**string (uuid) or null** UUID of the parent (group) adhoc check this check is a child of. `null` when the check is not part of a group.
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "6f9e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "check_number": "Q-A1B2CD",
        "check_type": "pep_sanction_check",
        "check_type_name": "PEP and Sanction Screening",
        "status": "complete",
        "failed_reason": null,
        "outcome": "warning",
        "check_summary": "Found 2 PEP matches",
        "data_summary": "John A Smith, 1980-06-23",
        "checked_at": "2025-01-01T00:00:00Z",
        "created_by": {
            "type": "user",
            "uuid": "00000000-0000-0000-0000-000000000000",
            "username": "string",
            "label": "string"
        },
        "parent_uuid": null
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

Download an adhoc check as PDF

GET `/adhoc-checks/{adhoc_check_uuid}/pdf`

Returns a printable PDF report for the adhoc check, suitable for inclusion in customer-facing files. The PDF mirrors the on-screen quick-check report in the portal: subject snapshot, configuration in force, per-match details and a results summary.

The response is `application/pdf` with a `Content-Disposition` header naming the file after the check's `check_number` (e.g. `adhoc-check-Q-A1B2CD.pdf`).

This endpoint does not bill - PDF generation reuses the existing on-record check data.

> **Note:** PDF rendering is performed on demand and typically takes between 10 and 60 seconds depending on the size of the check (number of matches, embedded fonts, etc.). Increase your HTTP client's read / response timeout accordingly - the default 30 second timeouts in most libraries (curl, Guzzle, axios, requests) are not enough for the larger reports.

Path parameters

Field	Description
`adhoc_check_uuid` REQUIRED	**string (uuid)** The adhoc check UUID

Responses

200 **PDF report** (application/pdf)

Binary PDF file

Standard error responses: 401, 403, 404, 429, 5XX (see Common error responses).

Adhoc ID Checks

Identity document checks run on supplied details, outside a program.

List adhoc ID checks

GET `/adhoc-id-checks`

Returns quick (adhoc) identification checks for the calling account. These are one-off checks not tied to a program or entity record.

Actor attribution is via the `created_by` discriminator (`user` for a portal launch or `api_key` for an API launch). Filter by `created_by` , `created_by_user_uuid` or `created_by_api_key_uuid` to narrow results.

Query parameters

Field	Description
`page`	**integer** The page of results to return, starting at 1. **Example:** `1`
`per_page`	**integer** The number of records per page (defaults to 30, max 500). **Example:** `30`

Field	Description
`filter[id_type]`	**string** Only ID checks of the given document type.
`filter[check_type]`	**string** Only records of the given check type. The endpoint description lists the values.
`filter[outcome]`	**string** Only records with the given outcome. The endpoint description lists the values. **Enum:** `pass`, `fail`, `pending`
`filter[created_by]`	**string** Only records created by the given kind of actor: `user` (a portal user) or `api_key`. **Enum:** `user`, `api_key`
`filter[created_by_user_uuid]`	**string (uuid)** Only records created by the given portal user.
`filter[created_by_api_key_uuid]`	**string (uuid)** Only records created by the given API key.
`filter[checked_after]`	**string (date-time)** Inclusive lower bound on `checked_at` (ISO 8601).
`filter[checked_before]`	**string (date-time)** Inclusive upper bound on `checked_at` (ISO 8601).
`sort`	**string** The field to sort by, prefixed with `-` for descending order. The endpoint description lists the supported fields. **Example:** `-checked_at`

Responses

200 **Paginated adhoc ID check list** (application/json)

Field	Description
`data`	**array of objects**
`data[].` `uuid`	**string (uuid)**
`data[].` `check_number`	**string**
`data[].` `id_type`	**string**

Field	Description
data[]. id_type_name	**string**
data[]. check_type	**string**
data[]. check_type_name	**string**
data[]. outcome	**string** **Enum:** `pass`, `fail`, `pending`
data[]. outcome_summary	**string or null**
data[]. data_summary	**string or null**
data[]. checked_at	**string (date-time) or null**
data[]. created_by	**object or null** Identifies who launched the adhoc ID check. The `type` discriminator picks the shape of the remaining fields: `{ type: "user", uuid, username }` - a portal user launched the check. `{ type: "api_key", uuid, label }` - an API key on this account launched the check. `null` - rare legacy rows from before actor attribution was recorded.
data[]. created_by. type	**string** Discriminator for the actor type. **Enum:** `user`, `api_key`
data[]. created_by. uuid	**string (uuid)** UUID of the user or API key that launched the check.
data[]. created_by. username	**string** Username of the portal user. Present only when `type` is `user`.
data[]. created_by. label	**string** Label of the API key. Present only when `type` is `api_key`.
data[]. idpass	**object** Present only when `id_type` is `idpass`. A lightweight shell of the linked IDPass plus `idpass_url`, the path to the full IDPass resource.
data[]. idpass. uuid	**string (uuid)** The IDPass identifier. Use it with the show, image and PDF endpoints. **Example:** `5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`

Field	Description
data[]. idpass. status	**string** Where the individual is in the hosted flow: `new` - the link has not been opened `opened` - the welcome page was passed `in_progress` - consent was given and documents are being submitted `complete` - the verification finished; read `verification_status` for the outcome `expired` - `link_validity_days` elapsed before completion `failed` - the verification could not be completed `cancelled` - cancelled in the portal The last four are terminal. See the IDPass section of the guide. **Enum:** `new`, `opened`, `in_progress`, `complete`, `expired`, `failed`, `cancelled` **Example:** `in_progress`
data[]. idpass. verification_status	**string or null** The outcome once `status` is terminal, null until then: `passed` - the identity was verified `review` - the identity was verified but flagged for review in the portal; treat it as not yet passed `failed` - the identity was not verified (including an expired or cancelled IDPass) **Enum:** `passed`, `review`, `failed` **Example:** `passed`
data[]. idpass. idpass_link	**string or null** The hosted link the individual opens to complete their verification. Treat it as opaque - pass it on as-is and do not parse or rebuild it. **Example:** `https://idpass.globaldata.net.au/idpass?token=abcd123456`
data[]. idpass. idpass_url	**string** Path to the full IDPass resource (GET /v1/idpasses/{uuid}). **Example:** `/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
meta	**object**
meta. current_page	**integer**
meta. per_page	**integer**
meta. total	**integer**
meta. last_page	**integer**

Field	Description
`api_reference`	**string (uuid)**

Sample response


```
{
    "data": [
        {
            "uuid": "00000000-0000-0000-0000-000000000000",
            "check_number": "string",
            "id_type": "string",
            "id_type_name": "string",
            "check_type": "string",
            "check_type_name": "string",
            "outcome": "pass",
            "outcome_summary": "string",
            "data_summary": "string",
            "checked_at": "2024-01-01T00:00:00Z",
            "created_by": {
                "type": "user",
                "uuid": "00000000-0000-0000-0000-000000000000",
                "username": "string",
                "label": "string"
            },
            "idpass": {
                "uuid": "5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a",
                "status": "in_progress",
                "verification_status": "passed",
                "idpass_link": "https://idpass.globaldata.net.au/idpass?token=abcd123456",
                "idpass_url": "/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a"
            }
        }
    ],
    "meta": {
        "current_page": 0,
        "per_page": 0,
        "total": 0,
        "last_page": 0
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 401, 403, 429, 5XX (see Common error responses).

Launch an adhoc ID check

POST `/adhoc-id-checks`

Runs a synchronous quick ID check and returns **200 OK** with the full completed payload. See launch entity ID check for consent, idempotency and billing behaviour.

Header parameters

Field	Description
`Idempotency-Key`	**string [max 255 characters]** Optional client-generated key (typically a UUID) that lets you safely retry a write request without risk of duplicate work or duplicate billing. A retry with the same key and body returns the original response verbatim, with the `Idempotent-Replay: true` response header. Honoured on `POST` requests only. See the Idempotency section of the API guide for the full contract. **Example:** `8c4d3a18-2f63-4f9a-9f3a-9b1f5a7c2d4f`

Request body

Field	Description
`id_type` **REQUIRED**	**string** Identification type to run. See SELECTABLE_ID_TYPES in the portal field reference. **Example:** `drivers_licence`
`consent_obtained` **REQUIRED**	**boolean** Must be `true`. Asserts that your system has obtained express, demonstrable consent from the individual being verified before this check is launched, and that the exact wording shown has been retained by your system for audit and compliance purposes. The specific wording you must show and the records you must keep depend on which check types you run and the regulatory regime you operate under. For DVS-based check types your consent must align with your account's DVS Agreement with the Australian Government, which prescribes specific disclosure and record-keeping requirements. See the Consent section of the API guide for your full obligations and an illustrative starting point for the wording.
`data` **REQUIRED**	**object** Per-type input fields inside the launch request. Document numbers are never returned in responses. Date of birth must be supplied as `birth_date` in `YYYY-MM-DD` format when required for the selected `id_type`. Other date fields (`card_expiry`, `acquisition_date`, `event_date`, `registration_date`) also use `YYYY-MM-DD`; month-only fields (e.g. Medicare and ASIC/MSIC `card_expiry`) use `YYYY-MM`. See the per-id_type tables in the API guide for the exact fields and formats accepted by each check.
`data.` `birth_date`	**string (date)** Date of birth in `YYYY-MM-DD` format. **Example:** `1980-06-23`

Field	Description
oac	**string or null** The DVS OAC code to use for this check. Must be one of the OAC codes configured on your account. Required when your account is configured with more than one OAC. When your account has a single OAC this may be omitted and that OAC is used. **Example:** `ABC123`

Sample request

```
{
    "id_type": "drivers_licence",
    "consent_obtained": true,
    "data": {
        "birth_date": "1980-06-23"
    },
    "oac": "ABC123"
}
```

Responses

200 **Adhoc ID check completed** (application/json)

Field	Description
data	**object** A quick (adhoc) identification check not tied to an entity.
data. uuid	**string (uuid)**
data. check_number	**string**
data. id_type	**string**
data. id_type_name	**string**
data. check_type	**string**
data. check_type_name	**string**
data. outcome	**string** **Enum:** `pass`, `fail`, `pending`
data. outcome_summary	**string or null**
data. data_summary	**string or null**

Field	Description
data. checked_at	**string (date-time) or null**
data. created_by	**object or null** Identifies who launched the adhoc ID check. The `type` discriminator picks the shape of the remaining fields: `{ type: "user", uuid, username }` - a portal user launched the check. `{ type: "api_key", uuid, label }` - an API key on this account launched the check. `null` - rare legacy rows from before actor attribution was recorded.
data. created_by. type	**string** Discriminator for the actor type. **Enum:** `user`, `api_key`
data. created_by. uuid	**string (uuid)** UUID of the user or API key that launched the check.
data. created_by. username	**string** Username of the portal user. Present only when `type` is `user`.
data. created_by. label	**string** Label of the API key. Present only when `type` is `api_key`.
data. idpass	**object** Present only when `id_type` is `idpass`. A lightweight shell of the linked IDPass plus `idpass_url`, the path to the full IDPass resource.
data. idpass. uuid	**string (uuid)** The IDPass identifier. Use it with the show, image and PDF endpoints. **Example:** `5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`

Field	Description
data. idpass. status	**string** Where the individual is in the hosted flow: `new` - the link has not been opened `opened` - the welcome page was passed `in_progress` - consent was given and documents are being submitted `complete` - the verification finished; read `verification_status` for the outcome `expired` - `link_validity_days` elapsed before completion `failed` - the verification could not be completed `cancelled` - cancelled in the portal The last four are terminal. See the IDPass section of the guide. **Enum:** `new`, `opened`, `in_progress`, `complete`, `expired`, `failed`, `cancelled` **Example:** `in_progress`
data. idpass. verification_status	**string or null** The outcome once `status` is terminal, null until then: `passed` - the identity was verified `review` - the identity was verified but flagged for review in the portal; treat it as not yet passed `failed` - the identity was not verified (including an expired or cancelled IDPass) **Enum:** `passed`, `review`, `failed` **Example:** `passed`
data. idpass. idpass_link	**string or null** The hosted link the individual opens to complete their verification. Treat it as opaque - pass it on as-is and do not parse or rebuild it. **Example:** `https://idpass.globaldata.net.au/idpass?token=abcd123456`
data. idpass. idpass_url	**string** Path to the full IDPass resource (GET /v1/idpasses/{uuid}). **Example:** `/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
data. consent	**object (dynamic)**
data. verification_details	**array of objects**
data. verification_details[]. field	**string**
data. verification_details[]. value	**string**

Field	Description
data. details	one of 6 shapes

Field	Description

OBJECT 1

Field	Description
verification_result_code	string Enum: `Y`, `N`, `D`
originating_agency_code	string or null
verification_request_number	string or null
additional_information	array of strings

OBJECT 2

Field	Description
api_reference	string or null
message	string or null
match_results	object or null
match_results. first_name	string
match_results. last_name	string
match_results. dob	string Date of birth match result from the upstream provider.

OBJECT 3

Field	Description
api_reference	string or null
person_match	string or null

OBJECT 4

Field	Description
api_reference	string or null
reporting_reference	string or null Unique transaction identifier from the data source.
match_status	string or null Overall match result. `Match` indicates the supplied identity was fully verified. Enum: `Match`, `NoMatch`
document_verified	boolean Whether NZTA reports the supplied licence details as verified.
additional_information	string or null Explanatory message returned by the data source, typically alongside a `NoMatch` result.

Field	Description
api_reference	string (uuid)

Sample response


```
{
    "data": {
        "uuid": "00000000-0000-0000-0000-000000000000",
        "check_number": "string",
        "id_type": "string",
        "id_type_name": "string",
        "check_type": "string",
        "check_type_name": "string",
        "outcome": "pass",
        "outcome_summary": "string",
        "data_summary": "string",
        "checked_at": "2024-01-01T00:00:00Z",
        "created_by": {
            "type": "user",
            "uuid": "00000000-0000-0000-0000-000000000000",
            "username": "string",
            "label": "string"
        },
        "idpass": {
            "uuid": "5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a",
            "status": "in_progress",
            "verification_status": "passed",
            "idpass_link": "https://idpass.globaldata.net.au/idpass?token=abcd123456",
            "idpass_url": "/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a"
        },
        "verification_details": [
            {
                "field": "string",
                "value": "string"
            }
        ],
        "details": {
            "verification_result_code": "Y",
            "originating_agency_code": "string",
            "verification_request_number": "string",
            "additional_information": [
                "string"
            ]
        }
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


422 **Idempotency-Key was reused with a different request body** (application/json)

Field	Description
message	string

Sample response

```
{
    "message": "string"
}
```

Standard error responses: 400, 401, 402, 403, 429, 5XX (see Common error responses).

Show an adhoc ID check

GET `/adhoc-id-checks/{adhoc_id_check_uuid}`

Returns a single adhoc ID check by UUID, including consent, verification details and the typed `details` block.

Path parameters

Field	Description
`adhoc_id_check_uuid` REQUIRED	**string (uuid)** The uuid of the adhoc ID check.

Responses

200 **Adhoc ID check response** (application/json)

Field	Description
`data`	**object** A quick (adhoc) identification check not tied to an entity.
`data.` `uuid`	**string (uuid)**
`data.` `check_number`	**string**
`data.` `id_type`	**string**
`data.` `id_type_name`	**string**
`data.` `check_type`	**string**
`data.` `check_type_name`	**string**
`data.` `outcome`	**string** **Enum:** `pass`, `fail`, `pending`

Field	Description
data. outcome_summary	**string or null**
data. data_summary	**string or null**
data. checked_at	**string (date-time) or null**
data. created_by	**object or null** Identifies who launched the adhoc ID check. The `type` discriminator picks the shape of the remaining fields: `{ type: "user", uuid, username }` - a portal user launched the check. `{ type: "api_key", uuid, label }` - an API key on this account launched the check. `null` - rare legacy rows from before actor attribution was recorded.
data. created_by. type	**string** Discriminator for the actor type. **Enum:** `user`, `api_key`
data. created_by. uuid	**string (uuid)** UUID of the user or API key that launched the check.
data. created_by. username	**string** Username of the portal user. Present only when `type` is `user`.
data. created_by. label	**string** Label of the API key. Present only when `type` is `api_key`.
data. idpass	**object** Present only when `id_type` is `idpass`. A lightweight shell of the linked IDPass plus `idpass_url`, the path to the full IDPass resource.
data. idpass. uuid	**string (uuid)** The IDPass identifier. Use it with the show, image and PDF endpoints. **Example:** `5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`

Field	Description
data. idpass. status	**string** Where the individual is in the hosted flow: `new` - the link has not been opened `opened` - the welcome page was passed `in_progress` - consent was given and documents are being submitted `complete` - the verification finished; read `verification_status` for the outcome `expired` - `link_validity_days` elapsed before completion `failed` - the verification could not be completed `cancelled` - cancelled in the portal The last four are terminal. See the IDPass section of the guide. **Enum:** `new`, `opened`, `in_progress`, `complete`, `expired`, `failed`, `cancelled` **Example:** `in_progress`
data. idpass. verification_status	**string or null** The outcome once `status` is terminal, null until then: `passed` - the identity was verified `review` - the identity was verified but flagged for review in the portal; treat it as not yet passed `failed` - the identity was not verified (including an expired or cancelled IDPass) **Enum:** `passed`, `review`, `failed` **Example:** `passed`
data. idpass. idpass_link	**string or null** The hosted link the individual opens to complete their verification. Treat it as opaque - pass it on as-is and do not parse or rebuild it. **Example:** `https://idpass.globaldata.net.au/idpass?token=abcd123456`
data. idpass. idpass_url	**string** Path to the full IDPass resource (GET /v1/idpasses/{uuid}). **Example:** `/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
data. consent	**object (dynamic)**
data. verification_details	**array of objects**
data. verification_details[]. field	**string**
data. verification_details[]. value	**string**

Field	Description
data. details	one of 6 shapes

Field	Description

OBJECT 1

Field	Description
verification_result_code	string Enum: `Y`, `N`, `D`
originating_agency_code	string or null
verification_request_number	string or null
additional_information	array of strings

OBJECT 2

Field	Description
api_reference	string or null
message	string or null
match_results	object or null
match_results. first_name	string
match_results. last_name	string
match_results. dob	string Date of birth match result from the upstream provider.

OBJECT 3

Field	Description
api_reference	string or null
person_match	string or null

OBJECT 4

Field	Description
api_reference	string or null
reporting_reference	string or null Unique transaction identifier from the data source.
match_status	string or null Overall match result. `Match` indicates the supplied identity was fully verified. Enum: `Match`, `NoMatch`
document_verified	boolean Whether NZTA reports the supplied licence details as verified.
additional_information	string or null Explanatory message returned by the data source, typically alongside a `NoMatch` result.

Field	Description
api_reference	string (uuid)

Sample response


```
{
    "data": {
        "uuid": "00000000-0000-0000-0000-000000000000",
        "check_number": "string",
        "id_type": "string",
        "id_type_name": "string",
        "check_type": "string",
        "check_type_name": "string",
        "outcome": "pass",
        "outcome_summary": "string",
        "data_summary": "string",
        "checked_at": "2024-01-01T00:00:00Z",
        "created_by": {
            "type": "user",
            "uuid": "00000000-0000-0000-0000-000000000000",
            "username": "string",
            "label": "string"
        },
        "idpass": {
            "uuid": "5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a",
            "status": "in_progress",
            "verification_status": "passed",
            "idpass_link": "https://idpass.globaldata.net.au/idpass?token=abcd123456",
            "idpass_url": "/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a"
        },
        "verification_details": [
            {
                "field": "string",
                "value": "string"
            }
        ],
        "details": {
            "verification_result_code": "Y",
            "originating_agency_code": "string",
            "verification_request_number": "string",
            "additional_information": [
                "string"
            ]
        }
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 401, 403, 404, 429, 5XX (see Common error responses).

Delete an adhoc ID check

DELETE `/adhoc-id-checks/{adhoc_id_check_uuid}`

Soft-deletes the adhoc ID check. See the Soft Deletion section of the API guide for the full policy (30-day recovery window, portal-only restore).

The record is no longer returned by list/show endpoints but remains in the database for compliance and audit purposes. The deleted adhoc ID check record is returned in the response as confirmation.

Path parameters

Field	Description
adhoc_id_check_uuid REQUIRED	string (uuid) The adhoc ID check UUID

Responses

200 **Adhoc ID check deleted (soft-delete - returned as confirmation)** (application/json)

Field	Description
data	object A quick (adhoc) identification check not tied to an entity.
data. uuid	string (uuid)
data. check_number	string
data. id_type	string
data. id_type_name	string
data. check_type	string
data. check_type_name	string
data. outcome	string Enum: `pass`, `fail`, `pending`
data. outcome_summary	string or null
data. data_summary	string or null
data. checked_at	string (date-time) or null

Field	Description
data. created_by	**object or null** Identifies who launched the adhoc ID check. The `type` discriminator picks the shape of the remaining fields: `{ type: "user", uuid, username }` - a portal user launched the check. `{ type: "api_key", uuid, label }` - an API key on this account launched the check. `null` - rare legacy rows from before actor attribution was recorded.
data. created_by. type	**string** Discriminator for the actor type. **Enum:** `user`, `api_key`
data. created_by. uuid	**string (uuid)** UUID of the user or API key that launched the check.
data. created_by. username	**string** Username of the portal user. Present only when `type` is `user`.
data. created_by. label	**string** Label of the API key. Present only when `type` is `api_key`.
data. idpass	**object** Present only when `id_type` is `idpass`. A lightweight shell of the linked IDPass plus `idpass_url`, the path to the full IDPass resource.
data. idpass. uuid	**string (uuid)** The IDPass identifier. Use it with the show, image and PDF endpoints. **Example:** `5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
data. idpass. status	**string** Where the individual is in the hosted flow: `new` - the link has not been opened `opened` - the welcome page was passed `in_progress` - consent was given and documents are being submitted `complete` - the verification finished; read `verification_status` for the outcome `expired` - `link_validity_days` elapsed before completion `failed` - the verification could not be completed `cancelled` - cancelled in the portal The last four are terminal. See the IDPass section of the guide. **Enum:** `new`, `opened`, `in_progress`, `complete`, `expired`, `failed`, `cancelled` **Example:** `in_progress`

Field	Description
data. idpass. verification_status	**string or null** The outcome once `status` is terminal, null until then: `passed` - the identity was verified `review` - the identity was verified but flagged for review in the portal; treat it as not yet passed `failed` - the identity was not verified (including an expired or cancelled IDPass) **Enum:** `passed`, `review`, `failed` **Example:** `passed`
data. idpass. idpass_link	**string or null** The hosted link the individual opens to complete their verification. Treat it as opaque - pass it on as-is and do not parse or rebuild it. **Example:** `https://idpass.globaldata.net.au/idpass?token=abcd123456`
data. idpass. idpass_url	**string** Path to the full IDPass resource (GET /v1/idpasses/{uuid}). **Example:** `/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "00000000-0000-0000-0000-000000000000",
        "check_number": "string",
        "id_type": "string",
        "id_type_name": "string",
        "check_type": "string",
        "check_type_name": "string",
        "outcome": "pass",
        "outcome_summary": "string",
        "data_summary": "string",
        "checked_at": "2024-01-01T00:00:00Z",
        "created_by": {
            "type": "user",
            "uuid": "00000000-0000-0000-0000-000000000000",
            "username": "string",
            "label": "string"
        },
        "idpass": {
            "uuid": "5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a",
            "status": "in_progress",
            "verification_status": "passed",
            "idpass_link": "https://idpass.globaldata.net.au/idpass?token=abcd123456",
            "idpass_url": "/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a"
        }
    },
```

```
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```

Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

Download an adhoc ID check certificate as PDF

GET `/adhoc-id-checks/{adhoc_id_check_uuid}/pdf`

Returns a verification certificate PDF for a completed adhoc ID check.

Status codes

- `200` - the PDF certificate is returned in the response body.
- `404` - either the adhoc ID check does not exist on the calling account, or the `id_type` is `idpass` (which does not produce a downloadable certificate).
- `409` - the adhoc ID check has not yet reached a terminal outcome. Poll show adhoc ID check until `outcome` is `pass` or `fail` before retrying.

Path parameters

Field	Description
`adhoc_id_check_uuid` REQUIRED	**string (uuid)** The uuid of the adhoc ID check.

Responses

200 **PDF certificate** (application/pdf)

Binary PDF file

409 **The adhoc ID check has not yet completed. Poll the show endpoint until `outcome` is `pass` or `fail` before retrying the download.** (application/json)

Field	Description
`message`	**string** **Example:** `The adhoc ID check has not completed yet. Wait for outcome to be 'pass' or 'fail' before downloading the certificate.`

Sample response

```
{
    "message": "The adhoc ID check has not completed yet. Wait for outcome to be 'pass' or 'fail' before downloa
}
```

Standard error responses: 401, 403, 404, 429, 5XX (see Common error responses).

Audits

The audit log of actions on the calling account, including API calls.

List audit entries

GET `/audits`

Returns audit log entries for the calling account. Results are paginated.

The audit log records significant activity on your account: who did what, when, from which IP, and against which resource. Use it to feed external SIEM / compliance systems, to reconstruct an entity's history, or to surface activity in your own UI.

Filters

The following filters can be applied via the `filter[...]` query parameters:

- `action` - exact action key (e.g. `created_watcheye_entity`)
- `action_type` - optional action sub-type (e.g. `password` for `login`)
- `user_uuid` - only actions performed by the given user
- `api_key_uuid` - only actions performed by the given API key
- `entity_uuid` - only actions associated with the given entity (covers direct entity actions plus actions on related resources like checks, events, and notes)
- `created_at_from` - inclusive lower bound on the audit timestamp (ISO 8601)
- `created_at_to` - inclusive upper bound on the audit timestamp (ISO 8601)

UUID-targeted filters return an empty page when the target uuid does not exist on the calling account.

Sorting

The `sort` query parameter accepts `created_at` (default: `-created_at`, newest first). Prefix with `-` for descending order.

Properties

The list endpoint does not return the per-audit `properties` object. Properties for some actions can be many kilobytes, which would balloon page payloads. Use the show endpoint to retrieve the full properties for a single audit entry.

Query parameters

Field	Description
`page`	**integer** The page of results to return, starting at 1. **Example:** `1`

Field	Description
`per_page`	**integer** The number of audit entries per page (defaults to 30, max 500) **Example:** `30`
`filter[action]`	**string** Exact action key **Example:** `created_watcheye_entity`
`filter[action_type]`	**string** Only audit entries of the given action type. **Example:** `password`
`filter[user_uuid]`	**string (uuid)** Only audit entries performed by the given portal user.
`filter[api_key_uuid]`	**string (uuid)** Only audit entries performed by the given API key.
`filter[entity_uuid]`	**string (uuid)** Only records for the given entity.
`filter[created_at_from]`	**string (date-time)** Inclusive lower bound on `created_at` (ISO 8601). **Example:** `2025-01-01T00:00:00Z`
`filter[created_at_to]`	**string (date-time)** Inclusive upper bound on `created_at` (ISO 8601). **Example:** `2025-12-31T23:59:59Z`
`sort`	**string** Field to sort by; prefix with `-` for descending order **Enum:** `created_at`, `-created_at` **Example:** `-created_at`

Responses

200 **Audit list response** (application/json)

Field	Description
`data`	**array of objects** An array of audit entries

Field	Description
data[]. uuid	**string (uuid)** The audit entry's UUID **Example:** `5e1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data[]. action	**string** Machine-readable action key. Use this for programmatic logic (filtering, branching). Example values include `created_watcheye_entity`, `updated_watcheye_entity`, `login`, `created_api_key`. **Example:** `created_watcheye_entity`
data[]. action_label	**string** Human-readable label for the action, suitable for display in UIs. Derived from the machine action key and is stable per action. **Example:** `Entity created`
data[]. action_type	**string or null** Optional sub-type qualifying the action. Used for actions that have meaningful variants (e.g. `login` -> `password` / `saml`, `login_failed` -> `password_incorrect` / `user_does_not_exist`). `null` for actions without sub-types. **Example:** `password`
data[]. description	**string** Human-readable description recorded at the time the audit was created. For most actions this is the action label; for some it includes additional context (e.g. the name of the resource touched). **Example:** `Entity created`
data[]. client_ip_address	**string or null** The IP address the action originated from, where one was captured. `null` for actions performed by the system. **Example:** `203.0.113.42`
data[]. agent	**object** Who performed the action. Always present; `name` and `uuid` are populated where the agent identity is available, and are `null` for system-initiated actions or where the agent record is no longer available.

Field	Description
data[]. agent. type	**string** `user` - a user performed the action `api_key` - an API key performed the action `system` - the system performed the action (no agent identity) **Enum:** `user`, `api_key`, `system` **Example:** `user`
data[]. agent. uuid	**string (uuid) or null** UUID of the user or API key, or `null` when not available. **Example:** `11111111-2222-3333-4444-555555555555`
data[]. agent. name	**string or null** Display name of the agent: a user's full name (or username if no name is set), or the API key's label. `null` when the agent is the system or the agent record is no longer available. **Example:** `Jane Doe`
data[]. subject	**object or null** The resource the action was performed on. `null` when the audit has no specific subject (e.g. `login`). `uuid` is `null` when the subject record is no longer available.
data[]. subject. type	**string** Identifies the resource type for this audit's subject. Use it together with `uuid` to deep-link back to the resource via the matching API endpoint. Common values include `watcheyeEntity`, `watcheyeProgram`, `watcheyeNote`, `watcheyeEvent`, `watcheyeCheck`, `user`, and `api_key`. **Example:** `watcheyeEntity`
data[]. subject. uuid	**string (uuid) or null** UUID of the subject resource, or `null` when not available. **Example:** `123e4567-e89b-12d3-a456-426614174000`
data[]. created_at	**string (date-time)** ISO 8601 timestamp at which the audited action occurred **Example:** `2025-01-01T00:00:00Z`
meta	**object**
meta. current_page	**integer** **Example:** `1`
meta. per_page	**integer** **Example:** `30`

Field	Description
meta. total	integer Example: `1`
meta. last_page	integer Example: `1`
api_reference	string (uuid)

Sample response


```
{
    "data": [
        {
            "uuid": "5e1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
            "action": "created_watcheye_entity",
            "action_label": "Entity created",
            "action_type": "password",
            "description": "Entity created",
            "client_ip_address": "203.0.113.42",
            "agent": {
                "type": "user",
                "uuid": "11111111-2222-3333-4444-555555555555",
                "name": "Jane Doe"
            },
            "subject": {
                "type": "watcheyeEntity",
                "uuid": "123e4567-e89b-12d3-a456-426614174000"
            },
            "created_at": "2025-01-01T00:00:00Z"
        }
    ],
    "meta": {
        "current_page": 1,
        "per_page": 30,
        "total": 1,
        "last_page": 1
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 403, 429, 5XX (see Common error responses).

Show an audit entry

GET `/audits/{audit_uuid}`

Returns a single audit entry by its UUID, including the full `properties` object for the action.

The shape of the `properties` object varies by action. Treat it as an action-specific bag of contextual data rather than a fixed schema; new keys may be added over time as new features are added.

Path parameters

Field	Description
`audit_uuid` REQUIRED	**string (uuid)** The audit entry UUID

Responses

200 **Audit entry response** (application/json)

Field	Description
`data`	**object** Same as Audit, plus the `properties` object. Returned by the show endpoint only; the list endpoint omits properties because they can be many kilobytes for some actions.
`data.` `uuid`	**string (uuid)** The audit entry's UUID **Example:** `5e1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
`data.` `action`	**string** Machine-readable action key (see Audit.action) **Example:** `updated_watcheye_entity`
`data.` `action_label`	**string** Human-readable label for the action **Example:** `Entity updated`
`data.` `action_type`	**string or null** Optional sub-type qualifying the action
`data.` `description`	**string** Human-readable description recorded at the time the audit was created
`data.` `client_ip_address`	**string or null** Originating IP address (`null` for actions performed by the system)
`data.` `agent`	**object** Who performed the action - see Audit.agent

`data.`
`agent.`
`type`
	string
	Enum: `user` , `api_key` , `system`

`data.`
`agent.`
`uuid`
	string (uuid) or null

Field	Description
data. agent. name	**string or null**
data. subject	**object or null** The resource the action was performed on - see Audit.subject
data. subject. type	**string**
data. subject. uuid	**string (uuid) or null**
data. properties	**object (dynamic) or null** Action-specific properties recorded at the time of the action. The exact keys vary by action - for example `created_watcheye_entity` includes the entity's attributes at creation time; `login_failed` includes the attempted username; `created_user` includes the new user's username and roles. Treat this as an action-specific bag of contextual data rather than a fixed schema; new keys may be added over time as new features are added. `null` when no properties are available for the action.
data. created_at	**string (date-time)** ISO 8601 timestamp at which the audited action occurred **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "5e1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "action": "updated_watcheye_entity",
        "action_label": "Entity updated",
        "action_type": "string",
        "description": "string",
        "client_ip_address": "string",
        "agent": {
            "type": "user",
            "uuid": "00000000-0000-0000-0000-000000000000",
            "name": "string"
        },
        "subject": {
            "type": "string",
            "uuid": "00000000-0000-0000-0000-000000000000"
        },
        "properties": {
            "attributes": {
                "entity_name": "Jane A Doe",
```

```
                "entity_type": "individual",
                "risk_level": "low"
            }
        },
        "created_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```

Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

Checks

Screening checks run against entities, and their results.

List checks

GET `/checks`

Returns checks for the calling account. Results are paginated.

Each check represents a single screening run against an entity - for example a PEP and Sanction screen, an email verification, a court records search. The list endpoint returns a compact at-a-glance envelope per check; use the show check endpoint to read the typed `details` block and the uniform `results_overview` array for a specific record.

Group checks

A `check_type` value of `group` represents a parent record that aggregates one or more child checks (for example a monitor run that triggers a PEP screen, an adverse media search and a phone check on the same entity). Group checks appear in the list as first-class records; use the `filter[parent_uuid]` filter to find their children, or fetch the group via show check to read the `children[]` summary.

Filters

The following filters can be applied via the `filter[...]` query parameters:

- `check_type` - one of the operation types (e.g. `pep_sanction_check`, `phone_check`, `business_check`) or the special value `group` for aggregate parent records
- `outcome` - `pass` or `warning`
- `program_uuid` - only checks for entities in the given program
- `entity_uuid` - only checks for the given entity
- `parent_uuid` - only checks whose parent is the given check (use this with a group's uuid to list its children)
- `checked_after` - inclusive lower bound on `checked_at` (ISO 8601)
- `checked_before` - inclusive upper bound on `checked_at` (ISO 8601)

The top-level `archived` query parameter accepts `true` (only archived) or `false` (only non-archived); omitting it returns both.

Filters that target a uuid (program, entity, parent) return an empty page when the target uuid does not exist on the calling account, identical to the behaviour for a uuid that does not exist at all.

Sorting

The `sort` query parameter accepts `checked_at` (default: `-checked_at`, newest first), `check_type` or `outcome`. Prefix with `-` for descending order.

Query parameters

Field	Description
`page`	**integer** The page of results to return, starting at 1. **Example:** `1`
`per_page`	**integer** The number of checks per page (defaults to 30, max 500) **Example:** `30`
`filter[check_type]`	**string** Only records of the given check type. The endpoint description lists the values. **Example:** `pep_sanction_check`
`filter[outcome]`	**string** Only records with the given outcome. The endpoint description lists the values. **Enum:** `pass`, `warning`
`filter[program_uuid]`	**string (uuid)** Only records for entities in the given program.
`filter[entity_uuid]`	**string (uuid)** Only records for the given entity.
`filter[parent_uuid]`	**string (uuid)** Only records whose parent is the given record. Use a group's uuid to list its children.
`filter[checked_after]`	**string (date-time)** Inclusive lower bound on `checked_at` (ISO 8601). **Example:** `2025-01-01T00:00:00Z`
`filter[checked_before]`	**string (date-time)** Inclusive upper bound on `checked_at` (ISO 8601). **Example:** `2025-12-31T23:59:59Z`

Field	Description
`sort`	**string** Field to sort by; prefix with `-` for descending order **Enum:** `checked_at`, `-checked_at`, `check_type`, `-check_type`, `outcome`, `-outcome` **Example:** `-checked_at`
`archived`	**boolean** Filter by archive status. `true` - only archived checks `false` - only non-archived checks Note: The API will accept the string 'true' or the number 1 for true and the string 'false' or the number 0 for false as well as the boolean values.

Responses

200 **Checks list response** (application/json)

Field	Description
`data`	**array of objects** An array of checks
`data[].` `uuid`	**string (uuid)** The check's UUID **Example:** `4d7e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
`data[].` `check_number`	**string** A short obfuscated public identifier for the check (e.g. "C-A1B2CD") **Example:** `C-A1B2CD`
`data[].` `check_type`	**string** The type of check that was run. One of the supported screening operations (e.g. `pep_sanction_check`, `phone_check`, `business_check`) or `group` for an aggregate record that owns one or more child checks. **Example:** `pep_sanction_check`
`data[].` `check_type_name`	**string** Display name for the check type (e.g. "PEP and Sanction Screening", "Phone Check", "Group Check"). Suitable for showing in UIs. **Example:** `PEP and Sanction Screening`

Field	Description
data[]. status	**string** Lifecycle status of the check: `pending` - the check has been created and charged but the upstream call has not started yet `processing` - the upstream provider is being called `complete` - the check finished successfully and `outcome` is populated `failed` - the check could not be completed; `failed_reason` is populated and the credit charged at launch has been refunded For group checks the status is rolled up from the children: any child still pending or processing keeps the group at `processing`; the group is `failed` when any child failed, otherwise `complete` when every child is complete. **Enum:** `pending`, `processing`, `complete`, `failed` **Example:** `complete`
data[]. failed_reason	**string or null** Short reason why the check failed. Populated only when `status = failed`, otherwise `null`.
data[]. outcome	**string or null** High-level outcome of the check: `pass` - nothing flagged `warning` - one or more matches found that you should review `null` for group checks that have no child results yet, and for checks that have not reached `status = complete`. **Enum:** `pass`, `warning` **Example:** `warning`
data[]. check_summary	**string or null** Short, human-readable summary of the result (e.g. "Found 2 PEP matches", "All phones connected", "Business record exists"). `null` if the operation does not produce a summary. **Example:** `Found 2 PEP matches`
data[]. data_summary	**string or null** Short, human-readable summary of the subject the check ran against (e.g. "John A Smith, 1980-06-23" or "ACME Pty Ltd, ABN: 12 345 678 901"). `null` for group checks and for any check whose operation does not produce a subject summary. **Example:** `John A Smith, 1980-06-23`

Field	Description
data[]. checked_at	**string (date-time) or null** ISO 8601 timestamp at which the check was run. `null` while the check is still `pending` or `processing`. **Example:** `2025-01-01T00:00:00Z`
data[]. archived_at	**string (date-time) or null** ISO 8601 timestamp at which the check was archived (hidden from the active list in the portal). `null` when the check is not archived.
data[]. program	**object or null** The program the check belongs to. `null` for adhoc checks that do not belong to a program.
data[]. program. uuid	**string (uuid)** UUID of the program **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data[]. program. program_name	**string** Display name of the program **Example:** `PEP & Sanction Daily`
data[]. entity	**object or null** The entity the check ran against. `null` for adhoc checks that do not target an entity record.
data[]. entity. uuid	**string (uuid)** UUID of the entity **Example:** `123e4567-e89b-12d3-a456-426614174000`
data[]. entity. entity_number	**string** A short obfuscated public identifier for the entity **Example:** `E-A1B2CD`
data[]. entity. entity_name	**string** Display name of the entity **Example:** `John A Smith`
data[]. parent_uuid	**string (uuid) or null** UUID of the parent (group) check this check is a child of. `null` when the check is not part of a group.

Field	Description
data[]. events_count	**integer** How many events have been raised against this check. `0` when no matches required an event. **Example:** `1`
meta	**object**
meta. current_page	**integer** **Example:** `1`
meta. per_page	**integer** **Example:** `30`
meta. total	**integer** **Example:** `1`
meta. last_page	**integer** **Example:** `1`
api_reference	**string (uuid)**

Sample response


```
{
    "data": [
        {
            "uuid": "4d7e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
            "check_number": "C-A1B2CD",
            "check_type": "pep_sanction_check",
            "check_type_name": "PEP and Sanction Screening",
            "status": "complete",
            "failed_reason": null,
            "outcome": "warning",
            "check_summary": "Found 2 PEP matches",
            "data_summary": "John A Smith, 1980-06-23",
            "checked_at": "2025-01-01T00:00:00Z",
            "archived_at": null,
            "program": {
                "uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
                "program_name": "PEP & Sanction Daily"
            },
            "entity": {
                "uuid": "123e4567-e89b-12d3-a456-426614174000",
                "entity_number": "E-A1B2CD",
                "entity_name": "John A Smith"
            },
            "parent_uuid": null,
            "events_count": 1
        }
    ],
    "meta": {
```

```
        "current_page": 1,
        "per_page": 30,
        "total": 1,
        "last_page": 1
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```

Standard error responses: 400, 401, 403, 429, 5XX (see Common error responses).

Show a check

GET `/checks/{check_uuid}`

Returns a single check by its UUID, including:

The base envelope (program, entity, outcome, timestamps, etc.) - same shape as the list endpoint

`results_overview[]` - one row per result, with a uniform `{title, status, alert_ids}` shape across all operation types

`children[]` - per-child summaries when the check is a group; empty otherwise

`details` - the typed per-operation details block. The exact shape depends on `check_type` and is described by the discriminated union under CheckDetail.details. Group checks (`check_type = group`) always return `details: null` - fetch each child via this endpoint to read its typed details.

PDF version

A printable PDF of the check (including subject, configuration and per-match details) is available via the PDF download endpoint.

Path parameters

Field	Description
`check_uuid` REQUIRED	**string (uuid)** The check UUID

Responses

200 **Check response** (application/json)

Field	Description
`data`	**object** Same as Check, plus the uniform `results_overview` and `children` arrays and the typed per-operation `details` block. Returned by the show endpoint so callers do not need a follow-up request to read match-level data.

Field	Description
data. uuid	**string (uuid)** The check's UUID **Example:** `4d7e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. check_number	**string** A short obfuscated public identifier for the check **Example:** `C-A1B2CD`
data. check_type	**string** The type of check that was run (see Check.check_type) **Example:** `pep_sanction_check`
data. check_type_name	**string** Display name for the check type **Example:** `PEP and Sanction Screening`
data. status	**string** Lifecycle status of the check (see Check.status) **Enum:** `pending`, `processing`, `complete`, `failed` **Example:** `complete`
data. failed_reason	**string or null** Short reason why the check failed (see Check.failed_reason)
data. outcome	**string or null** High-level outcome of the check (see Check.outcome) **Enum:** `pass`, `warning` **Example:** `warning`
data. check_summary	**string or null** Short, human-readable summary of the result **Example:** `Found 2 PEP matches`
data. data_summary	**string or null** Short, human-readable summary of the subject the check ran against **Example:** `John A Smith, 1980-06-23`
data. checked_at	**string (date-time) or null** **Example:** `2025-01-01T00:00:00Z`
data. archived_at	**string (date-time) or null** ISO 8601 timestamp at which the check was archived (null if not archived)

Field	Description
data. program	**object or null**
data. program. uuid	**string (uuid)**
data. program. program_name	**string**
data. entity	**object or null**
data. entity. uuid	**string (uuid)**
data. entity. entity_number	**string**
data. entity. entity_name	**string**
data. parent_uuid	**string (uuid) or null**
data. events_count	**integer**
data. results_overview	**array of objects** A uniform per-result summary across all operation types. Empty for group checks and for operations that pack everything into a single record (in which case `details.record` carries the full data).
data. results_overview[]. title	**string** Short label for the result, suitable for display in a list (e.g. "Match 1: John Smith - PEP", "0412 345 678 (Connected)", "5 records at 12 Main St, Sydney"). **Example:** `Match 1: John Smith - PEP`
data. results_overview[]. status	**string** Per-result status: `pass` - this individual result is clean `warning` - this individual result needs review (the default for any non-pass row when an operation does not assign a finer-grained status) `fail` - this individual result is a definitive failure (used by email and phone checks for undeliverable / disconnected entries) **Enum:** `pass`, `warning`, `fail` **Example:** `warning`

Field	Description
data. results_overview[]. alert_ids	**array of strings** Provider-side identifiers for the alert(s) this row generated. Use these to match up against `check_result_ids` on a related event from the events endpoint. Empty when the result is `pass`.
data. children	**array of objects** Summaries of the child checks for a group check. Empty for non-group checks. Fetch each child via the show check endpoint to get its full payload.
data. children[]. uuid	**string (uuid)** **Example:** `4d7e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. children[]. check_number	**string** **Example:** `C-A1B2CD`
data. children[]. check_type	**string** **Example:** `pep_sanction_check`
data. children[]. check_type_name	**string** **Example:** `PEP and Sanction Screening`
data. children[]. status	**string** Lifecycle status of the child check (see Check.status) **Enum:** `pending`, `processing`, `complete`, `failed` **Example:** `complete`
data. children[]. failed_reason	**string or null** Short reason why the child check failed (see Check.failed_reason)
data. children[]. outcome	**string or null** **Enum:** `pass`, `warning` **Example:** `warning`
data. children[]. check_summary	**string or null** Short human-readable summary of the child check result (see Check.check_summary). `null` while the child is pending / processing or when the child has no summary. **Example:** `Found 2 PEP matches`
data. children[]. checked_at	**string (date-time) or null** **Example:** `2025-01-01T00:00:00Z`

Field	Description
data. details	**one of 11 shapes** Typed per-operation details for the check. `null` for group checks. The exact shape depends on the `check_type`: `pep_sanction_check` - see CheckDetailsPepSanction `adverse_media` - see CheckDetailsAdverseMedia `adc_check` - see CheckDetailsAdc `adc_custodian_check` - see CheckDetailsAdcCustodian `banned_disqualified_persons` - see CheckDetailsBannedDisqualified `court_check` - see CheckDetailsCourt `email_check` - see CheckDetailsEmail `phone_check` - see CheckDetailsPhone `business_check` - see CheckDetailsBusiness `uk_business_check` - see CheckDetailsUkBusiness `realestate_check` - see CheckDetailsRealEstate

Field	Description

OBJECT 1

Field	Description
subject	**object**
subject. full_name	**string or null** **Example:** `John Smith`
subject. first_name	**string or null** **Example:** `John`
subject. middle_name	**string or null** **Example:** `Albert`
subject. last_name	**string or null** **Example:** `Smith`
subject. dob	**string or null** **Example:** `1980-06-23`
subject. business_name	**string or null**
subject. vessel_name	**string or null**
subject. address_country	**string or null** **Example:** `AU`
config	**object**
config. search_type	**string or null** **Enum:** `broad_search` , `medium_search` , `narrow_search` **Example:** `medium_search`
config. similarity_threshold	**integer or null** **Example:** `80`
config. pep_countries	**array of strings**
config. sanction_countries	**array of strings**
config. max_results	**integer or null** **Example:** `25`
config. extended_result	**boolean** **Example:** `true`
results	**array of objects**
results[]. uuid	**string or null** Provider-side identifier for the match. **Example:** `NK-12345`
results[]. name	**string or null** Canonical name on the matched record.

Field	Description
api_reference	string (uuid)

Sample response


```
{
    "data": {
        "uuid": "4d7e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "check_number": "C-A1B2CD",
        "check_type": "pep_sanction_check",
        "check_type_name": "PEP and Sanction Screening",
        "status": "complete",
        "failed_reason": null,
        "outcome": "warning",
        "check_summary": "Found 2 PEP matches",
        "data_summary": "John A Smith, 1980-06-23",
        "checked_at": "2025-01-01T00:00:00Z",
        "archived_at": null,
        "program": {
            "uuid": "00000000-0000-0000-0000-000000000000",
            "program_name": "string"
        },
        "entity": {
            "uuid": "00000000-0000-0000-0000-000000000000",
            "entity_number": "string",
            "entity_name": "string"
        },
        "parent_uuid": "00000000-0000-0000-0000-000000000000",
        "events_count": 0,
        "results_overview": [
            {
                "title": "Match 1: John Smith - PEP",
                "status": "warning",
                "alert_ids": [
                    "NK-12345"
                ]
            }
        ],
        "children": [
            {
                "uuid": "4d7e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
                "check_number": "C-A1B2CD",
                "check_type": "pep_sanction_check",
                "check_type_name": "PEP and Sanction Screening",
                "status": "complete",
                "failed_reason": null,
                "outcome": "warning",
                "check_summary": "Found 2 PEP matches",
                "checked_at": "2025-01-01T00:00:00Z"
            }
        ],
        "details": {
            "subject": {
                "full_name": "John Smith",
                "first_name": "John",
                "middle_name": "Albert",
```

```
            "last_name": "Smith",
            "dob": "1980-06-23",
            "business_name": null,
            "vessel_name": null,
            "address_country": "AU"
        },
        "config": {
            "search_type": "medium_search",
            "similarity_threshold": 80,
            "pep_countries": [],
            "sanction_countries": [
                "au",
                "nz",
                "gb",
                "us",
                "ca"
            ],
            "max_results": 25,
            "extended_result": true
        },
        "results": [
            {
                "uuid": "NK-12345",
                "name": "John Albert Smith",
                "matched_name": "J A Smith",
                "similarity": 0.95,
                "is_pep": true,
                "is_sanctioned": false,
                "pep_matches": [],
                "sanction_matches": [],
                "record": {
                    "id": "Q12345",
                    "caption": "John Albert Smith",
                    "schema": "Person",
                    "first_seen": "2018-01-01T00:00:00Z",
                    "last_seen": "2025-01-01T00:00:00Z",
                    "last_change": "2024-12-01T00:00:00Z"
                },
                "properties": {
                    "birth_date": [
                        "1980-06-23"
                    ],
                    "death_date": [],
                    "gender": [
                        "male"
                    ],
                    "nationality": [
                        "au"
                    ],
                    "citizenship": [
                        "au"
                    ],
                    "birth_place": [
                        "Sydney"
                    ],
                    "aliases": [
                        "J Smith",
                        "Johnny Smith"
                    ],
```

```
                        "positions": [
                            "Senator",
                            "Cabinet Minister"
                        ],
                        "phone": [],
                        "email": [],
                        "address": [
                            "12 Main St, Sydney NSW 2000"
                        ],
                        "education": [],
                        "religion": [],
                        "notes": [],
                        "wikidata_id": [
                            "Q12345"
                        ],
                        "source_urls": [
                            "https://example.com/source"
                        ],
                        "websites": []
                    }
                }
            ]
        }
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

Download a check as PDF

GET `/checks/{check_uuid}/pdf`

Returns a printable PDF report for the check, suitable for inclusion in customer-facing files. The PDF mirrors the on-screen check report in the portal: subject snapshot, configuration in force, per-match details and a results summary.

The response is `application/pdf` with a `Content-Disposition` header naming the file after the check's `check_number` (e.g. `check-C-A1B2CD.pdf`).

This endpoint does not bill - PDF generation reuses the existing on-record check data.

> **Note:** PDF rendering is performed on demand and typically takes between 10 and 60 seconds depending on the size of the check (number of matches, embedded fonts, etc.). Increase your HTTP client's read / response timeout accordingly - the default 30 second timeouts in most libraries (curl, Guzzle, axios, requests) are not enough for the larger reports.

Path parameters

Field	Description
`check_uuid` REQUIRED	string (uuid) The check UUID

Responses

200 **PDF report** (application/pdf)

Binary PDF file

Standard error responses: 401, 403, 404, 429, 5XX (see Common error responses).

Launch screening checks against an entity

POST `/entities/{entity_uuid}/checks`

Queues one or more screening checks against a saved entity and returns `202 Accepted` with the parent (or single child) check UUID. The screening runs in the background - poll `GET /v1/checks/{uuid}` and wait for `status` to be `complete` or `failed` before reading the results.

Each requested check type is charged at launch time. If the upstream provider returns an error, the check is marked as `failed` and the credit for that individual check is refunded.

When a single check type is requested, the response describes that check. When two or more types are requested, a parent group check is created and returned, with one child per requested type.

Pre-flight checks

The endpoint validates the request before any check rows are created or any credit is charged:

- `404` - the entity was not found on the calling account
- `400` - the request body failed validation (missing or unknown check type)
- `403 Forbidden` - the calling account is not entitled to one or more of the requested check types (the product is not enabled for the account)
- `402 Payment Required` - the calling account has insufficient credit to cover the total projected cost across all requested check types

Idempotency

This endpoint accepts the `Idempotency-Key` header so that retries on network failure are safe. See the Idempotency section of the API guide for the full contract.

Path parameters

Field	Description
`entity_uuid` REQUIRED	string (uuid) The entity UUID

Header parameters

Field	Description
`Idempotency-Key`	**string [max 255 characters]** Optional client-generated key (typically a UUID) that lets you safely retry a write request without risk of duplicate work or duplicate billing. A retry with the same key and body returns the original response verbatim, with the `Idempotent-Replay: true` response header. Honoured on `POST` requests only. See the Idempotency section of the API guide for the full contract. **Example:** `8c4d3a18-2f63-4f9a-9f3a-9b1f5a7c2d4f`

Request body

The launch parameters

Field	Description
`check_types` REQUIRED	**array of strings [min 1 items]** One or more operation names to launch. See the operation enum in the Check schema for the supported set.
`config`	**object** Per-operation configuration map keyed by operation name. Each listed operation has defaults that apply when its key is omitted, so you only need to send a block when you want to override a default. The five operations not listed below (`adverse_media` , `phone_check` , `email_check` , `adc_check` , `adc_custodian_check`) take no configuration.
`config.` `pep_sanction_check`	**object** Per-operation configuration for a `pep_sanction_check` launch. Every property is optional - any key you omit is filled with its default value before the check runs, and the resolved values are echoed back on the launch response so you can confirm what the check is using.

`config.`
`pep_sanction_check.`
`search_type`
	string
	Match strictness. `narrow_search` requires close name matches and returns the fewest false positives; `broad_search` returns more candidates for review; `medium_search` is the balanced default. Defaults to `medium_search` .
	Enum: `broad_search` , `medium_search` , `narrow_search`
	Example: `medium_search`

Field	Description
config. pep_sanction_check. similarity_threshold	**string** Minimum name similarity percentage (10..100). Match candidates below this score are dropped. Sent as a string. Defaults to `"80"`. **Enum:** `10`, `20`, `30`, `40`, `50`, `60`, `70`, `80`, `90`, `100` **Example:** `80`
config. pep_sanction_check. pep_countries	**array of strings** Two-letter ISO 3166-1 alpha-2 country codes (either case is accepted; returned in uppercase) to scope the PEP search. An empty array searches all PEP jurisdictions (the default).
config. pep_sanction_check. sanction_countries	**array of strings** Two-letter ISO 3166-1 alpha-2 country codes (either case is accepted; returned in uppercase) to scope the sanction search. Defaults to `["AU", "CA", "NZ", "GB", "US"]`.
config. pep_sanction_check. max_results	**integer** Maximum number of match candidates to return. Sent as an integer. Defaults to `25`. **Enum:** `10`, `15`, `20`, `25`, `50`, `100` **Example:** `25`
config. pep_sanction_check. pep_sanction_extended_result	**boolean** When `true`, includes the extended biographical and source-list detail on each match. Defaults to `true`. **Example:** `true`
config. banned_disqualified_persons	**object** Per-operation configuration for a `banned_disqualified_persons` launch. Every property is optional - any key you omit is filled with its default value before the check runs.
config. banned_disqualified_persons. search_type	**string** Match strictness. `narrow_search` requires close name matches and returns the fewest false positives; `broad_search` returns more candidates for review; `medium_search` is the balanced default. Defaults to `medium_search`. **Enum:** `broad_search`, `medium_search`, `narrow_search` **Example:** `medium_search`
config. banned_disqualified_persons. similarity_threshold	**string** Minimum name similarity percentage (10..100), sent as a string. Defaults to `"80"`. **Enum:** `10`, `20`, `30`, `40`, `50`, `60`, `70`, `80`, `90`, `100` **Example:** `80`

Field	Description
config. banned_disqualified_persons. banned_types	**array of strings** Restrict the search to one or more specific banned/disqualified registers. An empty array (the default) searches every register. **Enum:** `afs_banned_disqualified`, `banned_futures`, `banned_securities`, `credit_banned_disqualified`, `disqualified_director`, `disqualified_smsf`, `ato_disqualified_trustee`
config. court_check	**object** Per-operation configuration for a `court_check` launch. Every property is optional - any key you omit is filled with its default value before the check runs.
config. court_check. listing	**string** Which court listings to search. `criminal` is the default; `civil` searches civil matters only; `all` searches both. **Enum:** `all`, `civil`, `criminal` **Example:** `criminal`
config. court_check. search_party	**string** Whether the subject should be matched as the `defendant`, `plaintiff`, or `any` party to the case. Defaults to `defendant`. **Enum:** `any`, `plaintiff`, `defendant` **Example:** `defendant`
config. business_check	**object** Per-operation configuration for a `business_check` launch. Every property is optional - any key you omit is filled with its default value before the check runs. Each property selects whether the corresponding business-record signal raises an alert (`yes`), is recorded for audit but does not raise an alert (`on_change`), or is skipped entirely (`no`); the recent-window properties select a months-back window instead.
config. business_check. name_match	**string** How strictly the trading name on the entity must match the registered ASIC name. Defaults to `similar`. **Enum:** `exact`, `similar`, `no` **Example:** `similar`
config. business_check. abn_active	**string** Alert when the ABN is not active. Defaults to `yes`. **Enum:** `yes`, `on_change`, `no` **Example:** `yes`

Field	Description
config. business_check. acn_active	**string** Alert when the ACN is not active. Defaults to `yes`. **Enum:** `yes`, `on_change`, `no` **Example:** `yes`
config. business_check. gst_registered	**string** Alert when GST registration changes. Defaults to `yes`. **Enum:** `yes`, `on_change`, `no` **Example:** `yes`
config. business_check. recent_documents	**string** Alert when ASIC documents have been lodged within the last N months. `no` disables the check; `1` .. `12` set the months-back window. Defaults to `"3"`. **Enum:** `no`, `1`, `2`, `3`, `6`, `12` **Example:** `3`
config. business_check. recent_business_names	**string** Alert when registered business names have changed within the last N months. `no` disables the check; `1` .. `12` set the months-back window. Defaults to `"3"`. **Enum:** `no`, `1`, `2`, `3`, `6`, `12` **Example:** `3`
config. uk_business_check	**object** Per-operation configuration for a `uk_business_check` launch. Every property is optional - any key you omit is filled with its default value before the check runs.
config. uk_business_check. name_match	**string** How strictly the trading name must match the Companies House record. Defaults to `similar`. **Enum:** `exact`, `similar`, `no` **Example:** `similar`
config. uk_business_check. company_status	**string** Alert when the company status is not `active`. Defaults to `yes`. **Enum:** `yes`, `on_change`, `no` **Example:** `yes`
config. uk_business_check. recent_filings	**string** Alert when filings have been lodged within the last N months. Defaults to `"3"`. **Enum:** `no`, `1`, `2`, `3`, `6`, `12` **Example:** `3`

Field	Description
config. uk_business_check. officer_changes	**string** Alert when officers have changed within the last N months. Defaults to `"3"`. **Enum:** `no`, `1`, `2`, `3`, `6`, `12` **Example:** `3`
config. uk_business_check. registered_address_change	**string** Whether to record registered address changes. Defaults to `on_change`. **Enum:** `on_change`, `no` **Example:** `on_change`
config. uk_business_check. sic_codes_change	**string** Whether to record SIC code changes. Defaults to `on_change`. **Enum:** `on_change`, `no` **Example:** `on_change`
config. realestate_check	**object** Per-operation configuration for a `realestate_check` launch. Every property is optional.
config. realestate_check. date_from	**string (date) or null** ISO 8601 date (`YYYY-MM-DD`). When set, only listings on or after this date are considered. Omit to search all available history. **Example:** `2024-01-01`

Sample request

```
{
    "check_types": [
        "pep_sanction_check"
    ],
    "config": {
        "pep_sanction_check": {
            "search_type": "medium_search",
            "similarity_threshold": "80",
            "pep_countries": [],
            "sanction_countries": [
                "AU",
                "CA",
                "NZ",
                "GB",
                "US"
            ],
            "max_results": 25,
            "pep_sanction_extended_result": true
        },
        "banned_disqualified_persons": {
            "search_type": "medium_search",
            "similarity_threshold": "80",
```

```
            "banned_types": []
        },
        "court_check": {
            "listing": "criminal",
            "search_party": "defendant"
        },
        "business_check": {
            "name_match": "similar",
            "abn_active": "yes",
            "acn_active": "yes",
            "gst_registered": "yes",
            "recent_documents": "3",
            "recent_business_names": "3"
        },
        "uk_business_check": {
            "name_match": "similar",
            "company_status": "yes",
            "recent_filings": "3",
            "officer_changes": "3",
            "registered_address_change": "on_change",
            "sic_codes_change": "on_change"
        },
        "realestate_check": {
            "date_from": "2024-01-01"
        }
    }
}
```

Responses

202 **Check accepted - the parent (or single child) check has been queued for processing. Poll `GET /v1/checks/{uuid}` until `status` is `complete` or `failed`. The response shape depends on whether one check type or multiple were launched: * Single-check launch - the response describes that one check directly, with the resolved `config` block (defaults merged with user overrides) included so you can confirm exactly what the check is running with. `config` is `null` for operations that take no configuration. * Multi-check launch - the response describes the parent group check, with a `children[]` array carrying one entry per launched operation. Each child entry includes its own `uuid`, `status`, resolved `config`, and `detail_url` so you can start polling each child without first fetching the parent.** (application/json)

Field	Description
`data`	**object**
`data.` `uuid`	**string (uuid)**
`data.` `check_number`	**string**
`data.` `check_type`	**string** The operation name when a single type was launched, or `group` when multiple types were launched and a parent check was created.

Field	Description
data. status	**string** **Enum:** `pending` , `processing` , `complete` , `failed` **Example:** `pending`
data. config	**object (dynamic) or null** The resolved configuration for the launched check (defaults merged with any user-supplied values). Present on single-check launches; `null` for operations that take no configuration. Omitted on group launches - see `children[]` for per-child configs.
data. detail_url	**string** Path to the check detail endpoint. **Example:** `/v1/checks/3e36e9ec-0c44-4d65-a2f7-37b1b0a4d57e`
data. children	**array of objects** Returned on group launches (i.e. when `check_type = group`). One entry per launched operation, in the order they were requested. Each child has already been queued and can be polled independently via its `detail_url` .
data. children[]. uuid	**string (uuid)**
data. children[]. check_type	**string** **Example:** `court_check`
data. children[]. status	**string** **Enum:** `pending` , `processing` , `complete` , `failed` **Example:** `pending`
data. children[]. config	**object (dynamic) or null** The resolved configuration for this child (defaults merged with any user-supplied values). `null` for operations that take no configuration.
data. children[]. detail_url	**string** **Example:** `/v1/checks/4d7e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
api_reference	**string (uuid)**

Sample response

```
{
    "data": {
        "uuid": "00000000-0000-0000-0000-000000000000",
        "check_number": "string",
        "check_type": "string",
        "status": "pending",
        "config": null,
```

```
        "detail_url": "/v1/checks/3e36e9ec-0c44-4d65-a2f7-37b1b0a4d57e",
        "children": [
            {
                "uuid": "00000000-0000-0000-0000-000000000000",
                "check_type": "court_check",
                "status": "pending",
                "config": null,
                "detail_url": "/v1/checks/4d7e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f"
            }
        ]
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


422 **Idempotency-Key was reused with a different request body** (application/json)

Field	Description
`message`	**string** **Example:** `Idempotency-Key was reused with a different request body.`

Sample response


```
{
    "message": "Idempotency-Key was reused with a different request body."
}
```


Standard error responses: 400, 401, 402, 403, 404, 429, 5XX (see Common error responses).

Entities

The individuals, businesses, vessels and properties you screen, within a program.

List entities in a program

GET `/programs/{program_uuid}/entities`

Returns the entities in the specified program. Results are paginated.

Filters

The following filters can be applied via the `filter[...]` query parameters:

- `entity_type` - one of `individual`, `business`, `vessel`, `property`
- `risk_level` - one of `low`, `medium`, `high`, or `none` for entities with no risk assigned
- `flags` - return entities that have the given flag (e.g. `pep`, `sanction`, etc.)
- `reference_number` - exact match on the customer-supplied reference number

The top-level `archived` query parameter accepts `true` (only archived) or `false` (only non-archived); omitting it returns both.

Sorting

The `sort` query parameter accepts `created_at` (default: `-created_at`, newest first) or `entity_name`. Prefix with `-` for descending order.

Path parameters

Field	Description
`program_uuid` REQUIRED	**string (uuid)** The program UUID **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`

Query parameters

Field	Description
`page`	**integer** The page number to return **Example:** `1`
`per_page`	**integer** The number of entities per page (defaults to 30, max 500) **Example:** `30`
`filter[entity_type]`	**string** Only entities of the given type: `individual`, `business`, `vessel` or `property`. **Enum:** `individual`, `business`, `vessel`, `property`
`filter[risk_level]`	**string** Use `none` to return entities with no risk assigned **Enum:** `low`, `medium`, `high`, `none`
`filter[flags]`	**string** Return entities flagged with the given value **Enum:** `sanction`, `pep`, `adverse_media`, `fraud`, `hardship`, `bankruptcy`, `deceased`, `court_actions`, `deregistered`, `external_administration`
`filter[reference_number]`	**string** Only the entity with the given customer reference number. **Example:** `CUST-12345`

Field	Description
sort	**string** Field to sort by; prefix with `-` for descending order **Enum:** `created_at`, `-created_at`, `entity_name`, `-entity_name` **Example:** `-created_at`
archived	**boolean** Filter by archive status. `true` - only archived entities `false` - only non-archived entities Note: The API will accept the string 'true' or the number 1 for true and the string 'false' or the number 0 for false as well as the boolean values.

Responses

200 **Entities list response** (application/json)

Field	Description
data	**array of objects** An array of entities
data[]. uuid	**string (uuid)** The entity's UUID **Example:** `123e4567-e89b-12d3-a456-426614174000`
data[]. entity_number	**string** A short obfuscated public identifier for the entity (e.g. "E-A1-B2C-3D4") **Example:** `E-A1-B2C-3D4`
data[]. reference_number	**string or null** An optional customer-supplied reference number, unique within the program **Example:** `CUST-12345`
data[]. entity_type	**string** The type of entity: `individual` - A natural person `business` - A registered business `vessel` - A vessel `property` - A real estate property **Enum:** `individual`, `business`, `vessel`, `property` **Example:** `individual`

Field	Description
data[]. entity_name	**string or null** The display name of the entity, computed from the relevant type-specific fields **Example:** `Jane A Doe`
data[]. first_name	**string or null** First name (individuals only) **Example:** `Jane`
data[]. middle_name	**string or null** Middle name (individuals only) **Example:** `A`
data[]. last_name	**string or null** Last name (individuals only) **Example:** `Doe`
data[]. birth_date	**string (date) or null** Date of birth in `YYYY-MM-DD` format (individuals only) **Example:** `1980-05-15`
data[]. business_name	**string or null** Business name (businesses only) **Example:** `Acme Pty Ltd`
data[]. business_number	**string or null** Business identifier (businesses only); format depends on `business_number_type` **Example:** `12345678901`
data[]. business_number_type	**string or null** The type of `business_number`: `au_abn` - Australian Business Number (11 digits) `au_acn` - Australian Company Number (9 digits) `uk_crn` - UK Company Registration Number (8 alphanumeric) `other` - Other identifier (alphanumeric, max 32 chars) **Enum:** `au_abn`, `au_acn`, `uk_crn`, `other` **Example:** `au_abn`
data[]. vessel_name	**string or null** Vessel name (vessels only) **Example:** `HMAS Sydney`

Field	Description
data[]. property_name	**string or null** Property name (properties only) **Example:** `1 Example Street, Sydney`
data[]. address_line1	**string or null** First line of address **Example:** `1 Example Street`
data[]. address_line2	**string or null** Second line of address **Example:** `Unit 4`
data[]. address_suburb	**string or null** Suburb / locality **Example:** `Sydney`
data[]. address_state	**string or null** State / region (Australian state code if `address_country` is `AU`) **Example:** `NSW`
data[]. address_postcode	**string or null** Postal / zip code **Example:** `2000`
data[]. address_country	**string or null** Two-letter ISO 3166-1 alpha-2 country code (e.g. `AU`). Either case is accepted; responses use uppercase. **Example:** `AU`
data[]. phone1	**string or null** Primary phone number **Example:** `+61412345678`
data[]. phone2	**string or null** Secondary phone number
data[]. phone3	**string or null** Additional phone number
data[]. phone4	**string or null** Additional phone number

Field	Description
data[]. email1	**string (email) or null** Primary email address **Example:** `jane.doe@example.com`
data[]. email2	**string (email) or null** Secondary email address
data[]. email3	**string (email) or null** Additional email address
data[]. email4	**string (email) or null** Additional email address
data[]. flags	**array of strings** Manual flags applied to the entity. Permitted values: `sanction` - Sanctions `pep` - Politically Exposed `adverse_media` - Adverse Media `fraud` - Fraud `hardship` - Hardship `bankruptcy` - Bankruptcy `deceased` - Deceased `court_actions` - Court Actions `deregistered` - De-Registered `external_administration` - External Administration
data[]. risk_level	**string or null** Risk level assigned to the entity: `low` - Low `medium` - Medium `high` - High `null` - no risk assigned (the risk has not yet been established) **Enum:** `low`, `medium`, `high` **Example:** `low`
data[]. archived_at	**string (date-time) or null** ISO 8601 timestamp at which the entity was archived (null if not archived) **Example:** `2025-04-01T10:00:00Z`

Field	Description
data[]. deleted_at	**string (date-time) or null** ISO 8601 timestamp at which the entity was soft-deleted (null if not deleted). Only populated on the response from a DELETE call. **Example:** `2025-04-01T10:00:00Z`
data[]. program_uuid	**string (uuid)** UUID of the program this entity belongs to **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data[]. created_at	**string (date-time)** ISO 8601 timestamp at which the entity was created **Example:** `2025-01-01T00:00:00Z`
data[]. updated_at	**string (date-time)** ISO 8601 timestamp at which the entity was last updated **Example:** `2025-01-01T00:00:00Z`
meta	**object**
meta. current_page	**integer** **Example:** `1`
meta. per_page	**integer** **Example:** `30`
meta. total	**integer** **Example:** `1`
meta. last_page	**integer** **Example:** `1`
api_reference	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs and audit trail.

Sample response


```
{
    "data": [
        {
            "uuid": "123e4567-e89b-12d3-a456-426614174000",
            "entity_number": "E-A1-B2C-3D4",
            "reference_number": "CUST-12345",
            "entity_type": "individual",
            "entity_name": "Jane A Doe",
            "first_name": "Jane",
```

```
            "middle_name": "A",
            "last_name": "Doe",
            "birth_date": "1980-05-15",
            "business_name": "Acme Pty Ltd",
            "business_number": "12345678901",
            "business_number_type": "au_abn",
            "vessel_name": "HMAS Sydney",
            "property_name": "1 Example Street, Sydney",
            "address_line1": "1 Example Street",
            "address_line2": "Unit 4",
            "address_suburb": "Sydney",
            "address_state": "NSW",
            "address_postcode": "2000",
            "address_country": "AU",
            "phone1": "+61412345678",
            "phone2": "string",
            "phone3": "string",
            "phone4": "string",
            "email1": "jane.doe@example.com",
            "email2": "user@example.com",
            "email3": "user@example.com",
            "email4": "user@example.com",
            "flags": [
                "pep",
                "adverse_media"
            ],
            "risk_level": "low",
            "archived_at": "2025-04-01T10:00:00Z",
            "deleted_at": "2025-04-01T10:00:00Z",
            "program_uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
            "created_at": "2025-01-01T00:00:00Z",
            "updated_at": "2025-01-01T00:00:00Z"
        }
    ],
    "meta": {
        "current_page": 1,
        "per_page": 30,
        "total": 1,
        "last_page": 1
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 402, 403, 429, 5XX (see Common error responses).

Create an entity in a program

POST `/programs/{program_uuid}/entities`

Create a new entity in the specified program.

The supplied fields must be appropriate for the chosen `entity_type`:

- `individual` - requires at least one of `first_name` / `last_name`. May supply `birth_date`. Business, vessel and property fields are ignored.

`business` - requires `business_name`. Optionally `business_number` + `business_number_type`. Individual, vessel and property fields are ignored.

`vessel` - requires `vessel_name`. Individual, business and property fields are ignored.

`property` - requires `property_name`. Address fields are optional. Individual, business and vessel fields are ignored. Phone and email fields do not apply to properties and are ignored.

The created entity record is returned in the response.

Idempotency

This endpoint accepts the `Idempotency-Key` header so that retries on network failure are safe. See the Idempotency section of the API guide for the full contract.

Path parameters

Field	Description
`program_uuid` REQUIRED	**string (uuid)** The program UUID

Header parameters

Field	Description
`Idempotency-Key`	**string [max 255 characters]** Optional client-generated key (typically a UUID) that lets you safely retry a write request without risk of duplicate work or duplicate billing. A retry with the same key and body returns the original response verbatim, with the `Idempotent-Replay: true` response header. Honoured on `POST` requests only. See the Idempotency section of the API guide for the full contract. **Example:** `8c4d3a18-2f63-4f9a-9f3a-9b1f5a7c2d4f`

Request body

The entity details

Field	Description
`reference_number`	**string or null** An optional customer-supplied reference number, must be unique within the program **Example:** `CUST-12345`

Field	Description
`entity_type` REQUIRED	**string** **Enum:** `individual`, `business`, `vessel`, `property` **Example:** `individual`
`first_name`	**string or null** **Example:** `Jane`
`middle_name`	**string or null** **Example:** `A`
`last_name`	**string or null** **Example:** `Doe`
`birth_date`	**string (date) or null** Date of birth in `YYYY-MM-DD` format **Example:** `1980-05-15`
`business_name`	**string or null**
`business_number`	**string or null**
`business_number_type`	**string or null** **Enum:** `au_abn`, `au_acn`, `uk_crn`, `other`
`vessel_name`	**string or null**
`property_name`	**string or null**
`address_line1`	**string or null**
`address_line2`	**string or null**
`address_suburb`	**string or null**
`address_state`	**string or null**
`address_postcode`	**string or null**
`address_country`	**string or null** **Example:** `AU`
`phone1`	**string or null**
`phone2`	**string or null**
`phone3`	**string or null**
`phone4`	**string or null**
`email1`	**string (email) or null**

Field	Description
`email2`	**string (email) or null**
`email3`	**string (email) or null**
`email4`	**string (email) or null**
`flags`	**array of strings or null** **Enum:** `sanction`, `pep`, `adverse_media`, `fraud`, `hardship`, `bankruptcy`, `deceased`, `court_actions`, `deregistered`, `external_administration`
`risk_level`	**string or null** An explicit `null` creates the entity with no risk assigned. When omitted entirely, the program's `default_risk_level` is applied (falling back to `low`). **Enum:** `low`, `medium`, `high` **Example:** `low`

Sample request

```
{
    "reference_number": "CUST-12345",
    "entity_type": "individual",
    "first_name": "Jane",
    "middle_name": "A",
    "last_name": "Doe",
    "birth_date": "1980-05-15",
    "business_name": "string",
    "business_number": "string",
    "business_number_type": "au_abn",
    "vessel_name": "string",
    "property_name": "string",
    "address_line1": "string",
    "address_line2": "string",
    "address_suburb": "string",
    "address_state": "string",
    "address_postcode": "string",
    "address_country": "AU",
    "phone1": "string",
    "phone2": "string",
    "phone3": "string",
    "phone4": "string",
    "email1": "user@example.com",
    "email2": "user@example.com",
    "email3": "user@example.com",
    "email4": "user@example.com",
    "flags": [
        "sanction"
    ],
    "risk_level": "low"
}
```

Responses

201 **Entity created** (application/json)

Field	Description
`data`	**object**
`data.` `uuid`	**string (uuid)** The entity's UUID **Example:** `123e4567-e89b-12d3-a456-426614174000`
`data.` `entity_number`	**string** A short obfuscated public identifier for the entity (e.g. "E-A1-B2C-3D4") **Example:** `E-A1-B2C-3D4`
`data.` `reference_number`	**string or null** An optional customer-supplied reference number, unique within the program **Example:** `CUST-12345`
`data.` `entity_type`	**string** The type of entity: `individual` - A natural person `business` - A registered business `vessel` - A vessel `property` - A real estate property **Enum:** `individual`, `business`, `vessel`, `property` **Example:** `individual`
`data.` `entity_name`	**string or null** The display name of the entity, computed from the relevant type-specific fields **Example:** `Jane A Doe`
`data.` `first_name`	**string or null** First name (individuals only) **Example:** `Jane`
`data.` `middle_name`	**string or null** Middle name (individuals only) **Example:** `A`
`data.` `last_name`	**string or null** Last name (individuals only) **Example:** `Doe`

Field	Description
data. birth_date	**string (date) or null** Date of birth in `YYYY-MM-DD` format (individuals only) **Example:** `1980-05-15`
data. business_name	**string or null** Business name (businesses only) **Example:** `Acme Pty Ltd`
data. business_number	**string or null** Business identifier (businesses only); format depends on `business_number_type` **Example:** `12345678901`
data. business_number_type	**string or null** The type of `business_number`: `au_abn` - Australian Business Number (11 digits) `au_acn` - Australian Company Number (9 digits) `uk_crn` - UK Company Registration Number (8 alphanumeric) `other` - Other identifier (alphanumeric, max 32 chars) **Enum:** `au_abn`, `au_acn`, `uk_crn`, `other` **Example:** `au_abn`
data. vessel_name	**string or null** Vessel name (vessels only) **Example:** `HMAS Sydney`
data. property_name	**string or null** Property name (properties only) **Example:** `1 Example Street, Sydney`
data. address_line1	**string or null** First line of address **Example:** `1 Example Street`
data. address_line2	**string or null** Second line of address **Example:** `Unit 4`
data. address_suburb	**string or null** Suburb / locality **Example:** `Sydney`

Field	Description
data. address_state	**string or null** State / region (Australian state code if `address_country` is `AU`) **Example:** `NSW`
data. address_postcode	**string or null** Postal / zip code **Example:** `2000`
data. address_country	**string or null** Two-letter ISO 3166-1 alpha-2 country code (e.g. `AU`). Either case is accepted; responses use uppercase. **Example:** `AU`
data. phone1	**string or null** Primary phone number **Example:** `+61412345678`
data. phone2	**string or null** Secondary phone number
data. phone3	**string or null** Additional phone number
data. phone4	**string or null** Additional phone number
data. email1	**string (email) or null** Primary email address **Example:** `jane.doe@example.com`
data. email2	**string (email) or null** Secondary email address
data. email3	**string (email) or null** Additional email address
data. email4	**string (email) or null** Additional email address

Field	Description
data. flags	**array of strings** Manual flags applied to the entity. Permitted values: `sanction` - Sanctions `pep` - Politically Exposed `adverse_media` - Adverse Media `fraud` - Fraud `hardship` - Hardship `bankruptcy` - Bankruptcy `deceased` - Deceased `court_actions` - Court Actions `deregistered` - De-Registered `external_administration` - External Administration
data. risk_level	**string or null** Risk level assigned to the entity: `low` - Low `medium` - Medium `high` - High `null` - no risk assigned (the risk has not yet been established) **Enum:** `low`, `medium`, `high` **Example:** `low`
data. archived_at	**string (date-time) or null** ISO 8601 timestamp at which the entity was archived (null if not archived) **Example:** `2025-04-01T10:00:00Z`
data. deleted_at	**string (date-time) or null** ISO 8601 timestamp at which the entity was soft-deleted (null if not deleted). Only populated on the response from a DELETE call. **Example:** `2025-04-01T10:00:00Z`
data. program_uuid	**string (uuid)** UUID of the program this entity belongs to **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. created_at	**string (date-time)** ISO 8601 timestamp at which the entity was created **Example:** `2025-01-01T00:00:00Z`

Field	Description
data. updated_at	**string (date-time)** ISO 8601 timestamp at which the entity was last updated **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "123e4567-e89b-12d3-a456-426614174000",
        "entity_number": "E-A1-B2C-3D4",
        "reference_number": "CUST-12345",
        "entity_type": "individual",
        "entity_name": "Jane A Doe",
        "first_name": "Jane",
        "middle_name": "A",
        "last_name": "Doe",
        "birth_date": "1980-05-15",
        "business_name": "Acme Pty Ltd",
        "business_number": "12345678901",
        "business_number_type": "au_abn",
        "vessel_name": "HMAS Sydney",
        "property_name": "1 Example Street, Sydney",
        "address_line1": "1 Example Street",
        "address_line2": "Unit 4",
        "address_suburb": "Sydney",
        "address_state": "NSW",
        "address_postcode": "2000",
        "address_country": "AU",
        "phone1": "+61412345678",
        "phone2": "string",
        "phone3": "string",
        "phone4": "string",
        "email1": "jane.doe@example.com",
        "email2": "user@example.com",
        "email3": "user@example.com",
        "email4": "user@example.com",
        "flags": [
            "pep",
            "adverse_media"
        ],
        "risk_level": "low",
        "archived_at": "2025-04-01T10:00:00Z",
        "deleted_at": "2025-04-01T10:00:00Z",
        "program_uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


409 **Conflict - a request with this Idempotency-Key is currently being processed** (application/json)

Field	Description
`message`	**string** **Example:** `A request with this Idempotency-Key is already being processed.`

Sample response

```
{
    "message": "A request with this Idempotency-Key is already being processed."
}
```

422 **Idempotency-Key was reused with a different request body** (application/json)

Field	Description
`message`	**string** **Example:** `Idempotency-Key was reused with a different request body.`

Sample response

```
{
    "message": "Idempotency-Key was reused with a different request body."
}
```

Standard error responses: 400, 401, 402, 403, 429, 5XX (see Common error responses).

Show an entity

GET `/entities/{entity_uuid}`

Returns a single entity by its UUID. Unknown UUIDs return `404`.

Path parameters

Field	Description
`entity_uuid` REQUIRED	**string (uuid)** The entity UUID

Responses

200 **Entity response** (application/json)

Field	Description

`data`	**object**
`data.` `uuid`	**string (uuid)** The entity's UUID **Example:** `123e4567-e89b-12d3-a456-426614174000`
`data.` `entity_number`	**string** A short obfuscated public identifier for the entity (e.g. "E-A1-B2C-3D4") **Example:** `E-A1-B2C-3D4`
`data.` `reference_number`	**string or null** An optional customer-supplied reference number, unique within the program **Example:** `CUST-12345`
`data.` `entity_type`	**string** The type of entity: `individual` - A natural person `business` - A registered business `vessel` - A vessel `property` - A real estate property **Enum:** `individual`, `business`, `vessel`, `property` **Example:** `individual`
`data.` `entity_name`	**string or null** The display name of the entity, computed from the relevant type-specific fields **Example:** `Jane A Doe`
`data.` `first_name`	**string or null** First name (individuals only) **Example:** `Jane`
`data.` `middle_name`	**string or null** Middle name (individuals only) **Example:** `A`
`data.` `last_name`	**string or null** Last name (individuals only) **Example:** `Doe`
`data.` `birth_date`	**string (date) or null** Date of birth in `YYYY-MM-DD` format (individuals only) **Example:** `1980-05-15`

data. business_name	**string or null** Business name (businesses only) **Example:** `Acme Pty Ltd`
data. business_number	**string or null** Business identifier (businesses only); format depends on `business_number_type` **Example:** `12345678901`
data. business_number_type	**string or null** The type of `business_number`: `au_abn` - Australian Business Number (11 digits) `au_acn` - Australian Company Number (9 digits) `uk_crn` - UK Company Registration Number (8 alphanumeric) `other` - Other identifier (alphanumeric, max 32 chars) **Enum:** `au_abn`, `au_acn`, `uk_crn`, `other` **Example:** `au_abn`
data. vessel_name	**string or null** Vessel name (vessels only) **Example:** `HMAS Sydney`
data. property_name	**string or null** Property name (properties only) **Example:** `1 Example Street, Sydney`
data. address_line1	**string or null** First line of address **Example:** `1 Example Street`
data. address_line2	**string or null** Second line of address **Example:** `Unit 4`
data. address_suburb	**string or null** Suburb / locality **Example:** `Sydney`
data. address_state	**string or null** State / region (Australian state code if `address_country` is `AU`) **Example:** `NSW`

data. address_postcode	**string or null** Postal / zip code **Example:** `2000`
data. address_country	**string or null** Two-letter ISO 3166-1 alpha-2 country code (e.g. `AU`). Either case is accepted; responses use uppercase. **Example:** `AU`
data. phone1	**string or null** Primary phone number **Example:** `+61412345678`
data. phone2	**string or null** Secondary phone number
data. phone3	**string or null** Additional phone number
data. phone4	**string or null** Additional phone number
data. email1	**string (email) or null** Primary email address **Example:** `jane.doe@example.com`
data. email2	**string (email) or null** Secondary email address
data. email3	**string (email) or null** Additional email address
data. email4	**string (email) or null** Additional email address

data. flags	**array of strings** Manual flags applied to the entity. Permitted values: `sanction` - Sanctions `pep` - Politically Exposed `adverse_media` - Adverse Media `fraud` - Fraud `hardship` - Hardship `bankruptcy` - Bankruptcy `deceased` - Deceased `court_actions` - Court Actions `deregistered` - De-Registered `external_administration` - External Administration
data. risk_level	**string or null** Risk level assigned to the entity: `low` - Low `medium` - Medium `high` - High `null` - no risk assigned (the risk has not yet been established) **Enum:** `low`, `medium`, `high` **Example:** `low`
data. archived_at	**string (date-time) or null** ISO 8601 timestamp at which the entity was archived (null if not archived) **Example:** `2025-04-01T10:00:00Z`
data. deleted_at	**string (date-time) or null** ISO 8601 timestamp at which the entity was soft-deleted (null if not deleted). Only populated on the response from a DELETE call. **Example:** `2025-04-01T10:00:00Z`
data. program_uuid	**string (uuid)** UUID of the program this entity belongs to **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. created_at	**string (date-time)** ISO 8601 timestamp at which the entity was created **Example:** `2025-01-01T00:00:00Z`

data. updated_at	**string (date-time)** ISO 8601 timestamp at which the entity was last updated **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "123e4567-e89b-12d3-a456-426614174000",
        "entity_number": "E-A1-B2C-3D4",
        "reference_number": "CUST-12345",
        "entity_type": "individual",
        "entity_name": "Jane A Doe",
        "first_name": "Jane",
        "middle_name": "A",
        "last_name": "Doe",
        "birth_date": "1980-05-15",
        "business_name": "Acme Pty Ltd",
        "business_number": "12345678901",
        "business_number_type": "au_abn",
        "vessel_name": "HMAS Sydney",
        "property_name": "1 Example Street, Sydney",
        "address_line1": "1 Example Street",
        "address_line2": "Unit 4",
        "address_suburb": "Sydney",
        "address_state": "NSW",
        "address_postcode": "2000",
        "address_country": "AU",
        "phone1": "+61412345678",
        "phone2": "string",
        "phone3": "string",
        "phone4": "string",
        "email1": "jane.doe@example.com",
        "email2": "user@example.com",
        "email3": "user@example.com",
        "email4": "user@example.com",
        "flags": [
            "pep",
            "adverse_media"
        ],
        "risk_level": "low",
        "archived_at": "2025-04-01T10:00:00Z",
        "deleted_at": "2025-04-01T10:00:00Z",
        "program_uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 402, 403, 404, 429, 5XX (see Common error responses).

Update an entity

PATCH `/entities/{entity_uuid}`

Update an existing entity using PATCH semantics: only the fields you supply are changed, everything else is left as-is. Pass `null` to clear an optional field.

Notes:

- `entity_type` cannot be changed after creation. It is silently ignored if supplied.
- `birth_date` is silently ignored if the entity is locked because it has been used in an identification check.

Path parameters

Field	Description
`entity_uuid` REQUIRED	**string (uuid)** The entity UUID

Request body

The entity fields to update. All fields are optional.

Field	Description
`reference_number`	**string or null**
`first_name`	**string or null**
`middle_name`	**string or null**
`last_name`	**string or null**
`birth_date`	**string (date) or null**
`business_name`	**string or null**
`business_number`	**string or null**
`business_number_type`	**string or null** **Enum:** `au_abn`, `au_acn`, `uk_crn`, `other`
`vessel_name`	**string or null**
`property_name`	**string or null**
`address_line1`	**string or null**
`address_line2`	**string or null**
`address_suburb`	**string or null**

Field	Description
address_state	**string or null**
address_postcode	**string or null**
address_country	**string or null**
phone1	**string or null**
phone2	**string or null**
phone3	**string or null**
phone4	**string or null**
email1	**string (email) or null**
email2	**string (email) or null**
email3	**string (email) or null**
email4	**string (email) or null**
flags	**array of strings or null** **Enum:** `sanction`, `pep`, `adverse_media`, `fraud`, `hardship`, `bankruptcy`, `deceased`, `court_actions`, `deregistered`, `external_administration`
risk_level	**string or null** An explicit `null` clears the rating to "no risk assigned" **Enum:** `low`, `medium`, `high`

Sample request

```
{
    "reference_number": "string",
    "first_name": "string",
    "middle_name": "string",
    "last_name": "string",
    "birth_date": "2024-01-01",
    "business_name": "string",
    "business_number": "string",
    "business_number_type": "au_abn",
    "vessel_name": "string",
    "property_name": "string",
    "address_line1": "string",
    "address_line2": "string",
    "address_suburb": "string",
    "address_state": "string",
    "address_postcode": "string",
    "address_country": "string",
    "phone1": "string",
    "phone2": "string",
    "phone3": "string",
```

```
    "phone4": "string",
    "email1": "user@example.com",
    "email2": "user@example.com",
    "email3": "user@example.com",
    "email4": "user@example.com",
    "flags": [
        "sanction"
    ],
    "risk_level": "low"
}
```

Responses

200 **Entity updated** (application/json)

Field	Description
`data`	**object**
`data.` `uuid`	**string (uuid)** The entity's UUID **Example:** `123e4567-e89b-12d3-a456-426614174000`
`data.` `entity_number`	**string** A short obfuscated public identifier for the entity (e.g. "E-A1-B2C-3D4") **Example:** `E-A1-B2C-3D4`
`data.` `reference_number`	**string or null** An optional customer-supplied reference number, unique within the program **Example:** `CUST-12345`
`data.` `entity_type`	**string** The type of entity: `individual` - A natural person `business` - A registered business `vessel` - A vessel `property` - A real estate property **Enum:** `individual`, `business`, `vessel`, `property` **Example:** `individual`
`data.` `entity_name`	**string or null** The display name of the entity, computed from the relevant type-specific fields **Example:** `Jane A Doe`

Field	Description
data. first_name	**string or null** First name (individuals only) **Example:** `Jane`
data. middle_name	**string or null** Middle name (individuals only) **Example:** `A`
data. last_name	**string or null** Last name (individuals only) **Example:** `Doe`
data. birth_date	**string (date) or null** Date of birth in `YYYY-MM-DD` format (individuals only) **Example:** `1980-05-15`
data. business_name	**string or null** Business name (businesses only) **Example:** `Acme Pty Ltd`
data. business_number	**string or null** Business identifier (businesses only); format depends on `business_number_type` **Example:** `12345678901`
data. business_number_type	**string or null** The type of `business_number`: `au_abn` - Australian Business Number (11 digits) `au_acn` - Australian Company Number (9 digits) `uk_crn` - UK Company Registration Number (8 alphanumeric) `other` - Other identifier (alphanumeric, max 32 chars) **Enum:** `au_abn`, `au_acn`, `uk_crn`, `other` **Example:** `au_abn`
data. vessel_name	**string or null** Vessel name (vessels only) **Example:** `HMAS Sydney`
data. property_name	**string or null** Property name (properties only) **Example:** `1 Example Street, Sydney`

Field	Description
data. address_line1	**string or null** First line of address **Example:** `1 Example Street`
data. address_line2	**string or null** Second line of address **Example:** `Unit 4`
data. address_suburb	**string or null** Suburb / locality **Example:** `Sydney`
data. address_state	**string or null** State / region (Australian state code if `address_country` is `AU`) **Example:** `NSW`
data. address_postcode	**string or null** Postal / zip code **Example:** `2000`
data. address_country	**string or null** Two-letter ISO 3166-1 alpha-2 country code (e.g. `AU`). Either case is accepted; responses use uppercase. **Example:** `AU`
data. phone1	**string or null** Primary phone number **Example:** `+61412345678`
data. phone2	**string or null** Secondary phone number
data. phone3	**string or null** Additional phone number
data. phone4	**string or null** Additional phone number
data. email1	**string (email) or null** Primary email address **Example:** `jane.doe@example.com`

Field	Description
data. email2	**string (email) or null** Secondary email address
data. email3	**string (email) or null** Additional email address
data. email4	**string (email) or null** Additional email address
data. flags	**array of strings** Manual flags applied to the entity. Permitted values: `sanction` - Sanctions `pep` - Politically Exposed `adverse_media` - Adverse Media `fraud` - Fraud `hardship` - Hardship `bankruptcy` - Bankruptcy `deceased` - Deceased `court_actions` - Court Actions `deregistered` - De-Registered `external_administration` - External Administration
data. risk_level	**string or null** Risk level assigned to the entity: `low` - Low `medium` - Medium `high` - High `null` - no risk assigned (the risk has not yet been established) **Enum:** `low`, `medium`, `high` **Example:** `low`
data. archived_at	**string (date-time) or null** ISO 8601 timestamp at which the entity was archived (null if not archived) **Example:** `2025-04-01T10:00:00Z`
data. deleted_at	**string (date-time) or null** ISO 8601 timestamp at which the entity was soft-deleted (null if not deleted). Only populated on the response from a DELETE call. **Example:** `2025-04-01T10:00:00Z`

Field	Description
data. program_uuid	**string (uuid)** UUID of the program this entity belongs to **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. created_at	**string (date-time)** ISO 8601 timestamp at which the entity was created **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** ISO 8601 timestamp at which the entity was last updated **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "123e4567-e89b-12d3-a456-426614174000",
        "entity_number": "E-A1-B2C-3D4",
        "reference_number": "CUST-12345",
        "entity_type": "individual",
        "entity_name": "Jane A Doe",
        "first_name": "Jane",
        "middle_name": "A",
        "last_name": "Doe",
        "birth_date": "1980-05-15",
        "business_name": "Acme Pty Ltd",
        "business_number": "12345678901",
        "business_number_type": "au_abn",
        "vessel_name": "HMAS Sydney",
        "property_name": "1 Example Street, Sydney",
        "address_line1": "1 Example Street",
        "address_line2": "Unit 4",
        "address_suburb": "Sydney",
        "address_state": "NSW",
        "address_postcode": "2000",
        "address_country": "AU",
        "phone1": "+61412345678",
        "phone2": "string",
        "phone3": "string",
        "phone4": "string",
        "email1": "jane.doe@example.com",
        "email2": "user@example.com",
        "email3": "user@example.com",
        "email4": "user@example.com",
        "flags": [
            "pep",
            "adverse_media"
        ],
        "risk_level": "low",
```

```
        "archived_at": "2025-04-01T10:00:00Z",
        "deleted_at": "2025-04-01T10:00:00Z",
        "program_uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 402, 403, 404, 429, 5XX (see Common error responses).

Delete an entity

DELETE `/entities/{entity_uuid}`

Soft-deletes the entity and its dependent records (checks, ID checks, reports, events, notes and crossmatches). See the Soft Deletion section of the API guide for the full policy (30-day recovery window, portal-only restore, what gets cascaded).

If you need to keep the entity accessible for compliance / audit purposes but no longer want it surfaced in day-to-day workflows, prefer archiving instead of deleting. Archive the entity with `POST /v1/entities/{entity_uuid}/archive`; archived entities remain visible via list and show endpoints and can be filtered by the `archived` query parameter.

The records are no longer returned by list/show endpoints but are retained for compliance and audit purposes. The deleted entity record is returned in the response with `deleted_at` populated.

Path parameters

Field	Description
`entity_uuid` REQUIRED	**string (uuid)** The entity UUID

Responses

200 **Entity deleted** (application/json)

Field	Description
`data`	**object**
`data.` `uuid`	**string (uuid)** The entity's UUID **Example:** `123e4567-e89b-12d3-a456-426614174000`
`data.` `entity_number`	**string** A short obfuscated public identifier for the entity (e.g. "E-A1-B2C-3D4") **Example:** `E-A1-B2C-3D4`

Field	Description
data. reference_number	**string or null** An optional customer-supplied reference number, unique within the program **Example:** `CUST-12345`
data. entity_type	**string** The type of entity: `individual` - A natural person `business` - A registered business `vessel` - A vessel `property` - A real estate property **Enum:** `individual`, `business`, `vessel`, `property` **Example:** `individual`
data. entity_name	**string or null** The display name of the entity, computed from the relevant type-specific fields **Example:** `Jane A Doe`
data. first_name	**string or null** First name (individuals only) **Example:** `Jane`
data. middle_name	**string or null** Middle name (individuals only) **Example:** `A`
data. last_name	**string or null** Last name (individuals only) **Example:** `Doe`
data. birth_date	**string (date) or null** Date of birth in `YYYY-MM-DD` format (individuals only) **Example:** `1980-05-15`
data. business_name	**string or null** Business name (businesses only) **Example:** `Acme Pty Ltd`
data. business_number	**string or null** Business identifier (businesses only); format depends on `business_number_type` **Example:** `12345678901`

Field	Description
data. business_number_type	**string or null** The type of `business_number`: `au_abn` - Australian Business Number (11 digits) `au_acn` - Australian Company Number (9 digits) `uk_crn` - UK Company Registration Number (8 alphanumeric) `other` - Other identifier (alphanumeric, max 32 chars) **Enum:** `au_abn`, `au_acn`, `uk_crn`, `other` **Example:** `au_abn`
data. vessel_name	**string or null** Vessel name (vessels only) **Example:** `HMAS Sydney`
data. property_name	**string or null** Property name (properties only) **Example:** `1 Example Street, Sydney`
data. address_line1	**string or null** First line of address **Example:** `1 Example Street`
data. address_line2	**string or null** Second line of address **Example:** `Unit 4`
data. address_suburb	**string or null** Suburb / locality **Example:** `Sydney`
data. address_state	**string or null** State / region (Australian state code if `address_country` is `AU`) **Example:** `NSW`
data. address_postcode	**string or null** Postal / zip code **Example:** `2000`
data. address_country	**string or null** Two-letter ISO 3166-1 alpha-2 country code (e.g. `AU`). Either case is accepted; responses use uppercase. **Example:** `AU`

Field	Description
data. phone1	**string or null** Primary phone number **Example:** `+61412345678`
data. phone2	**string or null** Secondary phone number
data. phone3	**string or null** Additional phone number
data. phone4	**string or null** Additional phone number
data. email1	**string (email) or null** Primary email address **Example:** `jane.doe@example.com`
data. email2	**string (email) or null** Secondary email address
data. email3	**string (email) or null** Additional email address
data. email4	**string (email) or null** Additional email address
data. flags	**array of strings** Manual flags applied to the entity. Permitted values: `sanction` - Sanctions `pep` - Politically Exposed `adverse_media` - Adverse Media `fraud` - Fraud `hardship` - Hardship `bankruptcy` - Bankruptcy `deceased` - Deceased `court_actions` - Court Actions `deregistered` - De-Registered `external_administration` - External Administration

Field	Description
data. risk_level	**string or null** Risk level assigned to the entity: `low` - Low `medium` - Medium `high` - High `null` - no risk assigned (the risk has not yet been established) **Enum:** `low`, `medium`, `high` **Example:** `low`
data. archived_at	**string (date-time) or null** ISO 8601 timestamp at which the entity was archived (null if not archived) **Example:** `2025-04-01T10:00:00Z`
data. deleted_at	**string (date-time) or null** ISO 8601 timestamp at which the entity was soft-deleted (null if not deleted). Only populated on the response from a DELETE call. **Example:** `2025-04-01T10:00:00Z`
data. program_uuid	**string (uuid)** UUID of the program this entity belongs to **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. created_at	**string (date-time)** ISO 8601 timestamp at which the entity was created **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** ISO 8601 timestamp at which the entity was last updated **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "123e4567-e89b-12d3-a456-426614174000",
        "entity_number": "E-A1-B2C-3D4",
        "reference_number": "CUST-12345",
        "entity_type": "individual",
        "entity_name": "Jane A Doe",
        "first_name": "Jane",
        "middle_name": "A",
        "last_name": "Doe",
```

```
        "birth_date": "1980-05-15",
        "business_name": "Acme Pty Ltd",
        "business_number": "12345678901",
        "business_number_type": "au_abn",
        "vessel_name": "HMAS Sydney",
        "property_name": "1 Example Street, Sydney",
        "address_line1": "1 Example Street",
        "address_line2": "Unit 4",
        "address_suburb": "Sydney",
        "address_state": "NSW",
        "address_postcode": "2000",
        "address_country": "AU",
        "phone1": "+61412345678",
        "phone2": "string",
        "phone3": "string",
        "phone4": "string",
        "email1": "jane.doe@example.com",
        "email2": "user@example.com",
        "email3": "user@example.com",
        "email4": "user@example.com",
        "flags": [
            "pep",
            "adverse_media"
        ],
        "risk_level": "low",
        "archived_at": "2025-04-01T10:00:00Z",
        "deleted_at": "2025-04-01T10:00:00Z",
        "program_uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 402, 403, 404, 429, 5XX (see Common error responses).

Archive an entity

POST `/entities/{entity_uuid}/archive`

Archives the entity together with all checks, reports, events, notes and identification checks belonging to it. Archived records remain fully visible via list and show endpoints and can be included or excluded with the `archived` query parameter on the entities list.

Use archive instead of delete when you need to take a record out of day-to-day workflows but want to keep it accessible for compliance, audit or historical reporting. See the Soft Deletion section of the API guide for the differences between archive and delete.

Calling archive on an entity that is already archived returns the entity in its current state without recording a new audit entry or re-cascading to the child records.

The response body contains the entity with `archived_at` set to the ISO 8601 timestamp at which the archive took effect. (The `archived_at` field is always present on entity responses; it is `null` when the entity is not archived and an ISO 8601 timestamp when it is.)

Idempotency

This endpoint accepts the `Idempotency-Key` header so that retries on network failure are safe. See the Idempotency section of the API guide for the full contract.

Path parameters

Field	Description
`entity_uuid` REQUIRED	**string (uuid)** The entity UUID

Header parameters

Field	Description
`Idempotency-Key`	**string [max 255 characters]** Optional client-generated key (typically a UUID) that lets you safely retry a write request without risk of duplicate work or duplicate billing. A retry with the same key and body returns the original response verbatim, with the `Idempotent-Replay: true` response header. Honoured on `POST` requests only. See the Idempotency section of the API guide for the full contract. **Example:** `8c4d3a18-2f63-4f9a-9f3a-9b1f5a7c2d4f`

Responses

200 **Entity archived** (application/json)

Field	Description
`data`	**object**
`data.` `uuid`	**string (uuid)** The entity's UUID **Example:** `123e4567-e89b-12d3-a456-426614174000`
`data.` `entity_number`	**string** A short obfuscated public identifier for the entity (e.g. "E-A1-B2C-3D4") **Example:** `E-A1-B2C-3D4`
`data.` `reference_number`	**string or null** An optional customer-supplied reference number, unique within the program **Example:** `CUST-12345`

Field	Description
data. entity_type	**string** The type of entity: `individual` - A natural person `business` - A registered business `vessel` - A vessel `property` - A real estate property **Enum:** `individual`, `business`, `vessel`, `property` **Example:** `individual`
data. entity_name	**string or null** The display name of the entity, computed from the relevant type-specific fields **Example:** `Jane A Doe`
data. first_name	**string or null** First name (individuals only) **Example:** `Jane`
data. middle_name	**string or null** Middle name (individuals only) **Example:** `A`
data. last_name	**string or null** Last name (individuals only) **Example:** `Doe`
data. birth_date	**string (date) or null** Date of birth in `YYYY-MM-DD` format (individuals only) **Example:** `1980-05-15`
data. business_name	**string or null** Business name (businesses only) **Example:** `Acme Pty Ltd`
data. business_number	**string or null** Business identifier (businesses only); format depends on `business_number_type` **Example:** `12345678901`

Field	Description
data. business_number_type	**string or null** The type of `business_number`: `au_abn` - Australian Business Number (11 digits) `au_acn` - Australian Company Number (9 digits) `uk_crn` - UK Company Registration Number (8 alphanumeric) `other` - Other identifier (alphanumeric, max 32 chars) **Enum:** `au_abn`, `au_acn`, `uk_crn`, `other` **Example:** `au_abn`
data. vessel_name	**string or null** Vessel name (vessels only) **Example:** `HMAS Sydney`
data. property_name	**string or null** Property name (properties only) **Example:** `1 Example Street, Sydney`
data. address_line1	**string or null** First line of address **Example:** `1 Example Street`
data. address_line2	**string or null** Second line of address **Example:** `Unit 4`
data. address_suburb	**string or null** Suburb / locality **Example:** `Sydney`
data. address_state	**string or null** State / region (Australian state code if `address_country` is `AU`) **Example:** `NSW`
data. address_postcode	**string or null** Postal / zip code **Example:** `2000`
data. address_country	**string or null** Two-letter ISO 3166-1 alpha-2 country code (e.g. `AU`). Either case is accepted; responses use uppercase. **Example:** `AU`

Field	Description
data. phone1	**string or null** Primary phone number **Example:** `+61412345678`
data. phone2	**string or null** Secondary phone number
data. phone3	**string or null** Additional phone number
data. phone4	**string or null** Additional phone number
data. email1	**string (email) or null** Primary email address **Example:** `jane.doe@example.com`
data. email2	**string (email) or null** Secondary email address
data. email3	**string (email) or null** Additional email address
data. email4	**string (email) or null** Additional email address
data. flags	**array of strings** Manual flags applied to the entity. Permitted values: `sanction` - Sanctions `pep` - Politically Exposed `adverse_media` - Adverse Media `fraud` - Fraud `hardship` - Hardship `bankruptcy` - Bankruptcy `deceased` - Deceased `court_actions` - Court Actions `deregistered` - De-Registered `external_administration` - External Administration

Field	Description
data. risk_level	**string or null** Risk level assigned to the entity: `low` - Low `medium` - Medium `high` - High `null` - no risk assigned (the risk has not yet been established) **Enum:** `low`, `medium`, `high` **Example:** `low`
data. archived_at	**string (date-time) or null** ISO 8601 timestamp at which the entity was archived (null if not archived) **Example:** `2025-04-01T10:00:00Z`
data. deleted_at	**string (date-time) or null** ISO 8601 timestamp at which the entity was soft-deleted (null if not deleted). Only populated on the response from a DELETE call. **Example:** `2025-04-01T10:00:00Z`
data. program_uuid	**string (uuid)** UUID of the program this entity belongs to **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. created_at	**string (date-time)** ISO 8601 timestamp at which the entity was created **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** ISO 8601 timestamp at which the entity was last updated **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "123e4567-e89b-12d3-a456-426614174000",
        "entity_number": "E-A1-B2C-3D4",
        "reference_number": "CUST-12345",
        "entity_type": "individual",
        "entity_name": "Jane A Doe",
        "first_name": "Jane",
        "middle_name": "A",
        "last_name": "Doe",
```

```
        "birth_date": "1980-05-15",
        "business_name": "Acme Pty Ltd",
        "business_number": "12345678901",
        "business_number_type": "au_abn",
        "vessel_name": "HMAS Sydney",
        "property_name": "1 Example Street, Sydney",
        "address_line1": "1 Example Street",
        "address_line2": "Unit 4",
        "address_suburb": "Sydney",
        "address_state": "NSW",
        "address_postcode": "2000",
        "address_country": "AU",
        "phone1": "+61412345678",
        "phone2": "string",
        "phone3": "string",
        "phone4": "string",
        "email1": "jane.doe@example.com",
        "email2": "user@example.com",
        "email3": "user@example.com",
        "email4": "user@example.com",
        "flags": [
            "pep",
            "adverse_media"
        ],
        "risk_level": "low",
        "archived_at": "2025-04-01T10:00:00Z",
        "deleted_at": "2025-04-01T10:00:00Z",
        "program_uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


409 **Conflict - a request with this Idempotency-Key is currently being processed** (application/json)

Field	Description
message	**string** **Example:** `A request with this Idempotency-Key is already being processed.`

Sample response


```
{
    "message": "A request with this Idempotency-Key is already being processed."
}
```


422 **Idempotency-Key was reused with a different request body** (application/json)

Field	Description
message	**string** **Example:** `Idempotency-Key was reused with a different request body.`

Sample response

```
{
    "message": "Idempotency-Key was reused with a different request body."
}
```

Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

Unarchive an entity

POST `/entities/{entity_uuid}/unarchive`

Restores an archived entity together with all of its checks, reports, events, notes and identification checks. The entity returns to the active set and resumes appearing in monitor runs and default reports.

Calling unarchive on an entity that is not currently archived returns the entity in its current state without recording a new audit entry or re-cascading to the child records.

The response body contains the entity with `archived_at` set back to `null`. (The `archived_at` field is always present on entity responses; it is `null` when the entity is not archived and an ISO 8601 timestamp when it is.)

Idempotency

This endpoint accepts the `Idempotency-Key` header so that retries on network failure are safe. See the Idempotency section of the API guide for the full contract.

Path parameters

Field	Description
`entity_uuid` REQUIRED	**string (uuid)** The entity UUID

Header parameters

Field	Description
`Idempotency-Key`	**string [max 255 characters]** Optional client-generated key (typically a UUID) that lets you safely retry a write request without risk of duplicate work or duplicate billing. A retry with the same key and body returns the original response verbatim, with the `Idempotent-Replay: true` response header. Honoured on `POST` requests only. See the Idempotency section of the API guide for the full contract. **Example:** `8c4d3a18-2f63-4f9a-9f3a-9b1f5a7c2d4f`

Responses

200 **Entity unarchived** (application/json)

Field	Description
`data`	**object**
`data.` `uuid`	**string (uuid)** The entity's UUID **Example:** `123e4567-e89b-12d3-a456-426614174000`
`data.` `entity_number`	**string** A short obfuscated public identifier for the entity (e.g. "E-A1-B2C-3D4") **Example:** `E-A1-B2C-3D4`
`data.` `reference_number`	**string or null** An optional customer-supplied reference number, unique within the program **Example:** `CUST-12345`
`data.` `entity_type`	**string** The type of entity: `individual` - A natural person `business` - A registered business `vessel` - A vessel `property` - A real estate property **Enum:** `individual`, `business`, `vessel`, `property` **Example:** `individual`
`data.` `entity_name`	**string or null** The display name of the entity, computed from the relevant type-specific fields **Example:** `Jane A Doe`
`data.` `first_name`	**string or null** First name (individuals only) **Example:** `Jane`
`data.` `middle_name`	**string or null** Middle name (individuals only) **Example:** `A`
`data.` `last_name`	**string or null** Last name (individuals only) **Example:** `Doe`

Field	Description
data. birth_date	**string (date) or null** Date of birth in `YYYY-MM-DD` format (individuals only) **Example:** `1980-05-15`
data. business_name	**string or null** Business name (businesses only) **Example:** `Acme Pty Ltd`
data. business_number	**string or null** Business identifier (businesses only); format depends on `business_number_type` **Example:** `12345678901`
data. business_number_type	**string or null** The type of `business_number`: `au_abn` - Australian Business Number (11 digits) `au_acn` - Australian Company Number (9 digits) `uk_crn` - UK Company Registration Number (8 alphanumeric) `other` - Other identifier (alphanumeric, max 32 chars) **Enum:** `au_abn`, `au_acn`, `uk_crn`, `other` **Example:** `au_abn`
data. vessel_name	**string or null** Vessel name (vessels only) **Example:** `HMAS Sydney`
data. property_name	**string or null** Property name (properties only) **Example:** `1 Example Street, Sydney`
data. address_line1	**string or null** First line of address **Example:** `1 Example Street`
data. address_line2	**string or null** Second line of address **Example:** `Unit 4`
data. address_suburb	**string or null** Suburb / locality **Example:** `Sydney`

Field	Description
data. address_state	**string or null** State / region (Australian state code if `address_country` is `AU`) **Example:** `NSW`
data. address_postcode	**string or null** Postal / zip code **Example:** `2000`
data. address_country	**string or null** Two-letter ISO 3166-1 alpha-2 country code (e.g. `AU`). Either case is accepted; responses use uppercase. **Example:** `AU`
data. phone1	**string or null** Primary phone number **Example:** `+61412345678`
data. phone2	**string or null** Secondary phone number
data. phone3	**string or null** Additional phone number
data. phone4	**string or null** Additional phone number
data. email1	**string (email) or null** Primary email address **Example:** `jane.doe@example.com`
data. email2	**string (email) or null** Secondary email address
data. email3	**string (email) or null** Additional email address
data. email4	**string (email) or null** Additional email address

Field	Description
data. flags	**array of strings** Manual flags applied to the entity. Permitted values: `sanction` - Sanctions `pep` - Politically Exposed `adverse_media` - Adverse Media `fraud` - Fraud `hardship` - Hardship `bankruptcy` - Bankruptcy `deceased` - Deceased `court_actions` - Court Actions `deregistered` - De-Registered `external_administration` - External Administration
data. risk_level	**string or null** Risk level assigned to the entity: `low` - Low `medium` - Medium `high` - High `null` - no risk assigned (the risk has not yet been established) **Enum:** `low`, `medium`, `high` **Example:** `low`
data. archived_at	**string (date-time) or null** ISO 8601 timestamp at which the entity was archived (null if not archived) **Example:** `2025-04-01T10:00:00Z`
data. deleted_at	**string (date-time) or null** ISO 8601 timestamp at which the entity was soft-deleted (null if not deleted). Only populated on the response from a DELETE call. **Example:** `2025-04-01T10:00:00Z`
data. program_uuid	**string (uuid)** UUID of the program this entity belongs to **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. created_at	**string (date-time)** ISO 8601 timestamp at which the entity was created **Example:** `2025-01-01T00:00:00Z`

Field	Description
data. updated_at	**string (date-time)** ISO 8601 timestamp at which the entity was last updated **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "123e4567-e89b-12d3-a456-426614174000",
        "entity_number": "E-A1-B2C-3D4",
        "reference_number": "CUST-12345",
        "entity_type": "individual",
        "entity_name": "Jane A Doe",
        "first_name": "Jane",
        "middle_name": "A",
        "last_name": "Doe",
        "birth_date": "1980-05-15",
        "business_name": "Acme Pty Ltd",
        "business_number": "12345678901",
        "business_number_type": "au_abn",
        "vessel_name": "HMAS Sydney",
        "property_name": "1 Example Street, Sydney",
        "address_line1": "1 Example Street",
        "address_line2": "Unit 4",
        "address_suburb": "Sydney",
        "address_state": "NSW",
        "address_postcode": "2000",
        "address_country": "AU",
        "phone1": "+61412345678",
        "phone2": "string",
        "phone3": "string",
        "phone4": "string",
        "email1": "jane.doe@example.com",
        "email2": "user@example.com",
        "email3": "user@example.com",
        "email4": "user@example.com",
        "flags": [
            "pep",
            "adverse_media"
        ],
        "risk_level": "low",
        "archived_at": "2025-04-01T10:00:00Z",
        "deleted_at": "2025-04-01T10:00:00Z",
        "program_uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


409 **Conflict - a request with this Idempotency-Key is currently being processed** (application/json)

Field	Description
`message`	**string** **Example:** `A request with this Idempotency-Key is already being processed.`

Sample response

```
{
    "message": "A request with this Idempotency-Key is already being processed."
}
```

422 **Idempotency-Key was reused with a different request body** (application/json)

Field	Description
`message`	**string** **Example:** `Idempotency-Key was reused with a different request body.`

Sample response

```
{
    "message": "Idempotency-Key was reused with a different request body."
}
```

Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

Events

Screening results that need review, raised by checks and monitors.

List events

GET `/events`

Returns events for the calling account. Results are paginated.

Each event represents a "thing your team should look at" produced by a screening check - typically a positive PEP/sanction match, a phone or email match, an adverse media hit, etc. Events have lifecycle status (`new` -> `investigating` -> `closed_*`) that you can transition via the PATCH event endpoint.

Events do not get deleted on their own; closing them sets their `status` to one of the `closed_*` values.

Filters

The following filters can be applied via the `filter[...]` query parameters:

`status` - one of `new`, `investigating`, `closed_dismissed`, `closed_duplicate`, `closed_error`, `closed_confirmed`

`check_type` - one of the operation types (e.g. `pep_sanction_check`, `phone_check`, `business_check`)

`entity_uuid` - only events for the given entity

`program_uuid` - only events for entities in the given program

`check_uuid` - only events from the given check

`created_at_from` - inclusive lower bound on event creation time (ISO 8601)

`created_at_to` - inclusive upper bound on event creation time (ISO 8601)

The top-level `archived` query parameter accepts `true` (only archived) or `false` (only non-archived); omitting it returns both.

Filters that target a uuid (entity, program, check) return an empty page when the target uuid does not exist on the calling account, identical to the behaviour for a uuid that does not exist at all.

Sorting

The `sort` query parameter accepts `created_at` (default: `-created_at`, newest first) or `status`. Prefix with `-` for descending order.

Query parameters

Field	Description
`page`	**integer** The page of results to return, starting at 1. **Example:** `1`
`per_page`	**integer** The number of events per page (defaults to 30, max 500) **Example:** `30`
`filter[status]`	**string** Only records with the given status. The endpoint description lists the values. **Enum:** `new`, `investigating`, `closed_dismissed`, `closed_duplicate`, `closed_error`, `closed_confirmed`
`filter[check_type]`	**string** Only records of the given check type. The endpoint description lists the values. **Example:** `pep_sanction_check`
`filter[entity_uuid]`	**string (uuid)** Only records for the given entity.
`filter[program_uuid]`	**string (uuid)** Only records for entities in the given program.

Field	Description
`filter[check_uuid]`	**string (uuid)** Only events raised by the given check.
`filter[created_at_from]`	**string (date-time)** Inclusive lower bound on `created_at` (ISO 8601). **Example:** `2025-01-01T00:00:00Z`
`filter[created_at_to]`	**string (date-time)** Inclusive upper bound on `created_at` (ISO 8601). **Example:** `2025-12-31T23:59:59Z`
`sort`	**string** Field to sort by; prefix with `-` for descending order **Enum:** `created_at`, `-created_at`, `status`, `-status` **Example:** `-created_at`
`archived`	**boolean** Filter by archive status. `true` - only archived events `false` - only non-archived events Note: The API will accept the string 'true' or the number 1 for true and the string 'false' or the number 0 for false as well as the boolean values.

Responses

200 **Events list response** (application/json)

Field	Description
`data`	**array of objects** An array of events
`data[].` `uuid`	**string (uuid)** The event's UUID **Example:** `4d7e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
`data[].` `event_number`	**string** A short obfuscated public identifier for the event (e.g. "V-A1-B2C-3D4") **Example:** `V-A1-B2C-3D4`

Field	Description
data[]. status	**string** Lifecycle status of the event: `new` - the event has just been raised and not yet triaged `investigating` - someone is actively working the event `closed_dismissed` - reviewed and dismissed (e.g. false positive) `closed_duplicate` - duplicates another event `closed_error` - caused by a data entry error `closed_confirmed` - confirmed and acted on **Enum:** `new`, `investigating`, `closed_dismissed`, `closed_duplicate`, `closed_error`, `closed_confirmed` **Example:** `new`
data[]. check_type	**string** The operation type of the check that produced this event (e.g. `pep_sanction_check`, `phone_check`, `business_check`). **Example:** `pep_sanction_check`
data[]. check_summary	**string or null** Short, human-readable summary of the parent check (denormalised from the linked check so that listing events does not require an N+1 fetch into checks). **Example:** `Searched for "Jane Doe" in PEP and Sanction lists for AU, NZ, US, GB, CA.`
data[]. check_response_index	**integer or null** For `separate` event grouping, the zero-based index into the parent check's `response` array that this event represents. `null` for `grouped` events that aggregate every match for the entity into a single event. **Example:** `0`
data[]. check_result_ids	**array of integers** Provider-side result identifiers that triggered this event. Combined with the parent check's `response`, this lets integrators pull the underlying match data for the event.
data[]. archived_at	**string (date-time) or null** ISO 8601 timestamp at which the event was archived (null if not archived)
data[]. deleted_at	**string (date-time) or null** ISO 8601 timestamp at which the event was soft-deleted (null if not deleted)

Field	Description
data[]. entity_uuid	**string (uuid)** UUID of the entity this event was raised against **Example:** `123e4567-e89b-12d3-a456-426614174000`
data[]. program_uuid	**string (uuid)** UUID of the program the entity belongs to **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data[]. check_uuid	**string (uuid)** UUID of the check that produced this event **Example:** `0a1b2c3d-4e5f-6789-abcd-ef0123456789`
data[]. notes_count	**integer** How many notes (system + user) are attached to the event. Only the count is included on the list endpoint; the detail endpoint additionally embeds the notes themselves. **Example:** `0`
data[]. created_at	**string (date-time)** ISO 8601 timestamp at which the event was created **Example:** `2025-01-01T00:00:00Z`
meta	**object**
meta. current_page	**integer** **Example:** `1`
meta. per_page	**integer** **Example:** `30`
meta. total	**integer** **Example:** `1`
meta. last_page	**integer** **Example:** `1`
api_reference	**string (uuid)**

Sample response


```
{
    "data": [
        {
            "uuid": "4d7e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
            "event_number": "V-A1-B2C-3D4",
            "status": "new",
```

```
            "check_type": "pep_sanction_check",
            "check_summary": "Searched for \"Jane Doe\" in PEP and Sanction lists for AU, NZ, US, GB, CA.",
            "check_response_index": 0,
            "check_result_ids": [
                12345,
                12346
            ],
            "archived_at": null,
            "deleted_at": null,
            "entity_uuid": "123e4567-e89b-12d3-a456-426614174000",
            "program_uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
            "check_uuid": "0a1b2c3d-4e5f-6789-abcd-ef0123456789",
            "notes_count": 0,
            "created_at": "2025-01-01T00:00:00Z"
        }
    ],
    "meta": {
        "current_page": 1,
        "per_page": 30,
        "total": 1,
        "last_page": 1
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 403, 429, 5XX (see Common error responses).

Show an event

GET `/events/{event_uuid}`

Returns a single event by its UUID, including the full notes collection.

Path parameters

Field	Description
`event_uuid` REQUIRED	**string (uuid)** The event UUID

Responses

200 **Event response** (application/json)

Field	Description
`data`	**object** Same as Event, plus the embedded `notes` collection. Returned by the show and update endpoints so callers do not need a follow-up request to read the notes.

Field	Description
data. uuid	**string (uuid)** The event's UUID **Example:** `4d7e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. event_number	**string** A short obfuscated public identifier for the event **Example:** `V-A1-B2C-3D4`
data. status	**string** Lifecycle status of the event **Enum:** `new`, `investigating`, `closed_dismissed`, `closed_duplicate`, `closed_error`, `closed_confirmed`
data. check_type	**string** The operation type of the parent check **Example:** `pep_sanction_check`
data. check_summary	**string or null** Short, human-readable summary of the parent check
data. check_response_index	**integer or null** Index into the parent check's `response` array (`null` for grouped events)
data. check_result_ids	**array of integers** Provider-side result identifiers that triggered this event
data. archived_at	**string (date-time) or null**
data. deleted_at	**string (date-time) or null**
data. entity_uuid	**string (uuid)**
data. program_uuid	**string (uuid)**
data. check_uuid	**string (uuid)**
data. notes_count	**integer**
data. notes	**array of objects** All notes attached to the event, oldest first.

Field	Description
data. notes[]. uuid	**string (uuid)** The note's UUID **Example:** `7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. notes[]. note_text	**string** The free-text content of the note (max 5000 characters) **Example:** `Confirmed match against verified passport details.`
data. notes[]. created_by	**object or null** Identifies who created the note. The `type` discriminator picks the shape of the remaining fields: `{ type: "user", uuid, username }` - a portal user created the note. `{ type: "api_key", uuid, label }` - an API key on this account created the note. `null` - rare legacy notes from before actor attribution was recorded.
data. notes[]. created_by. type	**string** Discriminator for the actor type. **Enum:** `user`, `api_key`
data. notes[]. created_by. uuid	**string (uuid)** UUID of the user or API key that created the note.
data. notes[]. created_by. username	**string** Username of the portal user. Present only when `type` is `user`.
data. notes[]. created_by. label	**string** Label of the API key. Present only when `type` is `api_key`.
data. notes[]. is_immutable	**boolean** When true, the note is an immutable audit-trail entry (for example, a bulk-update record) and cannot be edited or deleted through any channel. **Example:** `false`
data. notes[]. created_at	**string (date-time)** ISO 8601 timestamp at which the note was created **Example:** `2025-01-01T00:00:00Z`
data. notes[]. updated_at	**string (date-time)** ISO 8601 timestamp at which the note was last updated **Example:** `2025-01-01T00:00:00Z`

Field	Description
data. created_at	string (date-time)
api_reference	string (uuid)

Sample response


```
{
    "data": {
        "uuid": "4d7e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "event_number": "V-A1-B2C-3D4",
        "status": "new",
        "check_type": "pep_sanction_check",
        "check_summary": "string",
        "check_response_index": 0,
        "check_result_ids": [
            0
        ],
        "archived_at": "2024-01-01T00:00:00Z",
        "deleted_at": "2024-01-01T00:00:00Z",
        "entity_uuid": "00000000-0000-0000-0000-000000000000",
        "program_uuid": "00000000-0000-0000-0000-000000000000",
        "check_uuid": "00000000-0000-0000-0000-000000000000",
        "notes_count": 0,
        "notes": [
            {
                "uuid": "7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
                "note_text": "Confirmed match against verified passport details.",
                "created_by": {
                    "type": "user",
                    "uuid": "00000000-0000-0000-0000-000000000000",
                    "username": "string",
                    "label": "string"
                },
                "is_immutable": false,
                "created_at": "2025-01-01T00:00:00Z",
                "updated_at": "2025-01-01T00:00:00Z"
            }
        ],
        "created_at": "2024-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

Update an event

PATCH `/events/{event_uuid}`

Update an event's status and/or attach a note. At least one of `status` or `note` must be supplied; sending neither returns `400 No updates to apply`.

Status

The `status` field accepts any of the lifecycle values:

- `new` - the default starting state
- `investigating` - someone (or some integration) is actively working the event
- `closed_dismissed` - reviewed and dismissed (e.g. false positive)
- `closed_duplicate` - this event duplicates another event
- `closed_error` - the event was caused by a data entry error
- `closed_confirmed` - the event has been confirmed and acted on

Notes

The `note` field accepts up to 5000 characters. Each PATCH that supplies a `note` value creates a **new** note attached to the event - this endpoint does not update an existing note. Notes created via this endpoint are attributed to the calling API key (the response `notes[].created_by` carries `{ type: "api_key", uuid, label }`) and cannot be edited or deleted from the portal. Any API key on the same account can manage them via the notes PATCH / DELETE endpoints. They appear in the portal alongside notes created by portal users.

Author attribution

When this endpoint changes the status, the event's "last actioned by" attribution in the portal is cleared because an API key is not a user. The audit log records the API key and the request `api_reference` UUID, so you can always identify which integration made the change.

Path parameters

Field	Description
`event_uuid` REQUIRED	**string (uuid)** The event UUID

Request body

At least one of `status` or `note` must be supplied.

Field	Description
`status`	**string** **Enum:** `new`, `investigating`, `closed_dismissed`, `closed_duplicate`, `closed_error`, `closed_confirmed` **Example:** `investigating`
`note`	**string [max 5000 characters]** Free-text note to attach to the event. Max 5000 characters. **Example:** `Confirmed match against verified passport details.`

Sample request

```
{
    "status": "investigating",
    "note": "Confirmed match against verified passport details."
}
```

Responses

200 **Event updated** (application/json)

Field	Description
`data`	**object** Same as Event, plus the embedded `notes` collection. Returned by the show and update endpoints so callers do not need a follow-up request to read the notes.
`data.` `uuid`	**string (uuid)** The event's UUID **Example:** `4d7e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
`data.` `event_number`	**string** A short obfuscated public identifier for the event **Example:** `V-A1-B2C-3D4`
`data.` `status`	**string** Lifecycle status of the event **Enum:** `new`, `investigating`, `closed_dismissed`, `closed_duplicate`, `closed_error`, `closed_confirmed`
`data.` `check_type`	**string** The operation type of the parent check **Example:** `pep_sanction_check`
`data.` `check_summary`	**string or null** Short, human-readable summary of the parent check
`data.` `check_response_index`	**integer or null** Index into the parent check's `response` array (`null` for grouped events)
`data.` `check_result_ids`	**array of integers** Provider-side result identifiers that triggered this event
`data.` `archived_at`	**string (date-time) or null**
`data.` `deleted_at`	**string (date-time) or null**

Field	Description
data. entity_uuid	**string (uuid)**
data. program_uuid	**string (uuid)**
data. check_uuid	**string (uuid)**
data. notes_count	**integer**
data. notes	**array of objects** All notes attached to the event, oldest first.
data. notes[]. uuid	**string (uuid)** The note's UUID **Example:** `7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. notes[]. note_text	**string** The free-text content of the note (max 5000 characters) **Example:** `Confirmed match against verified passport details.`
data. notes[]. created_by	**object or null** Identifies who created the note. The `type` discriminator picks the shape of the remaining fields: `{ type: "user", uuid, username }` - a portal user created the note. `{ type: "api_key", uuid, label }` - an API key on this account created the note. `null` - rare legacy notes from before actor attribution was recorded.
data. notes[]. created_by. type	**string** Discriminator for the actor type. **Enum:** `user`, `api_key`
data. notes[]. created_by. uuid	**string (uuid)** UUID of the user or API key that created the note.
data. notes[]. created_by. username	**string** Username of the portal user. Present only when `type` is `user`.
data. notes[]. created_by. label	**string** Label of the API key. Present only when `type` is `api_key`.

Field	Description
data. notes[]. is_immutable	**boolean** When true, the note is an immutable audit-trail entry (for example, a bulk-update record) and cannot be edited or deleted through any channel. **Example:** `false`
data. notes[]. created_at	**string (date-time)** ISO 8601 timestamp at which the note was created **Example:** `2025-01-01T00:00:00Z`
data. notes[]. updated_at	**string (date-time)** ISO 8601 timestamp at which the note was last updated **Example:** `2025-01-01T00:00:00Z`
data. created_at	**string (date-time)**
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "4d7e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "event_number": "V-A1-B2C-3D4",
        "status": "new",
        "check_type": "pep_sanction_check",
        "check_summary": "string",
        "check_response_index": 0,
        "check_result_ids": [
            0
        ],
        "archived_at": "2024-01-01T00:00:00Z",
        "deleted_at": "2024-01-01T00:00:00Z",
        "entity_uuid": "00000000-0000-0000-0000-000000000000",
        "program_uuid": "00000000-0000-0000-0000-000000000000",
        "check_uuid": "00000000-0000-0000-0000-000000000000",
        "notes_count": 0,
        "notes": [
            {
                "uuid": "7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
                "note_text": "Confirmed match against verified passport details.",
                "created_by": {
                    "type": "user",
                    "uuid": "00000000-0000-0000-0000-000000000000",
                    "username": "string",
                    "label": "string"
                },
                "is_immutable": false,
                "created_at": "2025-01-01T00:00:00Z",
                "updated_at": "2025-01-01T00:00:00Z"
            }
        ],
```

```
        "created_at": "2024-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```

Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

ID Checks

Identity document checks run against entities.

List ID checks

GET `/id-checks`

Returns entity-bound identification checks for the calling account. Results are paginated.

Each record represents a synchronous document verification run against a saved entity (DVS or non-DVS). The list endpoint returns a compact envelope; use show ID check to read consent, verification details and the typed `details` block.

ID checks run synchronously - there is no `status` field. The `outcome` is populated when the check completes (`pass` , `fail` , or `pending` for in-flight IDPass invitations only; IDPass is not available via this API in v1).

Filters

- `program_uuid` - only checks for entities in the given program
- `entity_uuid` - only checks for the given entity
- `id_type` - one of the supported identification types (e.g. `drivers_licence` , `passport`)
- `check_type` - the underlying check implementation (e.g. `drivers_licence_dvs`)
- `outcome` - `pass` , `fail` , or `pending`
- `checked_after` / `checked_before` - inclusive bounds on `checked_at` (ISO 8601)

The top-level `archived` query parameter accepts `true` , `false` , or omit for both.

Sorting

`sort` accepts `checked_at` (default `-checked_at`), `outcome` , or `id_type` .

Query parameters

Field	Description
page	integer The page of results to return, starting at 1. **Example:** `1`

Field	Description
`per_page`	**integer** The number of records per page (defaults to 30, max 500). **Example:** `30`
`archived`	**boolean** `true` returns only archived records, `false` only non-archived records; omit it to return both.
`filter[program_uuid]`	**string (uuid)** Only records for entities in the given program.
`filter[entity_uuid]`	**string (uuid)** Only records for the given entity.
`filter[id_type]`	**string** Only ID checks of the given document type. **Example:** `drivers_licence`
`filter[check_type]`	**string** Only records of the given check type. The endpoint description lists the values. **Example:** `drivers_licence_dvs`
`filter[outcome]`	**string** Only records with the given outcome. The endpoint description lists the values. **Enum:** `pass`, `fail`, `pending`
`filter[checked_after]`	**string (date-time)** Inclusive lower bound on `checked_at` (ISO 8601).
`filter[checked_before]`	**string (date-time)** Inclusive upper bound on `checked_at` (ISO 8601).
`sort`	**string** The field to sort by, prefixed with `-` for descending order. The endpoint description lists the supported fields. **Example:** `-checked_at`

Responses

200 **Paginated ID check list** (application/json)

Field	Description

`data`	**array of objects**
`data[].` `uuid`	**string (uuid)**
`data[].` `check_number`	**string** **Example:** `I-A1B2CD`
`data[].` `id_type`	**string** **Example:** `drivers_licence`
`data[].` `id_type_name`	**string** **Example:** `Drivers Licence`
`data[].` `check_type`	**string** **Example:** `drivers_licence_dvs`
`data[].` `check_type_name`	**string** **Example:** `Drivers Licence (DVS)`
`data[].` `outcome`	**string** **Enum:** `pass`, `fail`, `pending`
`data[].` `outcome_summary`	**string or null**
`data[].` `data_summary`	**string or null**
`data[].` `checked_at`	**string (date-time) or null**
`data[].` `archived_at`	**string (date-time) or null** ISO 8601 timestamp at which the identification check was archived (null if not archived)
`data[].` `program`	**object or null**

`data[].`
`program.`
`uuid`
	string (uuid)

`data[].`
`program.`
`program_name`
	string

`data[].` `entity`	**object or null**

`data[].`
`entity.`
`uuid`
	string (uuid)

`data[].`
`entity.`
`entity_number`
	string

data[]. entity. entity_name	**string**
data[]. idpass	**object** Present only when `id_type` is `idpass`. A lightweight shell of the linked IDPass plus `idpass_url`, the path to the full IDPass resource. Fetch that resource for the complete configuration, status, document/validation summaries and image metadata.
data[]. idpass. uuid	**string (uuid)** The IDPass identifier. Use it with the show, image and PDF endpoints. **Example:** `5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
data[]. idpass. status	**string** Where the individual is in the hosted flow: `new` - the link has not been opened `opened` - the welcome page was passed `in_progress` - consent was given and documents are being submitted `complete` - the verification finished; read `verification_status` for the outcome `expired` - `link_validity_days` elapsed before completion `failed` - the verification could not be completed `cancelled` - cancelled in the portal The last four are terminal. See the IDPass section of the guide. **Enum:** `new`, `opened`, `in_progress`, `complete`, `expired`, `failed`, `cancelled` **Example:** `in_progress`
data[]. idpass. verification_status	**string or null** The outcome once `status` is terminal, null until then: `passed` - the identity was verified `review` - the identity was verified but flagged for review in the portal; treat it as not yet passed `failed` - the identity was not verified (including an expired or cancelled IDPass) **Enum:** `passed`, `review`, `failed` **Example:** `passed`
data[]. idpass. idpass_link	**string or null** The hosted link the individual opens to complete their verification. Treat it as opaque - pass it on as-is and do not parse or rebuild it. **Example:** `https://idpass.globaldata.net.au/idpass?token=abcd123456`
data[]. idpass. idpass_url	**string** Path to the full IDPass resource (GET /v1/idpasses/{uuid}). **Example:** `/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`

meta	**object**
meta. current_page	**integer**
meta. per_page	**integer**
meta. total	**integer**
meta. last_page	**integer**
api_reference	**string (uuid)**

Sample response


```
{
    "data": [
        {
            "uuid": "00000000-0000-0000-0000-000000000000",
            "check_number": "I-A1B2CD",
            "id_type": "drivers_licence",
            "id_type_name": "Drivers Licence",
            "check_type": "drivers_licence_dvs",
            "check_type_name": "Drivers Licence (DVS)",
            "outcome": "pass",
            "outcome_summary": "string",
            "data_summary": "string",
            "checked_at": "2024-01-01T00:00:00Z",
            "archived_at": "2024-01-01T00:00:00Z",
            "program": {
                "uuid": "00000000-0000-0000-0000-000000000000",
                "program_name": "string"
            },
            "entity": {
                "uuid": "00000000-0000-0000-0000-000000000000",
                "entity_number": "string",
                "entity_name": "string"
            },
            "idpass": {
                "uuid": "5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a",
                "status": "in_progress",
                "verification_status": "passed",
                "idpass_link": "https://idpass.globaldata.net.au/idpass?token=abcd123456",
                "idpass_url": "/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a"
            }
        }
    ],
    "meta": {
        "current_page": 0,
        "per_page": 0,
        "total": 0,
        "last_page": 0
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 401, 403, 429, 5XX (see Common error responses).

Show an ID check

GET `/id-checks/{id_check_uuid}`

Returns a single entity-bound ID check by UUID, including consent, verification details and the typed `details` block. Document numbers and other sensitive input fields are never returned - see the ID check field reference for per-type input shapes.

PDF version

A verification certificate PDF is available via download ID check PDF.

Path parameters

Field	Description
`id_check_uuid` REQUIRED	**string (uuid)** The uuid of the ID check.

Responses

200 **ID check response** (application/json)

Field	Description
`data`	**object** An entity-bound identification check (list envelope).
`data.` `uuid`	**string (uuid)**
`data.` `check_number`	**string** **Example:** `I-A1B2CD`
`data.` `id_type`	**string** **Example:** `drivers_licence`
`data.` `id_type_name`	**string** **Example:** `Drivers Licence`
`data.` `check_type`	**string** **Example:** `drivers_licence_dvs`
`data.` `check_type_name`	**string** **Example:** `Drivers Licence (DVS)`
`data.` `outcome`	**string** **Enum:** `pass`, `fail`, `pending`

Field	Description
data. outcome_summary	**string or null**
data. data_summary	**string or null**
data. checked_at	**string (date-time) or null**
data. archived_at	**string (date-time) or null** ISO 8601 timestamp at which the identification check was archived (null if not archived)
data. program	**object or null**
data. program. uuid	**string (uuid)**
data. program. program_name	**string**
data. entity	**object or null**
data. entity. uuid	**string (uuid)**
data. entity. entity_number	**string**
data. entity. entity_name	**string**
data. idpass	**object** Present only when `id_type` is `idpass`. A lightweight shell of the linked IDPass plus `idpass_url`, the path to the full IDPass resource. Fetch that resource for the complete configuration, status, document/validation summaries and image metadata.
data. idpass. uuid	**string (uuid)** The IDPass identifier. Use it with the show, image and PDF endpoints. **Example:** `5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`

Field	Description
data. idpass. status	**string** Where the individual is in the hosted flow: `new` - the link has not been opened `opened` - the welcome page was passed `in_progress` - consent was given and documents are being submitted `complete` - the verification finished; read `verification_status` for the outcome `expired` - `link_validity_days` elapsed before completion `failed` - the verification could not be completed `cancelled` - cancelled in the portal The last four are terminal. See the IDPass section of the guide. **Enum:** `new`, `opened`, `in_progress`, `complete`, `expired`, `failed`, `cancelled` **Example:** `in_progress`
data. idpass. verification_status	**string or null** The outcome once `status` is terminal, null until then: `passed` - the identity was verified `review` - the identity was verified but flagged for review in the portal; treat it as not yet passed `failed` - the identity was not verified (including an expired or cancelled IDPass) **Enum:** `passed`, `review`, `failed` **Example:** `passed`
data. idpass. idpass_link	**string or null** The hosted link the individual opens to complete their verification. Treat it as opaque - pass it on as-is and do not parse or rebuild it. **Example:** `https://idpass.globaldata.net.au/idpass?token=abcd123456`
data. idpass. idpass_url	**string** Path to the full IDPass resource (GET /v1/idpasses/{uuid}). **Example:** `/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
data. consent	**object (dynamic)**
data. verification_details	**array of objects**
data. verification_details[]. field	**string**
data. verification_details[]. value	**string**

Field	Description
data. details	one of 6 shapes

Field	Description

OBJECT 1

Field	Description
verification_result_code	string Enum: `Y`, `N`, `D`
originating_agency_code	string or null
verification_request_number	string or null
additional_information	array of strings

OBJECT 2

Field	Description
api_reference	string or null
message	string or null
match_results	object or null
match_results. first_name	string
match_results. last_name	string
match_results. dob	string Date of birth match result from the upstream provider.

OBJECT 3

Field	Description
api_reference	string or null
person_match	string or null

OBJECT 4

Field	Description
api_reference	string or null
reporting_reference	string or null Unique transaction identifier from the data source.
match_status	string or null Overall match result. `Match` indicates the supplied identity was fully verified. Enum: `Match`, `NoMatch`
document_verified	boolean Whether NZTA reports the supplied licence details as verified.
additional_information	string or null Explanatory message returned by the data source, typically alongside a `NoMatch` result.

Field	Description
api_reference	string (uuid)

Sample response


```
{
    "data": {
        "uuid": "00000000-0000-0000-0000-000000000000",
        "check_number": "I-A1B2CD",
        "id_type": "drivers_licence",
        "id_type_name": "Drivers Licence",
        "check_type": "drivers_licence_dvs",
        "check_type_name": "Drivers Licence (DVS)",
        "outcome": "pass",
        "outcome_summary": "string",
        "data_summary": "string",
        "checked_at": "2024-01-01T00:00:00Z",
        "archived_at": "2024-01-01T00:00:00Z",
        "program": {
            "uuid": "00000000-0000-0000-0000-000000000000",
            "program_name": "string"
        },
        "entity": {
            "uuid": "00000000-0000-0000-0000-000000000000",
            "entity_number": "string",
            "entity_name": "string"
        },
        "idpass": {
            "uuid": "5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a",
            "status": "in_progress",
            "verification_status": "passed",
            "idpass_link": "https://idpass.globaldata.net.au/idpass?token=abcd123456",
            "idpass_url": "/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a"
        },
        "verification_details": [
            {
                "field": "string",
                "value": "string"
            }
        ],
        "details": {
            "verification_result_code": "Y",
            "originating_agency_code": "string",
            "verification_request_number": "string",
            "additional_information": [
                "string"
            ]
        }
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 401, 403, 404, 429, 5XX (see Common error responses).

Download an ID check certificate as PDF

GET `/id-checks/{id_check_uuid}/pdf`

Returns a verification certificate PDF for a completed entity-bound ID check.

This endpoint does not bill - PDF generation reuses the existing on-record check data.

Status codes

- `200` - the PDF certificate is returned in the response body.
- `404` - either the ID check does not exist on the calling account, or the `id_type` is `idpass` (the idpass identity verification flow is hosted entirely on the idpass platform and does not produce a downloadable WatchEye certificate). Use the response `message` to distinguish the two cases.
- `409` - the ID check has not yet reached a terminal outcome (`outcome` is not `pass` or `fail`). Poll show ID check until `outcome` is set, then retry the download.

Path parameters

Field	Description
`id_check_uuid` REQUIRED	**string (uuid)** The uuid of the ID check.

Responses

200 **PDF certificate** (application/pdf)

Binary PDF file

409 **The ID check has not yet completed. Poll the show endpoint until `outcome` is `pass` or `fail` before retrying the download.** (application/json)

Field	Description
`message`	**string** **Example:** `The ID check has not completed yet. Wait for outcome to be 'pass' or 'fail' before downloading the certificate.`

Sample response

```
{
    "message": "The ID check has not completed yet. Wait for outcome to be 'pass' or 'fail' before downloading th
}
```

Standard error responses: 401, 403, 404, 429, 5XX (see Common error responses).

Launch an ID check against an entity

POST `/entities/{entity_uuid}/id-checks`

Runs a synchronous identification check against a saved entity and returns **200 OK** with the full completed check payload inline. Unlike screening checks, ID checks do not queue for background processing and do not expose a `status` field.

Each launch is charged at run time after a successful upstream verification call.

Consent

`consent_obtained` must be `true`. By sending this flag you assert that your system has obtained express, demonstrable consent from the individual being verified, and that the exact consent wording shown has been retained by your system for audit and compliance purposes. The specific wording and records required depend on the check types you run and the regulatory regime you operate under; for DVS-based check types, your consent must align with your account's DVS Agreement with the Australian Government. See the Consent section of the API guide for your full obligations and an illustrative starting point.

Idempotency

Accepts the `Idempotency-Key` header. See the Idempotency section.

Path parameters

Field	Description
`entity_uuid` REQUIRED	**string (uuid)** The uuid of the entity.

Header parameters

Field	Description
`Idempotency-Key`	**string [max 255 characters]** Optional client-generated key (typically a UUID) that lets you safely retry a write request without risk of duplicate work or duplicate billing. A retry with the same key and body returns the original response verbatim, with the `Idempotent-Replay: true` response header. Honoured on `POST` requests only. See the Idempotency section of the API guide for the full contract. **Example:** `8c4d3a18-2f63-4f9a-9f3a-9b1f5a7c2d4f`

Request body

Field	Description

`id_type` **REQUIRED**	**string** Identification type to run. See SELECTABLE_ID_TYPES in the portal field reference. **Example:** `drivers_licence`
`consent_obtained` **REQUIRED**	**boolean** Must be `true`. Asserts that your system has obtained express, demonstrable consent from the individual being verified before this check is launched, and that the exact wording shown has been retained by your system for audit and compliance purposes. The specific wording you must show and the records you must keep depend on which check types you run and the regulatory regime you operate under. For DVS-based check types your consent must align with your account's DVS Agreement with the Australian Government, which prescribes specific disclosure and record-keeping requirements. See the Consent section of the API guide for your full obligations and an illustrative starting point for the wording.
`data` **REQUIRED**	**object** Per-type input fields inside the launch request. Document numbers are never returned in responses. Date of birth must be supplied as `birth_date` in `YYYY-MM-DD` format when required for the selected `id_type`. Other date fields (`card_expiry`, `acquisition_date`, `event_date`, `registration_date`) also use `YYYY-MM-DD`; month-only fields (e.g. Medicare and ASIC/MSIC `card_expiry`) use `YYYY-MM`. See the per-id_type tables in the API guide for the exact fields and formats accepted by each check.
`data.` `birth_date`	**string (date)** Date of birth in `YYYY-MM-DD` format. **Example:** `1980-06-23`
`oac`	**string or null** The DVS OAC code to use for this check. Must be one of the OAC codes configured on your account. Required when your account is configured with more than one OAC. When your account has a single OAC this may be omitted and that OAC is used. **Example:** `ABC123`

Sample request

```
{
    "id_type": "drivers_licence",
    "consent_obtained": true,
    "data": {
        "birth_date": "1980-06-23"
    },
    "oac": "ABC123"
}
```

Responses

200 **ID check completed - full detail payload returned inline** (application/json)

Field	Description
`data`	**object** An entity-bound identification check (list envelope).
`data.` `uuid`	**string (uuid)**
`data.` `check_number`	**string** **Example:** `I-A1B2CD`
`data.` `id_type`	**string** **Example:** `drivers_licence`
`data.` `id_type_name`	**string** **Example:** `Drivers Licence`
`data.` `check_type`	**string** **Example:** `drivers_licence_dvs`
`data.` `check_type_name`	**string** **Example:** `Drivers Licence (DVS)`
`data.` `outcome`	**string** **Enum:** `pass`, `fail`, `pending`
`data.` `outcome_summary`	**string or null**
`data.` `data_summary`	**string or null**
`data.` `checked_at`	**string (date-time) or null**
`data.` `archived_at`	**string (date-time) or null** ISO 8601 timestamp at which the identification check was archived (null if not archived)
`data.` `program`	**object or null**

`data.`
`program.`
`uuid`
	string (uuid)

`data.`
`program.`
`program_name`
	string

`data.` `entity`	**object or null**

Field	Description
data. entity. uuid	**string (uuid)**
data. entity. entity_number	**string**
data. entity. entity_name	**string**
data. idpass	**object** Present only when `id_type` is `idpass`. A lightweight shell of the linked IDPass plus `idpass_url`, the path to the full IDPass resource. Fetch that resource for the complete configuration, status, document/validation summaries and image metadata.
data. idpass. uuid	**string (uuid)** The IDPass identifier. Use it with the show, image and PDF endpoints. **Example:** `5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
data. idpass. status	**string** Where the individual is in the hosted flow: `new` - the link has not been opened `opened` - the welcome page was passed `in_progress` - consent was given and documents are being submitted `complete` - the verification finished; read `verification_status` for the outcome `expired` - `link_validity_days` elapsed before completion `failed` - the verification could not be completed `cancelled` - cancelled in the portal The last four are terminal. See the IDPass section of the guide. **Enum:** `new`, `opened`, `in_progress`, `complete`, `expired`, `failed`, `cancelled` **Example:** `in_progress`
data. idpass. verification_status	**string or null** The outcome once `status` is terminal, null until then: `passed` - the identity was verified `review` - the identity was verified but flagged for review in the portal; treat it as not yet passed `failed` - the identity was not verified (including an expired or cancelled IDPass) **Enum:** `passed`, `review`, `failed` **Example:** `passed`

Field	Description
data. idpass. idpass_link	**string or null** The hosted link the individual opens to complete their verification. Treat it as opaque - pass it on as-is and do not parse or rebuild it. **Example:** `https://idpass.globaldata.net.au/idpass?token=abcd123456`
data. idpass. idpass_url	**string** Path to the full IDPass resource (GET /v1/idpasses/{uuid}). **Example:** `/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
data. consent	**object (dynamic)**
data. verification_details	**array of objects**
data. verification_details[]. field	**string**
data. verification_details[]. value	**string**
data. details	**one of 6 shapes**

Field	Description

OBJECT 1

Field	Description
verification_result_code	string Enum: `Y`, `N`, `D`
originating_agency_code	string or null
verification_request_number	string or null
additional_information	array of strings

OBJECT 2

Field	Description
api_reference	string or null
message	string or null
match_results	object or null
match_results. first_name	string
match_results. last_name	string
match_results. dob	string Date of birth match result from the upstream provider.

OBJECT 3

Field	Description
api_reference	string or null
person_match	string or null

OBJECT 4

Field	Description
api_reference	string or null
reporting_reference	string or null Unique transaction identifier from the data source.
match_status	string or null Overall match result. `Match` indicates the supplied identity was fully verified. Enum: `Match`, `NoMatch`
document_verified	boolean Whether NZTA reports the supplied licence details as verified.
additional_information	string or null Explanatory message returned by the data source, typically alongside a `NoMatch` result.

Field	Description
api_reference	string (uuid)

Sample response


```
{
    "data": {
        "uuid": "00000000-0000-0000-0000-000000000000",
        "check_number": "I-A1B2CD",
        "id_type": "drivers_licence",
        "id_type_name": "Drivers Licence",
        "check_type": "drivers_licence_dvs",
        "check_type_name": "Drivers Licence (DVS)",
        "outcome": "pass",
        "outcome_summary": "string",
        "data_summary": "string",
        "checked_at": "2024-01-01T00:00:00Z",
        "archived_at": "2024-01-01T00:00:00Z",
        "program": {
            "uuid": "00000000-0000-0000-0000-000000000000",
            "program_name": "string"
        },
        "entity": {
            "uuid": "00000000-0000-0000-0000-000000000000",
            "entity_number": "string",
            "entity_name": "string"
        },
        "idpass": {
            "uuid": "5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a",
            "status": "in_progress",
            "verification_status": "passed",
            "idpass_link": "https://idpass.globaldata.net.au/idpass?token=abcd123456",
            "idpass_url": "/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a"
        },
        "verification_details": [
            {
                "field": "string",
                "value": "string"
            }
        ],
        "details": {
            "verification_result_code": "Y",
            "originating_agency_code": "string",
            "verification_request_number": "string",
            "additional_information": [
                "string"
            ]
        }
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


422 **Idempotency-Key was reused with a different request body** (application/json)

Field	Description

`message`	**string**

Sample response

```
{
    "message": "string"
}
```

Standard error responses: 400, 401, 402, 403, 404, 429, 5XX (see Common error responses).

IDPass

Hosted identity verifications completed by the person themselves.

List IDPasses

GET `/idpasses`

Returns a paginated list of IDPasses on the calling account. Each IDPass is a remote identity verification that the individual completes via a hosted link; the result is updated asynchronously, so poll show IDPass until `status` is terminal.

Query parameters

Field	Description
`page`	**integer [min 1]** The page of results to return, starting at 1.
`per_page`	**integer [1..500]** The number of records per page (defaults to 30, max 500).
`filter[status]`	**string** Only records with the given status. The endpoint description lists the values. **Enum:** `new`, `opened`, `in_progress`, `complete`, `expired`, `failed`, `cancelled`
`filter[verification_status]`	**string** Only IDPasses with the given verification status. **Enum:** `passed`, `review`, `failed`
`filter[source]`	**string** Only IDPasses launched from the given source (the `source` field of the IDPass). **Enum:** `entity`, `adhoc`

Field	Description
`filter[entity_uuid]`	**string (uuid)** Only records for the given entity.
`filter[program_uuid]`	**string (uuid)** Only records for entities in the given program.
`filter[created_after]`	**string (date-time)** Inclusive lower bound on `created_at` (ISO 8601).
`filter[created_before]`	**string (date-time)** Inclusive upper bound on `created_at` (ISO 8601).
`sort`	**string** Sort field. Prefix with `-` for descending. Defaults to `-created_at`. **Enum:** `created_at`, `-created_at`, `status`, `-status`

Responses

200 **A paginated list of IDPasses** (application/json)

Field	Description
`data`	**array of objects**
`data[].` `uuid`	**string (uuid)** The IDPass identifier. Use it with the show, image and PDF endpoints. **Example:** `5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
`data[].` `source`	**string** Whether the IDPass is bound to a saved entity or was launched adhoc. **Enum:** `entity`, `adhoc` **Example:** `entity`

Field	Description
data[]. status	**string** Where the individual is in the hosted flow: `new` - the link has not been opened `opened` - the welcome page was passed `in_progress` - consent was given and documents are being submitted `complete` - the verification finished; read `verification_status` for the outcome `expired` - `link_validity_days` elapsed before completion `failed` - the verification could not be completed `cancelled` - cancelled in the portal The last four are terminal. **Enum:** `new`, `opened`, `in_progress`, `complete`, `expired`, `failed`, `cancelled` **Example:** `complete`
data[]. completed	**boolean** True once the IDPass has reached a terminal status. **Example:** `true`
data[]. verification_status	**string or null** The outcome once `status` is terminal, null until then: `passed` - the identity was verified `review` - the identity was verified but flagged for review in the portal; treat it as not yet passed `failed` - the identity was not verified (including an expired or cancelled IDPass) **Enum:** `passed`, `review`, `failed` **Example:** `passed`
data[]. requester_name	**string** Your organisation's name as shown to the individual on the consent page. It comes from the DVS identity (OAC) the IDPass was issued under. **Example:** `Sample Company Pty Ltd`
data[]. check_liveness	**boolean** Whether the individual was asked to complete a face liveness check. **Example:** `true`
data[]. require_id_photo	**boolean** Whether the individual was asked to supply a standalone ID photo. **Example:** `true`

Field	Description
data[]. document_allowed_types	**object** The allowed document types for each of the (up to three) verification steps.
data[]. document_allowed_types. document_1	**array of strings** The document types the individual could present for the first step: `licence` - Australian driver licence `passport` - Australian passport `medicare` - Medicare card `visa` - foreign passport with an Australian visa `centrelink` - Centrelink card (details typed in, no photo) `birth_certificate` - Australian birth certificate (details typed in, no photo) `nz_licence` - New Zealand driver licence **Enum:** `licence`, `passport`, `medicare`, `visa`, `centrelink`, `birth_certificate`, `nz_licence`
data[]. document_allowed_types. document_2	**array of strings** The document types for the second step, from the same list as `document_1`. Empty when there is no second step. **Enum:** `licence`, `passport`, `medicare`, `visa`, `centrelink`, `birth_certificate`, `nz_licence`
data[]. document_allowed_types. document_3	**array of strings** The document types for the third step, from the same list as `document_1`. Empty when there is no third step. **Enum:** `licence`, `passport`, `medicare`, `visa`, `centrelink`, `birth_certificate`, `nz_licence`
data[]. link_validity_days	**integer** How many days the hosted link stays open before the IDPass expires. **Example:** `7`
data[]. return_verification_images	**boolean** Whether verification images are retained and made available for download. **Example:** `true`

Field	Description
data[]. delivery_method	**string or null** How the link reached the individual: `sms` - WatchEye sent it to `delivery_phone` by text message `manual` - you delivered `idpass_link` yourself **Enum:** `manual`, `sms` **Example:** `sms`
data[]. idpass_link	**string or null** The hosted link the individual opens to complete their verification. Treat it as opaque - pass it on as-is and do not parse or rebuild it. **Example:** `https://idpass.globaldata.net.au/idpass?token=abcd123456`
data[]. sent_at	**string (date-time) or null** When the link was sent by SMS. Null when `delivery_method` is `manual`. **Example:** `2026-08-01T02:15:30+00:00`
data[]. created_at	**string (date-time) or null** When the IDPass was launched. **Example:** `2026-08-01T02:15:28+00:00`
data[]. detail_url	**string** Path to the full IDPass resource (GET /v1/idpasses/{uuid}). **Example:** `/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
meta	**object**
meta. current_page	**integer**
meta. per_page	**integer**
meta. total	**integer**
meta. last_page	**integer**
api_reference	**string (uuid)**

Sample response


```
{
    "data": [
        {
            "uuid": "5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a",
            "source": "entity",
            "status": "complete",
```

```
            "completed": true,
            "verification_status": "passed",
            "requester_name": "Sample Company Pty Ltd",
            "check_liveness": true,
            "require_id_photo": true,
            "document_allowed_types": {
                "document_1": [
                    "licence",
                    "passport"
                ],
                "document_2": [
                    "medicare"
                ],
                "document_3": []
            },
            "link_validity_days": 7,
            "return_verification_images": true,
            "delivery_method": "sms",
            "idpass_link": "https://idpass.globaldata.net.au/idpass?token=abcd123456",
            "sent_at": "2026-08-01T02:15:30+00:00",
            "created_at": "2026-08-01T02:15:28+00:00",
            "detail_url": "/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a"
        }
    ],
    "meta": {
        "current_page": 0,
        "per_page": 0,
        "total": 0,
        "last_page": 0
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 403, 429, 5XX (see Common error responses).

Show an IDPass

GET `/idpasses/{idpass_uuid}`

Returns a single IDPass by UUID with its full configuration, current status, verification outcome, document/validation summaries, activity log, available image metadata and a link back to the originating ID check.

Asynchronous result

An IDPass is completed by the individual via a hosted link, so the result arrives asynchronously. Poll this endpoint until `status` reaches a terminal value (`complete` , `expired` , `failed` or `cancelled`).

Images and PDF

The `images` array contains metadata only. Fetch image bytes via download IDPass image and the verification certificate via download IDPass PDF.

Path parameters

Field	Description
`idpass_uuid` REQUIRED	**string (uuid)** The uuid of the IDPass.

Responses

200 **IDPass response** (application/json)

Field	Description
`data`	**object** A remote identity verification (IDPass) - list envelope.
`data.` `uuid`	**string (uuid)** The IDPass identifier. Use it with the show, image and PDF endpoints. **Example:** `5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
`data.` `source`	**string** Whether the IDPass is bound to a saved entity or was launched adhoc. **Enum:** `entity`, `adhoc` **Example:** `entity`
`data.` `status`	**string** Where the individual is in the hosted flow: `new` - the link has not been opened `opened` - the welcome page was passed `in_progress` - consent was given and documents are being submitted `complete` - the verification finished; read `verification_status` for the outcome `expired` - `link_validity_days` elapsed before completion `failed` - the verification could not be completed `cancelled` - cancelled in the portal The last four are terminal. **Enum:** `new`, `opened`, `in_progress`, `complete`, `expired`, `failed`, `cancelled` **Example:** `complete`
`data.` `completed`	**boolean** True once the IDPass has reached a terminal status. **Example:** `true`

Field	Description
data. verification_status	**string or null** The outcome once `status` is terminal, null until then: `passed` - the identity was verified `review` - the identity was verified but flagged for review in the portal; treat it as not yet passed `failed` - the identity was not verified (including an expired or cancelled IDPass) **Enum:** `passed`, `review`, `failed` **Example:** `passed`
data. requester_name	**string** Your organisation's name as shown to the individual on the consent page. It comes from the DVS identity (OAC) the IDPass was issued under. **Example:** `Sample Company Pty Ltd`
data. check_liveness	**boolean** Whether the individual was asked to complete a face liveness check. **Example:** `true`
data. require_id_photo	**boolean** Whether the individual was asked to supply a standalone ID photo. **Example:** `true`
data. document_allowed_types	**object** The allowed document types for each of the (up to three) verification steps.
data. document_allowed_types. document_1	**array of strings** The document types the individual could present for the first step: `licence` - Australian driver licence `passport` - Australian passport `medicare` - Medicare card `visa` - foreign passport with an Australian visa `centrelink` - Centrelink card (details typed in, no photo) `birth_certificate` - Australian birth certificate (details typed in, no photo) `nz_licence` - New Zealand driver licence **Enum:** `licence`, `passport`, `medicare`, `visa`, `centrelink`, `birth_certificate`, `nz_licence`

Field	Description
data. document_allowed_types. document_2	**array of strings** The document types for the second step, from the same list as `document_1`. Empty when there is no second step. **Enum:** `licence`, `passport`, `medicare`, `visa`, `centrelink`, `birth_certificate`, `nz_licence`
data. document_allowed_types. document_3	**array of strings** The document types for the third step, from the same list as `document_1`. Empty when there is no third step. **Enum:** `licence`, `passport`, `medicare`, `visa`, `centrelink`, `birth_certificate`, `nz_licence`
data. link_validity_days	**integer** How many days the hosted link stays open before the IDPass expires. **Example:** `7`
data. return_verification_images	**boolean** Whether verification images are retained and made available for download. **Example:** `true`
data. delivery_method	**string or null** How the link reached the individual: `sms` - WatchEye sent it to `delivery_phone` by text message `manual` - you delivered `idpass_link` yourself **Enum:** `manual`, `sms` **Example:** `sms`
data. idpass_link	**string or null** The hosted link the individual opens to complete their verification. Treat it as opaque - pass it on as-is and do not parse or rebuild it. **Example:** `https://idpass.globaldata.net.au/idpass?token=abcd123456`
data. sent_at	**string (date-time) or null** When the link was sent by SMS. Null when `delivery_method` is `manual`. **Example:** `2026-08-01T02:15:30+00:00`
data. created_at	**string (date-time) or null** When the IDPass was launched. **Example:** `2026-08-01T02:15:28+00:00`

Field	Description
data. detail_url	**string** Path to the full IDPass resource (GET /v1/idpasses/{uuid}). **Example:** `/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
data. id_photo_purpose	**string or null** The purpose text shown to the individual when an ID photo is required, completing the sentence "This image will be used for ...". Null when no ID photo was required. **Example:** `the SampleCo Membership Card`
data. return_url	**string or null** The URL the individual is offered once the verification is complete, if one was configured. **Example:** `https://example.com/verification-complete`
data. privacy_policy_url	**string or null** The privacy policy linked from the consent page. It comes from the OAC configuration on your account. **Example:** `https://example.com/privacy`
data. dvs_oac	**string or null** The DVS identity (Originating Agency Code) the IDPass was issued under. **Example:** `ABC123`
data. delivery_phone	**string or null** The mobile number the link was sent to. Null when `delivery_method` is `manual`. **Example:** `0412345678`
data. verification_description	**string or null** A human-readable reason for `verification_status`, as reported by the verification platform. Null until the IDPass is terminal. **Example:** `The identity was verified successfully`

Field	Description
data. document_summary	**object (dynamic) or null** What happened with each document the individual presented, keyed `document_1`, `document_2` and `document_3` (plus `id_photo` when one was required). Empty until a document has been processed. Each document carries: `document_type` - one of the allowed document types. `ocr_status` and `ocr_attempts` - how the photo of the document was read (`running`, `failed` or `complete`; null for manually entered types). `validation_status` and `validation_attempts` - how the document was verified. `validation_result` - the verifier's answer. For DVS-verified documents this holds `verification_result_code` (`Y` valid, `N` not valid, `D` details not found), `verification_request_number`, `additional_information` (the DVS codes and messages behind an `N`) and `checked_at`. `biometric_result` - when a liveness check was performed and the document has a photo: `passed`, `similarity`, `threshold` and `face_occluded`. `ocr_data` - the fields read from the document, and `validation_data` - the fields actually submitted for verification, after any corrections the individual made.
data. validation_summary	**object (dynamic) or null** The verification outcome as a whole. Empty until the individual has finished. Contains: `liveness_result` - when a liveness check was performed: `validated`, `confidence` and `liveness_attempts`. `id_photo_result` - when an ID photo was required: `status`, `image_passed` and `biometric_passed`. `document_1_result`, `document_2_result`, `document_3_result` - per step: `document_type`, the `validated_fields` (name and date of birth) that were checked, `biographic_validated`, `biometric_validated`, `biometric_similarity`, `biometric_threshold` and any `changes` the individual made to the fields read from the document before verification. `verified_identity` - the name and date of birth established across the documents, or null when no consistent identity could be established. `requested_identity_mismatch` - true when the identity you supplied in `data` did not match the verified identity. Informational only; it does not fail the verification.
data. log	**array of strings or null** The activity log of the hosted flow, one entry per step the individual attempted. Useful for support when someone reports trouble completing the link.
data. consent_given_at	**string (date-time) or null** When the individual accepted the consent statement. Null until they do. **Example:** `2026-08-01T02:16:40+00:00`

Field	Description
data. images	**array of objects** Metadata for each available image (no binary payload). Fetch the bytes via download IDPass image. Images are only present when the IDPass was launched with `return_verification_images` set to `true`.
data. images[]. type	**string** The image identifier to pass to the image endpoint. Document images are named `{document_type}_{side}` (for example `licence_front`, `licence_back`, `passport_photo`); the liveness selfie is `probe_image` and the standalone ID photo is `id_photo`, each also available cropped to the face as `probe_image_cropped` and `id_photo_cropped`. **Example:** `licence_front`
data. images[]. title	**string** A display label for the image. **Example:** `Drivers Licence Front`
data. images[]. mime_type	**string** **Example:** `image/jpeg`
data. images[]. url	**string** Path to the image bytes (GET /v1/idpasses/{uuid}/images/{type}). **Example:** `/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a/images/licence_front`
data. id_check	**object or null** The originating ID check (entity-bound or adhoc) plus a link to it.
data. id_check. type	**string** Whether the ID check belongs to an entity or was adhoc. **Enum:** `entity`, `adhoc` **Example:** `entity`
data. id_check. uuid	**string (uuid)** **Example:** `8a0d2f4e-6b1c-4e9a-b3d7-2c5e1f6a9b0d`
data. id_check. check_number	**string** The ID check's reference number as shown in the portal. **Example:** `IDC-000123`
data. id_check. url	**string** Path to the ID check resource. **Example:** `/v1/id-checks/8a0d2f4e-6b1c-4e9a-b3d7-2c5e1f6a9b0d`

Field	Description
data. pdf_url	**string** Path to the verification certificate (GET /v1/idpasses/{uuid}/pdf), available once the IDPass is terminal. **Example:** `/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a/pdf`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a",
        "source": "entity",
        "status": "complete",
        "completed": true,
        "verification_status": "passed",
        "requester_name": "Sample Company Pty Ltd",
        "check_liveness": true,
        "require_id_photo": true,
        "document_allowed_types": {
            "document_1": [
                "licence",
                "passport"
            ],
            "document_2": [
                "medicare"
            ],
            "document_3": []
        },
        "link_validity_days": 7,
        "return_verification_images": true,
        "delivery_method": "sms",
        "idpass_link": "https://idpass.globaldata.net.au/idpass?token=abcd123456",
        "sent_at": "2026-08-01T02:15:30+00:00",
        "created_at": "2026-08-01T02:15:28+00:00",
        "detail_url": "/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a",
        "id_photo_purpose": "the SampleCo Membership Card",
        "return_url": "https://example.com/verification-complete",
        "privacy_policy_url": "https://example.com/privacy",
        "dvs_oac": "ABC123",
        "delivery_phone": "0412345678",
        "verification_description": "The identity was verified successfully",
        "document_summary": {
            "document_1": {
                "document_type": "licence",
                "ocr_status": "complete",
                "ocr_attempts": 1,
                "validation_status": "complete",
                "validation_attempts": 1,
                "validation_result": {
                    "strategy": "dvs",
                    "verification_result_code": "Y",
                    "verification_request_number": "7a3ed6e1-b707-4e2d-bacb-e2dfe8f57406",
```

```
                    "additional_information": [],
                    "checked_at": "2026-08-01 02:28:53"
                },
                "biometric_result": {
                    "passed": true,
                    "threshold": 80,
                    "similarity": 98.876,
                    "face_occluded": false
                },
                "ocr_data": {
                    "first_name": "John",
                    "last_name": "Smith",
                    "date_of_birth": "1980-06-23",
                    "licence_number": "123456789"
                },
                "validation_data": {
                    "first_name": "John",
                    "last_name": "Smith",
                    "date_of_birth": "1980-06-23",
                    "licence_number": "123456789"
                }
            }
        },
        "validation_summary": {
            "liveness_result": {
                "validated": true,
                "confidence": "99.811",
                "liveness_attempts": 1
            },
            "document_1_result": {
                "document_type": "licence",
                "validated_fields": {
                    "first_name": "John",
                    "last_name": "Smith",
                    "date_of_birth": "1980-06-23"
                },
                "biographic_validated": true,
                "biometric_validated": true,
                "biometric_similarity": "98.876",
                "biometric_threshold": 80,
                "changes": []
            },
            "verified_identity": {
                "first_name": "John",
                "middle_name": null,
                "last_name": "Smith",
                "date_of_birth": "1980-06-23"
            },
            "requested_identity_mismatch": false
        },
        "log": [
            "2026-08-01 02:16:02 Welcome page opened",
            "2026-08-01 02:16:40 Consent given",
            "2026-08-01 02:18:15 Document 1 (licence) captured",
            "2026-08-01 02:19:03 Liveness check passed"
        ],
        "consent_given_at": "2026-08-01T02:16:40+00:00",
        "images": [
            {
```

```
                "type": "licence_front",
                "title": "Drivers Licence Front",
                "mime_type": "image/jpeg",
                "url": "/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a/images/licence_front"
            }
        ],
        "id_check": {
            "type": "entity",
            "uuid": "8a0d2f4e-6b1c-4e9a-b3d7-2c5e1f6a9b0d",
            "check_number": "IDC-000123",
            "url": "/v1/id-checks/8a0d2f4e-6b1c-4e9a-b3d7-2c5e1f6a9b0d"
        },
        "pdf_url": "/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a/pdf"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 401, 403, 404, 429, 5XX (see Common error responses).

Launch an IDPass against an entity

POST `/entities/{entity_uuid}/idpasses`

Creates a remote identity verification (IDPass) for a saved individual entity and returns **202 Accepted** with the IDPass shell, including its hosted `idpass_link`. The entity must be an individual. `data.dob` is optional - when supplied it must match the entity's recorded date of birth, and when omitted the identity documents are still fully DVS-verified but the supplied details are matched against the documents on name only.

The verification itself is completed by the individual on the hosted platform, so the result is updated asynchronously. Poll show IDPass until `status` is terminal.

Each launch is charged at creation time. The product billed depends on the number of document verification steps configured (`document_1` / `document_2` / `document_3`).

Account requirements

Your account needs the IDPass product, and a DVS identity (OAC) with a privacy policy URL configured: the OAC supplies the `requester_name` and `privacy_policy_url` shown to the individual on the consent page. A launch that fails one of these returns `403` ; insufficient credit returns `402` ; an invalid `config` (for example a step that reuses a document type from an earlier step) returns `400` .

Delivery

When `config.delivery_method` is `sms` , the hosted link is sent to `config.delivery_phone` . When it is `manual` , no message is sent and you are responsible for delivering `idpass_link` to the individual.

Idempotency

Accepts the `Idempotency-Key` header. See the Idempotency section.

Path parameters

Field	Description
`entity_uuid` REQUIRED	**string (uuid)** The uuid of the entity.

Header parameters

Field	Description
`Idempotency-Key`	**string [max 255 characters]** Optional client-generated key (typically a UUID) that lets you safely retry a write request without risk of duplicate work or duplicate billing. A retry with the same key and body returns the original response verbatim, with the `Idempotent-Replay: true` response header. Honoured on `POST` requests only. See the Idempotency section of the API guide for the full contract. **Example:** `8c4d3a18-2f63-4f9a-9f3a-9b1f5a7c2d4f`

Request body

Field	Description
`data` REQUIRED	**object** The identity being verified. The name and date of birth the individual's documents establish are compared against these.
`data.` `first_name`	**string [max 50 characters]** Required unless `last_name` is supplied. **Example:** `John`
`data.` `middle_name`	**string [max 50 characters] or null** Optional. Does not satisfy the name requirement on its own. **Example:** `Andrew`
`data.` `last_name`	**string [max 50 characters]** Required unless `first_name` is supplied. **Example:** `Smith`

Field	Description
data. dob	**string (date) or null** Date of birth in `YYYY-MM-DD` format. Optional - when omitted, the identity documents are still fully DVS-verified, but the supplied details are matched against the documents on name only. For an entity-bound launch a supplied date of birth must match the entity's recorded date of birth. **Example:** `1980-06-23`
oac	**string or null** The DVS OAC code the IDPass should be issued under. Must be one of the OAC codes configured on your account. The selected OAC determines the requesting organisation name and privacy policy shown to the person being verified. Required when your account is configured with more than one OAC. When your account has a single OAC this may be omitted and that OAC is used. **Example:** `ABC123`
config REQUIRED	**object** How the hosted verification runs. The keys listed as required must all be supplied.
config. link_validity_days REQUIRED	**integer [1..14]** How many days the hosted link stays open, from 1 to 14. When it lapses the IDPass moves to `expired`, and the launch charge is refunded if the link was never used. **Example:** `7`
config. check_liveness REQUIRED	**boolean** Whether to perform a face liveness check. The selfie it captures is also compared biometrically against the photo on each document that has one. **Example:** `true`
config. document_1_allowed_types REQUIRED	**array of strings [min 1 items]** The document types the individual may present for the first verification step: `licence` - Australian driver licence `passport` - Australian passport `medicare` - Medicare card `visa` - foreign passport with an Australian visa `centrelink` - Centrelink card (details typed in, no photo) `birth_certificate` - Australian birth certificate (details typed in, no photo) `nz_licence` - New Zealand driver licence **Enum:** `licence`, `passport`, `medicare`, `visa`, `centrelink`, `birth_certificate`, `nz_licence`

Field	Description
config. document_2_allowed_types	**array of strings** Optional second verification step. Requires document 1 to be supplied, and a step cannot reuse a document type already consumed by an earlier step. The same document types as the first step: `licence` - Australian driver licence `passport` - Australian passport `medicare` - Medicare card `visa` - foreign passport with an Australian visa `centrelink` - Centrelink card (details typed in, no photo) `birth_certificate` - Australian birth certificate (details typed in, no photo) `nz_licence` - New Zealand driver licence **Enum:** `licence` , `passport` , `medicare` , `visa` , `centrelink` , `birth_certificate` , `nz_licence`
config. document_3_allowed_types	**array of strings** Optional third verification step. Requires document 2 to be supplied, and a step cannot reuse a document type already consumed by an earlier step. The same document types as the first step: `licence` - Australian driver licence `passport` - Australian passport `medicare` - Medicare card `visa` - foreign passport with an Australian visa `centrelink` - Centrelink card (details typed in, no photo) `birth_certificate` - Australian birth certificate (details typed in, no photo) `nz_licence` - New Zealand driver licence **Enum:** `licence` , `passport` , `medicare` , `visa` , `centrelink` , `birth_certificate` , `nz_licence`
config. require_id_photo REQUIRED	**boolean** Whether the individual must also supply a standalone ID photo, for example for a membership card. **Example:** `true`
config. id_photo_purpose	**string [3..255 characters] or null** Required when `require_id_photo` is `true` . Shown to the individual on the ID photo page to complete the sentence "This image will be used for ...", so write it to follow "for". **Example:** `the SampleCo Membership Card`

Field	Description
config. return_url	**string (uri) or null** A URL to offer the individual once the verification is complete, for example to return them to your application. **Example:** `https://example.com/verification-complete`
config. return_verification_images REQUIRED	**boolean** Whether the verification images should be retained and made available for download afterwards via the IDPass image endpoint. **Example:** `true`
config. delivery_method REQUIRED	**string** How the hosted link reaches the individual: `sms` - WatchEye sends the link to `delivery_phone` by text message `manual` - the link is only returned as `idpass_link` and you deliver it yourself In sandbox no text message is sent either way. **Enum:** `manual`, `sms` **Example:** `sms`
config. delivery_phone	**string or null** Australian mobile number (04XXXXXXXX). Required when `delivery_method` is `sms`, in which case the hosted link is sent to this number. **Example:** `0412345678`

Sample request

```
{
    "data": {
        "first_name": "John",
        "middle_name": "Andrew",
        "last_name": "Smith",
        "dob": "1980-06-23"
    },
    "oac": "ABC123",
    "config": {
        "link_validity_days": 7,
        "check_liveness": true,
        "document_1_allowed_types": [
            "licence",
            "passport"
        ],
        "document_2_allowed_types": [
            "medicare"
        ],
        "document_3_allowed_types": [
            "passport"
        ],
```

```
        "require_id_photo": true,
        "id_photo_purpose": "the SampleCo Membership Card",
        "return_url": "https://example.com/verification-complete",
        "return_verification_images": true,
        "delivery_method": "sms",
        "delivery_phone": "0412345678"
    }
}
```

Responses

202 **IDPass created - poll the show endpoint for the asynchronous result** (application/json)

Field	Description
`data`	**object** A newly launched IDPass: its identifier, configuration, delivery details and the hosted link, plus the ID check it belongs to. The verification outcome, document and validation summaries, log and images do not exist yet - poll show IDPass for them once the individual has completed the link.
`data.` `uuid`	**string (uuid)** The IDPass identifier. Use it with the show, image and PDF endpoints. **Example:** `5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
`data.` `source`	**string** Whether the IDPass is bound to a saved entity or was launched adhoc. **Enum:** `entity` , `adhoc` **Example:** `adhoc`
`data.` `requester_name`	**string** Your organisation's name as shown to the individual on the consent page. It comes from the DVS identity (OAC) the IDPass was issued under. **Example:** `Sample Company Pty Ltd`
`data.` `check_liveness`	**boolean** Whether the individual will be asked to complete a face liveness check. **Example:** `true`
`data.` `require_id_photo`	**boolean** Whether the individual will be asked to supply a standalone ID photo. **Example:** `false`
`data.` `document_allowed_types`	**object** The allowed document types for each of the (up to three) verification steps, as configured.

Field	Description
data. document_allowed_types. document_1	**array of strings** **Enum:** `licence` , `passport` , `medicare` , `visa` , `centrelink` , `birth_certificate` , `nz_licence`
data. document_allowed_types. document_2	**array of strings** **Enum:** `licence` , `passport` , `medicare` , `visa` , `centrelink` , `birth_certificate` , `nz_licence`
data. document_allowed_types. document_3	**array of strings** **Enum:** `licence` , `passport` , `medicare` , `visa` , `centrelink` , `birth_certificate` , `nz_licence`
data. link_validity_days	**integer** How many days the hosted link stays open before the IDPass expires. **Example:** `7`
data. return_verification_images	**boolean** Whether verification images will be retained and made available for download. **Example:** `true`
data. delivery_method	**string** How the link reaches the individual: `sms` - WatchEye has sent it to `delivery_phone` by text message `manual` - you deliver `idpass_link` yourself **Enum:** `manual` , `sms` **Example:** `sms`
data. idpass_link	**string** The hosted link the individual opens to complete their verification. Treat it as opaque - pass it on as-is and do not parse or rebuild it. **Example:** `https://idpass.globaldata.net.au/idpass?token=abcd123456`
data. sent_at	**string (date-time) or null** When the link was sent by SMS. Null when `delivery_method` is `manual` . **Example:** `2026-08-01T02:15:30+00:00`
data. created_at	**string (date-time)** When the IDPass was launched. **Example:** `2026-08-01T02:15:28+00:00`

Field	Description
data. detail_url	**string** Path to the full IDPass resource (GET /v1/idpasses/{uuid}). Poll it for the result. **Example:** `/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
data. id_check	**object or null** The ID check this IDPass was created under (entity-bound or adhoc) plus a link to it.
data. id_check. type	**string** Whether the ID check belongs to an entity or was adhoc. **Enum:** `entity`, `adhoc` **Example:** `adhoc`
data. id_check. uuid	**string (uuid)** **Example:** `8a0d2f4e-6b1c-4e9a-b3d7-2c5e1f6a9b0d`
data. id_check. check_number	**string** The ID check's reference number as shown in the portal. **Example:** `IDC-000123`
data. id_check. url	**string** Path to the ID check resource. **Example:** `/v1/adhoc-id-checks/8a0d2f4e-6b1c-4e9a-b3d7-2c5e1f6a9b0d`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a",
        "source": "adhoc",
        "requester_name": "Sample Company Pty Ltd",
        "check_liveness": true,
        "require_id_photo": false,
        "document_allowed_types": {
            "document_1": [
                "licence"
            ],
            "document_2": [
                "medicare"
            ],
            "document_3": []
        },
        "link_validity_days": 7,
        "return_verification_images": true,
        "delivery_method": "sms",
        "idpass_link": "https://idpass.globaldata.net.au/idpass?token=abcd123456",
        "sent_at": "2026-08-01T02:15:30+00:00",
```

```
        "created_at": "2026-08-01T02:15:28+00:00",
        "detail_url": "/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a",
        "id_check": {
            "type": "adhoc",
            "uuid": "8a0d2f4e-6b1c-4e9a-b3d7-2c5e1f6a9b0d",
            "check_number": "IDC-000123",
            "url": "/v1/adhoc-id-checks/8a0d2f4e-6b1c-4e9a-b3d7-2c5e1f6a9b0d"
        }
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


422 **Idempotency-Key was reused with a different request body** (application/json)

Field	Description
`message`	**string**

Sample response


```
{
    "message": "string"
}
```


Standard error responses: 400, 401, 402, 403, 404, 429, 5XX (see Common error responses).

Launch an adhoc IDPass

POST `/adhoc-idpasses`

Creates a remote identity verification (IDPass) that is not tied to a saved entity and returns **202 Accepted** with the IDPass shell, including its hosted `idpass_link`. Use this for one-off verifications where you do not need to persist the individual as an entity. `data.dob` is optional - when omitted, the identity documents are still fully DVS-verified but the supplied details are matched against the documents on name only.

The verification is completed by the individual on the hosted platform, so the result is updated asynchronously. Poll show IDPass until `status` is terminal.

Each launch is charged at creation time. The product billed depends on the number of document verification steps configured.

Account requirements

Your account needs the IDPass product, and a DVS identity (OAC) with a privacy policy URL configured: the OAC supplies the `requester_name` and `privacy_policy_url` shown to the individual on the consent page. A launch that fails one of these returns `403`; insufficient credit returns `402`; an invalid `config` (for example a step that reuses a document type from an earlier step) returns `400`.

Delivery

When `config.delivery_method` is `sms`, the hosted link is sent to `config.delivery_phone`. When it is `manual`, you deliver `idpass_link` to the individual yourself.

Idempotency

Accepts the `Idempotency-Key` header. See the Idempotency section.

Header parameters

Field	Description
`Idempotency-Key`	**string [max 255 characters]** Optional client-generated key (typically a UUID) that lets you safely retry a write request without risk of duplicate work or duplicate billing. A retry with the same key and body returns the original response verbatim, with the `Idempotent-Replay: true` response header. Honoured on `POST` requests only. See the Idempotency section of the API guide for the full contract. **Example:** `8c4d3a18-2f63-4f9a-9f3a-9b1f5a7c2d4f`

Request body

Field	Description
`data` REQUIRED	**object** The identity being verified. The name and date of birth the individual's documents establish are compared against these.
`data.` `first_name`	**string [max 50 characters]** Required unless `last_name` is supplied. **Example:** `John`
`data.` `middle_name`	**string [max 50 characters] or null** Optional. Does not satisfy the name requirement on its own. **Example:** `Andrew`
`data.` `last_name`	**string [max 50 characters]** Required unless `first_name` is supplied. **Example:** `Smith`

Field	Description
data. dob	**string (date) or null** Date of birth in `YYYY-MM-DD` format. Optional - when omitted, the identity documents are still fully DVS-verified, but the supplied details are matched against the documents on name only. For an entity-bound launch a supplied date of birth must match the entity's recorded date of birth. **Example:** `1980-06-23`
oac	**string or null** The DVS OAC code the IDPass should be issued under. Must be one of the OAC codes configured on your account. The selected OAC determines the requesting organisation name and privacy policy shown to the person being verified. Required when your account is configured with more than one OAC. When your account has a single OAC this may be omitted and that OAC is used. **Example:** `ABC123`
config REQUIRED	**object** How the hosted verification runs. The keys listed as required must all be supplied.
config. link_validity_days REQUIRED	**integer [1..14]** How many days the hosted link stays open, from 1 to 14. When it lapses the IDPass moves to `expired`, and the launch charge is refunded if the link was never used. **Example:** `7`
config. check_liveness REQUIRED	**boolean** Whether to perform a face liveness check. The selfie it captures is also compared biometrically against the photo on each document that has one. **Example:** `true`
config. document_1_allowed_types REQUIRED	**array of strings [min 1 items]** The document types the individual may present for the first verification step: `licence` - Australian driver licence `passport` - Australian passport `medicare` - Medicare card `visa` - foreign passport with an Australian visa `centrelink` - Centrelink card (details typed in, no photo) `birth_certificate` - Australian birth certificate (details typed in, no photo) `nz_licence` - New Zealand driver licence **Enum:** `licence`, `passport`, `medicare`, `visa`, `centrelink`, `birth_certificate`, `nz_licence`

Field	Description
config. document_2_allowed_types	**array of strings** Optional second verification step. Requires document 1 to be supplied, and a step cannot reuse a document type already consumed by an earlier step. The same document types as the first step: `licence` - Australian driver licence `passport` - Australian passport `medicare` - Medicare card `visa` - foreign passport with an Australian visa `centrelink` - Centrelink card (details typed in, no photo) `birth_certificate` - Australian birth certificate (details typed in, no photo) `nz_licence` - New Zealand driver licence **Enum:** `licence` , `passport` , `medicare` , `visa` , `centrelink` , `birth_certificate` , `nz_licence`
config. document_3_allowed_types	**array of strings** Optional third verification step. Requires document 2 to be supplied, and a step cannot reuse a document type already consumed by an earlier step. The same document types as the first step: `licence` - Australian driver licence `passport` - Australian passport `medicare` - Medicare card `visa` - foreign passport with an Australian visa `centrelink` - Centrelink card (details typed in, no photo) `birth_certificate` - Australian birth certificate (details typed in, no photo) `nz_licence` - New Zealand driver licence **Enum:** `licence` , `passport` , `medicare` , `visa` , `centrelink` , `birth_certificate` , `nz_licence`
config. require_id_photo REQUIRED	**boolean** Whether the individual must also supply a standalone ID photo, for example for a membership card. **Example:** `true`
config. id_photo_purpose	**string [3..255 characters] or null** Required when `require_id_photo` is `true` . Shown to the individual on the ID photo page to complete the sentence "This image will be used for ...", so write it to follow "for". **Example:** `the SampleCo Membership Card`

Field	Description
config. return_url	**string (uri) or null** A URL to offer the individual once the verification is complete, for example to return them to your application. **Example:** `https://example.com/verification-complete`
config. return_verification_images REQUIRED	**boolean** Whether the verification images should be retained and made available for download afterwards via the IDPass image endpoint. **Example:** `true`
config. delivery_method REQUIRED	**string** How the hosted link reaches the individual: `sms` - WatchEye sends the link to `delivery_phone` by text message `manual` - the link is only returned as `idpass_link` and you deliver it yourself In sandbox no text message is sent either way. **Enum:** `manual`, `sms` **Example:** `sms`
config. delivery_phone	**string or null** Australian mobile number (04XXXXXXXX). Required when `delivery_method` is `sms`, in which case the hosted link is sent to this number. **Example:** `0412345678`

Sample request

```
{
    "data": {
        "first_name": "John",
        "middle_name": "Andrew",
        "last_name": "Smith",
        "dob": "1980-06-23"
    },
    "oac": "ABC123",
    "config": {
        "link_validity_days": 7,
        "check_liveness": true,
        "document_1_allowed_types": [
            "licence",
            "passport"
        ],
        "document_2_allowed_types": [
            "medicare"
        ],
        "document_3_allowed_types": [
            "passport"
        ],
```

```
        "require_id_photo": true,
        "id_photo_purpose": "the SampleCo Membership Card",
        "return_url": "https://example.com/verification-complete",
        "return_verification_images": true,
        "delivery_method": "sms",
        "delivery_phone": "0412345678"
    }
}
```

Responses

202 **IDPass created - poll the show endpoint for the asynchronous result** (application/json)

Field	Description
`data`	**object** A newly launched IDPass: its identifier, configuration, delivery details and the hosted link, plus the ID check it belongs to. The verification outcome, document and validation summaries, log and images do not exist yet - poll show IDPass for them once the individual has completed the link.
`data.` `uuid`	**string (uuid)** The IDPass identifier. Use it with the show, image and PDF endpoints. **Example:** `5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
`data.` `source`	**string** Whether the IDPass is bound to a saved entity or was launched adhoc. **Enum:** `entity` , `adhoc` **Example:** `adhoc`
`data.` `requester_name`	**string** Your organisation's name as shown to the individual on the consent page. It comes from the DVS identity (OAC) the IDPass was issued under. **Example:** `Sample Company Pty Ltd`
`data.` `check_liveness`	**boolean** Whether the individual will be asked to complete a face liveness check. **Example:** `true`
`data.` `require_id_photo`	**boolean** Whether the individual will be asked to supply a standalone ID photo. **Example:** `false`
`data.` `document_allowed_types`	**object** The allowed document types for each of the (up to three) verification steps, as configured.

Field	Description
data. document_allowed_types. document_1	**array of strings** **Enum:** `licence` , `passport` , `medicare` , `visa` , `centrelink` , `birth_certificate` , `nz_licence`
data. document_allowed_types. document_2	**array of strings** **Enum:** `licence` , `passport` , `medicare` , `visa` , `centrelink` , `birth_certificate` , `nz_licence`
data. document_allowed_types. document_3	**array of strings** **Enum:** `licence` , `passport` , `medicare` , `visa` , `centrelink` , `birth_certificate` , `nz_licence`
data. link_validity_days	**integer** How many days the hosted link stays open before the IDPass expires. **Example:** `7`
data. return_verification_images	**boolean** Whether verification images will be retained and made available for download. **Example:** `true`
data. delivery_method	**string** How the link reaches the individual: `sms` - WatchEye has sent it to `delivery_phone` by text message `manual` - you deliver `idpass_link` yourself **Enum:** `manual` , `sms` **Example:** `sms`
data. idpass_link	**string** The hosted link the individual opens to complete their verification. Treat it as opaque - pass it on as-is and do not parse or rebuild it. **Example:** `https://idpass.globaldata.net.au/idpass?token=abcd123456`
data. sent_at	**string (date-time) or null** When the link was sent by SMS. Null when `delivery_method` is `manual` . **Example:** `2026-08-01T02:15:30+00:00`
data. created_at	**string (date-time)** When the IDPass was launched. **Example:** `2026-08-01T02:15:28+00:00`

Field	Description
data. detail_url	**string** Path to the full IDPass resource (GET /v1/idpasses/{uuid}). Poll it for the result. **Example:** `/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a`
data. id_check	**object or null** The ID check this IDPass was created under (entity-bound or adhoc) plus a link to it.
data. id_check. type	**string** Whether the ID check belongs to an entity or was adhoc. **Enum:** `entity`, `adhoc` **Example:** `adhoc`
data. id_check. uuid	**string (uuid)** **Example:** `8a0d2f4e-6b1c-4e9a-b3d7-2c5e1f6a9b0d`
data. id_check. check_number	**string** The ID check's reference number as shown in the portal. **Example:** `IDC-000123`
data. id_check. url	**string** Path to the ID check resource. **Example:** `/v1/adhoc-id-checks/8a0d2f4e-6b1c-4e9a-b3d7-2c5e1f6a9b0d`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a",
        "source": "adhoc",
        "requester_name": "Sample Company Pty Ltd",
        "check_liveness": true,
        "require_id_photo": false,
        "document_allowed_types": {
            "document_1": [
                "licence"
            ],
            "document_2": [
                "medicare"
            ],
            "document_3": []
        },
        "link_validity_days": 7,
        "return_verification_images": true,
        "delivery_method": "sms",
        "idpass_link": "https://idpass.globaldata.net.au/idpass?token=abcd123456",
        "sent_at": "2026-08-01T02:15:30+00:00",
```

```
        "created_at": "2026-08-01T02:15:28+00:00",
        "detail_url": "/v1/idpasses/5b1c7c8e-0b6e-4f0a-9b3a-1f2e3d4c5b6a",
        "id_check": {
            "type": "adhoc",
            "uuid": "8a0d2f4e-6b1c-4e9a-b3d7-2c5e1f6a9b0d",
            "check_number": "IDC-000123",
            "url": "/v1/adhoc-id-checks/8a0d2f4e-6b1c-4e9a-b3d7-2c5e1f6a9b0d"
        }
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


422 **Idempotency-Key was reused with a different request body** (application/json)

Field	Description
`message`	**string**

Sample response


```
{
    "message": "string"
}
```


Standard error responses: 400, 401, 402, 403, 429, 5XX (see Common error responses).

Download an IDPass image

GET `/idpasses/{idpass_uuid}/images/{image_type}`

Returns a single IDPass image as its raw bytes, mirroring the download IDPass PDF endpoint. The `Content-Type` is the image's stored MIME type (typically `image/jpeg`) and a `Content-Disposition` header suggests an inline filename. The `image_type` is one of the file types listed in the `images` array of the show IDPass response (e.g. `licence_front`, `id_photo`, `probe_image`); that array is also where the per-image `title` and `mime_type` metadata live.

Images are only retained when the IDPass was launched with `return_verification_images` set to `true`. Unknown or unavailable types return `404`.

Path parameters

Field	Description
`idpass_uuid` REQUIRED	**string (uuid)** The uuid of the IDPass.

Field	Description
`image_type` REQUIRED	**string** The image file type, as listed in the `images` array of the show response. **Example:** `id_photo`

Responses

200 **The requested image, returned as raw bytes** (image/jpeg, image/png)

Standard error responses: 401, 403, 404, 429, 5XX (see Common error responses).

Download an IDPass certificate as PDF

GET `/idpasses/{idpass_uuid}/pdf`

Returns the verification certificate PDF for a completed IDPass.

This endpoint does not bill - the certificate is generated from the on-record IDPass data.

Status codes

- `200` - the PDF certificate is returned in the response body.
- `404` - the IDPass does not exist on the calling account.
- `409` - the IDPass has not yet reached a terminal status. Poll show IDPass until `status` is terminal, then retry the download.

Path parameters

Field	Description
`idpass_uuid` REQUIRED	**string (uuid)** The uuid of the IDPass.

Responses

200 **PDF certificate** (application/pdf)

Binary PDF file

409 **The IDPass has not yet completed. Poll the show endpoint until `status` is terminal before retrying the download.** (application/json)

Field	Description

`message`	**string** **Example:** `The IDPass has not completed yet. Wait for a terminal status before downloading the certificate.`

Sample response

```
{
    "message": "The IDPass has not completed yet. Wait for a terminal status before downloading the certificate."
}
```

Standard error responses: 401, 403, 404, 429, 5XX (see Common error responses).

Monitors

Scheduled screening configured per program.

List monitors in a program

GET `/programs/{program_uuid}/monitors`

Returns the monitors that belong to the given program. Soft-deleted monitors are excluded. Results are paginated.

Filters

The following filters can be applied via the `filter[...]` query parameters:

- `status` - one of `active`, `paused`
- `process_status` - one of `pending`, `processing`, `failed`, `paused`
- `monitor_check_type` - one of the operation types enabled on the account
- `schedule` - one of the schedule values listed on the Monitor schema

Sorting

The `sort` query parameter accepts `created_at`, `last_run_at` (default: `-last_run_at`, most recently run first - monitors that have never run sort last) or `monitor_name`. Prefix with `-` for descending order.

Path parameters

Field	Description
`program_uuid` REQUIRED	**string (uuid)** The program UUID

Query parameters

Field	Description
`page`	**integer** The page number to return **Example:** `1`
`per_page`	**integer** The number of monitors per page (defaults to 30, max 500) **Example:** `30`
`filter[status]`	**string** Only records with the given status. The endpoint description lists the values. **Enum:** `active`, `paused`
`filter[process_status]`	**string** Only monitors with the given process status. **Enum:** `pending`, `processing`, `failed`, `paused`
`filter[monitor_check_type]`	**string** Only monitors of the given operation type.
`filter[schedule]`	**string** Only monitors on the given schedule.
`sort`	**string** Field to sort by; prefix with `-` for descending order **Enum:** `created_at`, `-created_at`, `last_run_at`, `-last_run_at`, `monitor_name`, `-monitor_name` **Example:** `-last_run_at`

Responses

200 **Monitors list response** (application/json)

Field	Description
`data`	**array of objects** An array of monitors
`data[].` `uuid`	**string (uuid)** The monitor's UUID **Example:** `5d6c8f93-91d4-4d6a-bb5e-2c64e9a3b1f0`

Field	Description
data[]. monitor_number	**string** A short obfuscated public identifier for the monitor (e.g. "M-12-34-56") **Example:** `M-12-34-56`
data[]. monitor_name	**string** The display name of the monitor **Example:** `Weekly PEP Sweep`
data[]. monitor_check_type	**string** The screening operation this monitor runs. Cannot be changed after creation - create a new monitor instead. The list of allowed values depends on the products enabled on the calling account. **Example:** `pep_sanction_check`
data[]. monitor_check_type_name	**string** Human-readable name for `monitor_check_type` (e.g. "PEP & Sanction Check") **Example:** `PEP & Sanction Check`
data[]. status	**string** Lifecycle status: `active` - the monitor runs on its schedule and can be triggered manually `paused` - the monitor does not run on its schedule and cannot be triggered manually **Enum:** `active`, `paused` **Example:** `active`
data[]. process_status	**string or null** In-flight run state, or `null` when the monitor is idle: `pending` - queued `processing` - currently running `failed` - the most recent run failed (cleared on next successful run) `paused` - the run was paused mid-flight (e.g. due to insufficient credit) Callers polling after `POST /monitors/{uuid}/run` should wait for this field to transition out of `pending` / `processing` before reading results. **Enum:** `pending`, `processing`, `failed`, `paused`
data[]. process_status_comment	**string or null** Optional human-readable detail accompanying `process_status` (e.g. failure message)

Field	Description
data[]. schedule	**string** How often the monitor runs unattended. `manual` means it never runs on its own and must be triggered via the run endpoint. **Enum:** `manual`, `every_day`, `weekdays`, `weekends`, `monday`, `tuesday`, `wednesday`, `thursday`, `friday`, `saturday`, `sunday`, `monthly_1`, `monthly_15`, `monthly_1_15`, `quarterly` **Example:** `weekdays`
data[]. schedule_name	**string or null** Human-readable description of the schedule (e.g. "Every Day (Mon-Sun)") **Example:** `Weekdays (Mon-Fri)`
data[]. config	**object (dynamic)** Per-operation configuration. The shape depends on `monitor_check_type`; see the Monitor Configuration section of the API guide for the schema of each operation. The portal's monitor create/edit screens render the same fields you'd send here.
data[]. last_run_at	**string (date-time) or null** ISO 8601 timestamp of the most recent run, or `null` if the monitor has never run **Example:** `2025-01-15T03:00:00Z`
data[]. has_run_before	**boolean** `true` if the monitor has processed at least one entity. Used by the "new" run mode to know whether there is a cursor to resume from. **Example:** `true`
data[]. program_uuid	**string (uuid)** UUID of the program this monitor belongs to **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data[]. created_at	**string (date-time)** ISO 8601 timestamp at which the monitor was created **Example:** `2025-01-01T00:00:00Z`
data[]. updated_at	**string (date-time)** ISO 8601 timestamp at which the monitor was last updated **Example:** `2025-01-01T00:00:00Z`
meta	**object**
meta. current_page	**integer** **Example:** `1`

Field	Description
meta. per_page	integer Example: 30
meta. total	integer Example: 1
meta. last_page	integer Example: 1
api_reference	string (uuid) A unique identifier for this request. This can be used to track the request in the logs and audit trail.

Sample response


```
{
    "data": [
        {
            "uuid": "5d6c8f93-91d4-4d6a-bb5e-2c64e9a3b1f0",
            "monitor_number": "M-12-34-56",
            "monitor_name": "Weekly PEP Sweep",
            "monitor_check_type": "pep_sanction_check",
            "monitor_check_type_name": "PEP & Sanction Check",
            "status": "active",
            "process_status": null,
            "process_status_comment": null,
            "schedule": "weekdays",
            "schedule_name": "Weekdays (Mon-Fri)",
            "config": {
                "search_type": "pep_and_sanction",
                "similarity_threshold": 90
            },
            "last_run_at": "2025-01-15T03:00:00Z",
            "has_run_before": true,
            "program_uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
            "created_at": "2025-01-01T00:00:00Z",
            "updated_at": "2025-01-01T00:00:00Z"
        }
    ],
    "meta": {
        "current_page": 1,
        "per_page": 30,
        "total": 1,
        "last_page": 1
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 402, 403, 404, 429, 5XX (see Common error responses).

Create a monitor

POST `/programs/{program_uuid}/monitors`

Create a new monitor in the given program. The `monitor_check_type` is fixed at creation - to change a monitor's operation type, delete it and create a new one.

Configuration

The `config` object is per-operation. Its shape is whatever the chosen `monitor_check_type` expects. The full list of configuration keys, allowed values and defaults for every operation type is documented in the Monitor Configuration section of the API guide. Any required config keys missing from your request will produce a `400` with the relevant validation errors.

Idempotency

This endpoint accepts the `Idempotency-Key` header so that retries on network failure are safe. See the Idempotency section of the API guide for the full contract.

Path parameters

Field	Description
`program_uuid` REQUIRED	**string (uuid)** The program UUID

Header parameters

Field	Description
`Idempotency-Key`	**string [max 255 characters]** Optional client-generated key (typically a UUID) that lets you safely retry a write request without risk of duplicate work or duplicate billing. A retry with the same key and body returns the original response verbatim, with the `Idempotent-Replay: true` response header. Honoured on `POST` requests only. See the Idempotency section of the API guide for the full contract. **Example:** `8c4d3a18-2f63-4f9a-9f3a-9b1f5a7c2d4f`

Request body

The monitor details

Field	Description

`monitor_name` REQUIRED	**string [3..255 characters]** **Example:** `Weekly PEP Sweep`
`monitor_check_type` REQUIRED	**string** The screening operation to run. Must be one of the operation types enabled on the calling account. **Example:** `pep_sanction_check`
`status` REQUIRED	**string** **Enum:** `active`, `paused` **Example:** `active`
`schedule` REQUIRED	**string** **Enum:** `manual`, `every_day`, `weekdays`, `weekends`, `monday`, `tuesday`, `wednesday`, `thursday`, `friday`, `saturday`, `sunday`, `monthly_1`, `monthly_15`, `monthly_1_15`, `quarterly` **Example:** `weekdays`
`config`	**object (dynamic)** Per-operation configuration; see the Configuration section above.

Sample request

```
{
    "monitor_name": "Weekly PEP Sweep",
    "monitor_check_type": "pep_sanction_check",
    "status": "active",
    "schedule": "weekdays",
    "config": {
        "search_type": "pep_and_sanction",
        "similarity_threshold": 90
    }
}
```

Responses

201 **Monitor created** (application/json)

Field	Description
`data`	**object** A monitor is a scheduled (or manual) screening job that runs a single screening operation (PEP/sanction, ASIC, address verification, etc.) against the entities in its parent program. The operation type is fixed at creation; only configuration, name, status and schedule can be changed after that.

Field	Description
data. uuid	**string (uuid)** The monitor's UUID **Example:** `5d6c8f93-91d4-4d6a-bb5e-2c64e9a3b1f0`
data. monitor_number	**string** A short obfuscated public identifier for the monitor (e.g. "M-12-34-56") **Example:** `M-12-34-56`
data. monitor_name	**string** The display name of the monitor **Example:** `Weekly PEP Sweep`
data. monitor_check_type	**string** The screening operation this monitor runs. Cannot be changed after creation - create a new monitor instead. The list of allowed values depends on the products enabled on the calling account. **Example:** `pep_sanction_check`
data. monitor_check_type_name	**string** Human-readable name for `monitor_check_type` (e.g. "PEP & Sanction Check") **Example:** `PEP & Sanction Check`
data. status	**string** Lifecycle status: `active` - the monitor runs on its schedule and can be triggered manually `paused` - the monitor does not run on its schedule and cannot be triggered manually **Enum:** `active`, `paused` **Example:** `active`
data. process_status	**string or null** In-flight run state, or `null` when the monitor is idle: `pending` - queued `processing` - currently running `failed` - the most recent run failed (cleared on next successful run) `paused` - the run was paused mid-flight (e.g. due to insufficient credit) Callers polling after `POST /monitors/{uuid}/run` should wait for this field to transition out of `pending` / `processing` before reading results. **Enum:** `pending`, `processing`, `failed`, `paused`
data. process_status_comment	**string or null** Optional human-readable detail accompanying `process_status` (e.g. failure message)

Field	Description
data. schedule	**string** How often the monitor runs unattended. `manual` means it never runs on its own and must be triggered via the run endpoint. **Enum:** `manual`, `every_day`, `weekdays`, `weekends`, `monday`, `tuesday`, `wednesday`, `thursday`, `friday`, `saturday`, `sunday`, `monthly_1`, `monthly_15`, `monthly_1_15`, `quarterly` **Example:** `weekdays`
data. schedule_name	**string or null** Human-readable description of the schedule (e.g. "Every Day (Mon-Sun)") **Example:** `Weekdays (Mon-Fri)`
data. config	**object (dynamic)** Per-operation configuration. The shape depends on `monitor_check_type`; see the Monitor Configuration section of the API guide for the schema of each operation. The portal's monitor create/edit screens render the same fields you'd send here.
data. last_run_at	**string (date-time) or null** ISO 8601 timestamp of the most recent run, or `null` if the monitor has never run **Example:** `2025-01-15T03:00:00Z`
data. has_run_before	**boolean** `true` if the monitor has processed at least one entity. Used by the "new" run mode to know whether there is a cursor to resume from. **Example:** `true`
data. program_uuid	**string (uuid)** UUID of the program this monitor belongs to **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. created_at	**string (date-time)** ISO 8601 timestamp at which the monitor was created **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** ISO 8601 timestamp at which the monitor was last updated **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response

```
{
    "data": {
        "uuid": "5d6c8f93-91d4-4d6a-bb5e-2c64e9a3b1f0",
        "monitor_number": "M-12-34-56",
        "monitor_name": "Weekly PEP Sweep",
        "monitor_check_type": "pep_sanction_check",
        "monitor_check_type_name": "PEP & Sanction Check",
        "status": "active",
        "process_status": null,
        "process_status_comment": null,
        "schedule": "weekdays",
        "schedule_name": "Weekdays (Mon-Fri)",
        "config": {
            "search_type": "pep_and_sanction",
            "similarity_threshold": 90
        },
        "last_run_at": "2025-01-15T03:00:00Z",
        "has_run_before": true,
        "program_uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


409 **Conflict - a request with this Idempotency-Key is currently being processed** (application/json)

Field	Description
message	string **Example:** `A request with this Idempotency-Key is already being processed.`

Sample response


```
{
    "message": "A request with this Idempotency-Key is already being processed."
}
```


422 **Idempotency-Key was reused with a different request body** (application/json)

Field	Description
message	string **Example:** `Idempotency-Key was reused with a different request body.`

Sample response


```
{
    "message": "Idempotency-Key was reused with a different request body."
}
```

Standard error responses: 400, 401, 402, 403, 404, 429, 5XX (see Common error responses).

Show a monitor

GET `/monitors/{monitor_uuid}`

Returns a single monitor by its UUID. Unknown UUIDs return `404`.

Path parameters

Field	Description
`monitor_uuid` REQUIRED	**string (uuid)** The monitor UUID

Responses

200 **Monitor response** (application/json)

Field	Description
`data`	**object** A monitor is a scheduled (or manual) screening job that runs a single screening operation (PEP/sanction, ASIC, address verification, etc.) against the entities in its parent program. The operation type is fixed at creation; only configuration, name, status and schedule can be changed after that.
`data.` `uuid`	**string (uuid)** The monitor's UUID **Example:** `5d6c8f93-91d4-4d6a-bb5e-2c64e9a3b1f0`
`data.` `monitor_number`	**string** A short obfuscated public identifier for the monitor (e.g. "M-12-34-56") **Example:** `M-12-34-56`
`data.` `monitor_name`	**string** The display name of the monitor **Example:** `Weekly PEP Sweep`
`data.` `monitor_check_type`	**string** The screening operation this monitor runs. Cannot be changed after creation - create a new monitor instead. The list of allowed values depends on the products enabled on the calling account. **Example:** `pep_sanction_check`

Field	Description
data. monitor_check_type_name	**string** Human-readable name for `monitor_check_type` (e.g. "PEP & Sanction Check") **Example:** `PEP & Sanction Check`
data. status	**string** Lifecycle status: `active` - the monitor runs on its schedule and can be triggered manually `paused` - the monitor does not run on its schedule and cannot be triggered manually **Enum:** `active`, `paused` **Example:** `active`
data. process_status	**string or null** In-flight run state, or `null` when the monitor is idle: `pending` - queued `processing` - currently running `failed` - the most recent run failed (cleared on next successful run) `paused` - the run was paused mid-flight (e.g. due to insufficient credit) Callers polling after `POST /monitors/{uuid}/run` should wait for this field to transition out of `pending` / `processing` before reading results. **Enum:** `pending`, `processing`, `failed`, `paused`
data. process_status_comment	**string or null** Optional human-readable detail accompanying `process_status` (e.g. failure message)
data. schedule	**string** How often the monitor runs unattended. `manual` means it never runs on its own and must be triggered via the run endpoint. **Enum:** `manual`, `every_day`, `weekdays`, `weekends`, `monday`, `tuesday`, `wednesday`, `thursday`, `friday`, `saturday`, `sunday`, `monthly_1`, `monthly_15`, `monthly_1_15`, `quarterly` **Example:** `weekdays`
data. schedule_name	**string or null** Human-readable description of the schedule (e.g. "Every Day (Mon-Sun)") **Example:** `Weekdays (Mon-Fri)`
data. config	**object (dynamic)** Per-operation configuration. The shape depends on `monitor_check_type`; see the Monitor Configuration section of the API guide for the schema of each operation. The portal's monitor create/edit screens render the same fields you'd send here.

Field	Description
data. last_run_at	**string (date-time) or null** ISO 8601 timestamp of the most recent run, or `null` if the monitor has never run **Example:** `2025-01-15T03:00:00Z`
data. has_run_before	**boolean** `true` if the monitor has processed at least one entity. Used by the "new" run mode to know whether there is a cursor to resume from. **Example:** `true`
data. program_uuid	**string (uuid)** UUID of the program this monitor belongs to **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. created_at	**string (date-time)** ISO 8601 timestamp at which the monitor was created **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** ISO 8601 timestamp at which the monitor was last updated **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "5d6c8f93-91d4-4d6a-bb5e-2c64e9a3b1f0",
        "monitor_number": "M-12-34-56",
        "monitor_name": "Weekly PEP Sweep",
        "monitor_check_type": "pep_sanction_check",
        "monitor_check_type_name": "PEP & Sanction Check",
        "status": "active",
        "process_status": null,
        "process_status_comment": null,
        "schedule": "weekdays",
        "schedule_name": "Weekdays (Mon-Fri)",
        "config": {
            "search_type": "pep_and_sanction",
            "similarity_threshold": 90
        },
        "last_run_at": "2025-01-15T03:00:00Z",
        "has_run_before": true,
        "program_uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
```

```
}
```

Standard error responses: 400, 401, 402, 403, 404, 429, 5XX (see Common error responses).

Update a monitor

PATCH `/monitors/{monitor_uuid}`

Update an existing monitor using PATCH semantics: only the fields you supply are changed; everything else is left as-is.

Restrictions

- `monitor_check_type` cannot be changed - delete the monitor and create a new one if you need a different operation type.
- The monitor must not be mid-run (`process_status` of `pending` / `processing` / `paused`). Attempting to update a mid-run monitor returns `409 Conflict` .

Partial config updates

The `config` object is merged key-by-key with the existing configuration: keys you do not supply are preserved. Required-config rules for the underlying operation only fire for keys you actually supply, so you can safely send a single key (e.g. `{ "config": { "similarity_threshold": 90 } }`) without echoing the entire config.

The full list of configuration keys, allowed values and defaults for every operation type is documented in the Monitor Configuration section of the API guide.

Path parameters

Field	Description
`monitor_uuid` REQUIRED	**string (uuid)** The monitor UUID

Request body

The monitor fields to update. All fields are optional.

Field	Description
`monitor_name`	**string [3..255 characters]**
`status`	**string** **Enum:** `active` , `paused`
`schedule`	**string** **Enum:** `manual` , `every_day` , `weekdays` , `weekends` , `monday` , `tuesday` , `wednesday` , `thursday` , `friday` , `saturday` , `sunday` , `monthly_1` , `monthly_15` , `monthly_1_15` , `quarterly`

Field	Description
config	**object (dynamic)** Per-operation configuration; merged with the existing configuration.

Sample request

```
{
    "monitor_name": "string",
    "status": "active",
    "schedule": "manual"
}
```

Responses

200 **Monitor updated** (application/json)

Field	Description
data	**object** A monitor is a scheduled (or manual) screening job that runs a single screening operation (PEP/sanction, ASIC, address verification, etc.) against the entities in its parent program. The operation type is fixed at creation; only configuration, name, status and schedule can be changed after that.
data. uuid	**string (uuid)** The monitor's UUID **Example:** `5d6c8f93-91d4-4d6a-bb5e-2c64e9a3b1f0`
data. monitor_number	**string** A short obfuscated public identifier for the monitor (e.g. "M-12-34-56") **Example:** `M-12-34-56`
data. monitor_name	**string** The display name of the monitor **Example:** `Weekly PEP Sweep`
data. monitor_check_type	**string** The screening operation this monitor runs. Cannot be changed after creation - create a new monitor instead. The list of allowed values depends on the products enabled on the calling account. **Example:** `pep_sanction_check`

Field	Description
data. monitor_check_type_name	**string** Human-readable name for `monitor_check_type` (e.g. "PEP & Sanction Check") **Example:** `PEP & Sanction Check`
data. status	**string** Lifecycle status: `active` - the monitor runs on its schedule and can be triggered manually `paused` - the monitor does not run on its schedule and cannot be triggered manually **Enum:** `active`, `paused` **Example:** `active`
data. process_status	**string or null** In-flight run state, or `null` when the monitor is idle: `pending` - queued `processing` - currently running `failed` - the most recent run failed (cleared on next successful run) `paused` - the run was paused mid-flight (e.g. due to insufficient credit) Callers polling after `POST /monitors/{uuid}/run` should wait for this field to transition out of `pending` / `processing` before reading results. **Enum:** `pending`, `processing`, `failed`, `paused`
data. process_status_comment	**string or null** Optional human-readable detail accompanying `process_status` (e.g. failure message)
data. schedule	**string** How often the monitor runs unattended. `manual` means it never runs on its own and must be triggered via the run endpoint. **Enum:** `manual`, `every_day`, `weekdays`, `weekends`, `monday`, `tuesday`, `wednesday`, `thursday`, `friday`, `saturday`, `sunday`, `monthly_1`, `monthly_15`, `monthly_1_15`, `quarterly` **Example:** `weekdays`
data. schedule_name	**string or null** Human-readable description of the schedule (e.g. "Every Day (Mon-Sun)") **Example:** `Weekdays (Mon-Fri)`
data. config	**object (dynamic)** Per-operation configuration. The shape depends on `monitor_check_type`; see the Monitor Configuration section of the API guide for the schema of each operation. The portal's monitor create/edit screens render the same fields you'd send here.

Field	Description
data. last_run_at	**string (date-time) or null** ISO 8601 timestamp of the most recent run, or `null` if the monitor has never run **Example:** `2025-01-15T03:00:00Z`
data. has_run_before	**boolean** `true` if the monitor has processed at least one entity. Used by the "new" run mode to know whether there is a cursor to resume from. **Example:** `true`
data. program_uuid	**string (uuid)** UUID of the program this monitor belongs to **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. created_at	**string (date-time)** ISO 8601 timestamp at which the monitor was created **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** ISO 8601 timestamp at which the monitor was last updated **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "5d6c8f93-91d4-4d6a-bb5e-2c64e9a3b1f0",
        "monitor_number": "M-12-34-56",
        "monitor_name": "Weekly PEP Sweep",
        "monitor_check_type": "pep_sanction_check",
        "monitor_check_type_name": "PEP & Sanction Check",
        "status": "active",
        "process_status": null,
        "process_status_comment": null,
        "schedule": "weekdays",
        "schedule_name": "Weekdays (Mon-Fri)",
        "config": {
            "search_type": "pep_and_sanction",
            "similarity_threshold": 90
        },
        "last_run_at": "2025-01-15T03:00:00Z",
        "has_run_before": true,
        "program_uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
```

```
}
```

409 **Conflict - the monitor cannot be edited in its current state (mid-run)** (application/json)

Field	Description
`message`	**string** **Example:** `Monitor cannot be edited in its current state`

Sample response

```
{
    "message": "Monitor cannot be edited in its current state"
}
```

Standard error responses: 400, 401, 402, 403, 404, 429, 5XX (see Common error responses).

Delete a monitor

DELETE `/monitors/{monitor_uuid}`

Soft-deletes the monitor. The deleted monitor is returned in the response as confirmation; subsequent show/list calls for it will return `404`.

See the Soft Deletion section of the API guide for the full policy (30-day recovery window, portal-only restore).

If you only want to stop the monitor running but keep it visible (for example to retain its configuration and history for audit), set its `status` to `paused` via `PATCH /v1/monitors/{uuid}` instead of deleting it. Monitors do not have a separate archive state.

Restrictions

The monitor must not be mid-run (`process_status` of `pending` / `processing` / `paused`). Attempting to delete a mid-run monitor returns `409 Conflict`. There is no API equivalent of the portal's "Restore monitor" - undeleting is a portal-only action.

Path parameters

Field	Description
`monitor_uuid` REQUIRED	**string (uuid)** The monitor UUID

Responses

200 **Monitor deleted** (application/json)

Field	Description

`data`	**object** A monitor is a scheduled (or manual) screening job that runs a single screening operation (PEP/sanction, ASIC, address verification, etc.) against the entities in its parent program. The operation type is fixed at creation; only configuration, name, status and schedule can be changed after that.
`data.` `uuid`	**string (uuid)** The monitor's UUID **Example:** `5d6c8f93-91d4-4d6a-bb5e-2c64e9a3b1f0`
`data.` `monitor_number`	**string** A short obfuscated public identifier for the monitor (e.g. "M-12-34-56") **Example:** `M-12-34-56`
`data.` `monitor_name`	**string** The display name of the monitor **Example:** `Weekly PEP Sweep`
`data.` `monitor_check_type`	**string** The screening operation this monitor runs. Cannot be changed after creation - create a new monitor instead. The list of allowed values depends on the products enabled on the calling account. **Example:** `pep_sanction_check`
`data.` `monitor_check_type_name`	**string** Human-readable name for `monitor_check_type` (e.g. "PEP & Sanction Check") **Example:** `PEP & Sanction Check`
`data.` `status`	**string** Lifecycle status: `active` - the monitor runs on its schedule and can be triggered manually `paused` - the monitor does not run on its schedule and cannot be triggered manually **Enum:** `active`, `paused` **Example:** `active`

data. process_status	**string or null** In-flight run state, or `null` when the monitor is idle: `pending` - queued `processing` - currently running `failed` - the most recent run failed (cleared on next successful run) `paused` - the run was paused mid-flight (e.g. due to insufficient credit) Callers polling after `POST /monitors/{uuid}/run` should wait for this field to transition out of `pending` / `processing` before reading results. **Enum:** `pending`, `processing`, `failed`, `paused`
data. process_status_comment	**string or null** Optional human-readable detail accompanying `process_status` (e.g. failure message)
data. schedule	**string** How often the monitor runs unattended. `manual` means it never runs on its own and must be triggered via the run endpoint. **Enum:** `manual`, `every_day`, `weekdays`, `weekends`, `monday`, `tuesday`, `wednesday`, `thursday`, `friday`, `saturday`, `sunday`, `monthly_1`, `monthly_15`, `monthly_1_15`, `quarterly` **Example:** `weekdays`
data. schedule_name	**string or null** Human-readable description of the schedule (e.g. "Every Day (Mon-Sun)") **Example:** `Weekdays (Mon-Fri)`
data. config	**object (dynamic)** Per-operation configuration. The shape depends on `monitor_check_type`; see the Monitor Configuration section of the API guide for the schema of each operation. The portal's monitor create/edit screens render the same fields you'd send here.
data. last_run_at	**string (date-time) or null** ISO 8601 timestamp of the most recent run, or `null` if the monitor has never run **Example:** `2025-01-15T03:00:00Z`
data. has_run_before	**boolean** `true` if the monitor has processed at least one entity. Used by the "new" run mode to know whether there is a cursor to resume from. **Example:** `true`
data. program_uuid	**string (uuid)** UUID of the program this monitor belongs to **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`

data. created_at	**string (date-time)** ISO 8601 timestamp at which the monitor was created **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** ISO 8601 timestamp at which the monitor was last updated **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "5d6c8f93-91d4-4d6a-bb5e-2c64e9a3b1f0",
        "monitor_number": "M-12-34-56",
        "monitor_name": "Weekly PEP Sweep",
        "monitor_check_type": "pep_sanction_check",
        "monitor_check_type_name": "PEP & Sanction Check",
        "status": "active",
        "process_status": null,
        "process_status_comment": null,
        "schedule": "weekdays",
        "schedule_name": "Weekdays (Mon-Fri)",
        "config": {
            "search_type": "pep_and_sanction",
            "similarity_threshold": 90
        },
        "last_run_at": "2025-01-15T03:00:00Z",
        "has_run_before": true,
        "program_uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


409 **Conflict - the monitor cannot be deleted in its current state (mid-run)** (application/json)

Field	Description
message	**string** **Example:** `Monitor cannot be deleted in its current state`

Sample response


```
{
    "message": "Monitor cannot be deleted in its current state"
}
```


Standard error responses: 400, 401, 402, 403, 404, 429, 5XX (see Common error responses).

Run a monitor on demand

POST `/monitors/{monitor_uuid}/run`

Queues an immediate run of the monitor and returns `202 Accepted` once the run has been scheduled. The actual screening happens in the background - poll `GET /monitors/{monitor_uuid}` and wait for `process_status` to transition out of `pending` / `processing` before reading the results.

Run types

The `run_type` field controls which entities the monitor processes:

- `new` - process only entities created since the last run (the typical scheduled-run behaviour). On a never-run monitor this processes every active entity.
- `all` - process every active entity in the program, regardless of when it was created or whether it has been processed before. Use this to re-screen everything from scratch (for example, after a config change).

Both modes only ever bill for entities that are eligible to be screened by the underlying operation (e.g. a UK ASIC check skips non-AU businesses).

Pre-flight checks

The endpoint runs several pre-flight checks before queueing the job. Each rejection is surfaced with a distinct status code so you can branch appropriately:

- `404` - the monitor was not found
- `409 Conflict` - the monitor is not currently runnable. The most common cause is that the monitor is already mid-run (`process_status` of `pending` , `processing` or `paused`); other causes include the monitor being paused or the parent account being inactive. The body's `message` describes the reason.
- `402 Payment Required` - the calling account has insufficient credit to cover the projected billable count. The response body includes a `billing` object with the cost breakdown so you can show the user exactly what would have been charged versus what they have available.

Idempotency

This endpoint accepts the `Idempotency-Key` header so that retries on network failure are safe. See the Idempotency section of the API guide for the full contract.

Path parameters

Field	Description
`monitor_uuid` REQUIRED	**string (uuid)** The monitor UUID

Header parameters

Field	Description
`Idempotency-Key`	**string [max 255 characters]** Optional client-generated key (typically a UUID) that lets you safely retry a write request without risk of duplicate work or duplicate billing. A retry with the same key and body returns the original response verbatim, with the `Idempotent-Replay: true` response header. Honoured on `POST` requests only. See the Idempotency section of the API guide for the full contract. **Example:** `8c4d3a18-2f63-4f9a-9f3a-9b1f5a7c2d4f`

Request body

The run parameters

Field	Description
`run_type` REQUIRED	**string** Which entities to process. See the description above for the difference between `new` and `all`. **Enum:** `new`, `all` **Example:** `new`

Sample request

```
{
    "run_type": "new"
}
```

Responses

202 **Run accepted - the monitor has been queued for processing. The response body shows the monitor with `process_status: pending`. Poll the show endpoint to observe the status transition out of `pending` / `processing`.**
(application/json)

Field	Description
`data`	**object** A monitor is a scheduled (or manual) screening job that runs a single screening operation (PEP/sanction, ASIC, address verification, etc.) against the entities in its parent program. The operation type is fixed at creation; only configuration, name, status and schedule can be changed after that.

Field	Description
data. uuid	**string (uuid)** The monitor's UUID **Example:** `5d6c8f93-91d4-4d6a-bb5e-2c64e9a3b1f0`
data. monitor_number	**string** A short obfuscated public identifier for the monitor (e.g. "M-12-34-56") **Example:** `M-12-34-56`
data. monitor_name	**string** The display name of the monitor **Example:** `Weekly PEP Sweep`
data. monitor_check_type	**string** The screening operation this monitor runs. Cannot be changed after creation - create a new monitor instead. The list of allowed values depends on the products enabled on the calling account. **Example:** `pep_sanction_check`
data. monitor_check_type_name	**string** Human-readable name for `monitor_check_type` (e.g. "PEP & Sanction Check") **Example:** `PEP & Sanction Check`
data. status	**string** Lifecycle status: `active` - the monitor runs on its schedule and can be triggered manually `paused` - the monitor does not run on its schedule and cannot be triggered manually **Enum:** `active`, `paused` **Example:** `active`
data. process_status	**string or null** In-flight run state, or `null` when the monitor is idle: `pending` - queued `processing` - currently running `failed` - the most recent run failed (cleared on next successful run) `paused` - the run was paused mid-flight (e.g. due to insufficient credit) Callers polling after `POST /monitors/{uuid}/run` should wait for this field to transition out of `pending` / `processing` before reading results. **Enum:** `pending`, `processing`, `failed`, `paused`
data. process_status_comment	**string or null** Optional human-readable detail accompanying `process_status` (e.g. failure message)

Field	Description
data. schedule	**string** How often the monitor runs unattended. `manual` means it never runs on its own and must be triggered via the run endpoint. **Enum:** `manual`, `every_day`, `weekdays`, `weekends`, `monday`, `tuesday`, `wednesday`, `thursday`, `friday`, `saturday`, `sunday`, `monthly_1`, `monthly_15`, `monthly_1_15`, `quarterly` **Example:** `weekdays`
data. schedule_name	**string or null** Human-readable description of the schedule (e.g. "Every Day (Mon-Sun)") **Example:** `Weekdays (Mon-Fri)`
data. config	**object (dynamic)** Per-operation configuration. The shape depends on `monitor_check_type`; see the Monitor Configuration section of the API guide for the schema of each operation. The portal's monitor create/edit screens render the same fields you'd send here.
data. last_run_at	**string (date-time) or null** ISO 8601 timestamp of the most recent run, or `null` if the monitor has never run **Example:** `2025-01-15T03:00:00Z`
data. has_run_before	**boolean** `true` if the monitor has processed at least one entity. Used by the "new" run mode to know whether there is a cursor to resume from. **Example:** `true`
data. program_uuid	**string (uuid)** UUID of the program this monitor belongs to **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. created_at	**string (date-time)** ISO 8601 timestamp at which the monitor was created **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** ISO 8601 timestamp at which the monitor was last updated **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response

```
{
    "data": {
        "uuid": "5d6c8f93-91d4-4d6a-bb5e-2c64e9a3b1f0",
        "monitor_number": "M-12-34-56",
        "monitor_name": "Weekly PEP Sweep",
        "monitor_check_type": "pep_sanction_check",
        "monitor_check_type_name": "PEP & Sanction Check",
        "status": "active",
        "process_status": null,
        "process_status_comment": null,
        "schedule": "weekdays",
        "schedule_name": "Weekdays (Mon-Fri)",
        "config": {
            "search_type": "pep_and_sanction",
            "similarity_threshold": 90
        },
        "last_run_at": "2025-01-15T03:00:00Z",
        "has_run_before": true,
        "program_uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


409 **Conflict - the monitor cannot be run in its current state. The most common cause is that the monitor is already mid-run; other causes include the monitor being paused or the parent account being inactive.** (application/json)

Field	Description
message	string **Example:** `Monitor cannot be run while it is already pending`

Sample response


```
{
    "message": "Monitor cannot be run while it is already pending"
}
```


422 **Idempotency-Key was reused with a different request body** (application/json)

Field	Description
message	string **Example:** `Idempotency-Key was reused with a different request body.`

Sample response


```
{
    "message": "Idempotency-Key was reused with a different request body."
}
```


Standard error responses: 400, 401, 402, 403, 404, 429, 5XX (see Common error responses).

Notes

Free-text notes attached to entities and events.

List notes

GET `/notes`

Returns notes for the calling account. Results are paginated.

Notes are always attached to an entity (and the entity's program). Notes created in the context of an event additionally carry an `event_uuid`. Use the filters to narrow to a specific entity, program, event, or to a specific author or author type.

Filters

The following filters can be applied via the `filter[...]` query parameters:

- `entity_uuid` - only notes for the given entity
- `event_uuid` - only notes attached to the given event
- `program_uuid` - only notes for entities in the given program
- `created_by` - one of `user` (only notes authored by any portal user on this account) or `api_key` (only notes authored by any API key on this account). Use this for the coarse "human vs automation" distinction.
- `created_by_user_uuid` - only notes authored by the given portal user. The uuid must be a user on the calling account; otherwise the result is an empty page.
- `created_by_api_key_uuid` - only notes authored by the given API key. The uuid must be an API key on the calling account; otherwise the result is an empty page.
- `created_at_from` - inclusive lower bound on note creation time (ISO 8601)
- `created_at_to` - inclusive upper bound on note creation time (ISO 8601)

The top-level `archived` query parameter accepts `true` (only archived) or `false` (only non-archived); omitting it returns both. (Archive is a portal-only operation - the API surfaces the state but does not currently expose archive/unarchive endpoints.)

Filters that target a uuid (entity, event, program) return an empty page when the target uuid does not exist on the calling account, identical to the behaviour for a uuid that does not exist at all.

Sorting

The `sort` query parameter accepts `created_at` (default: `-created_at`, newest first) or `updated_at`. Prefix with `-` for descending order.

Query parameters

Field	Description

`page`	**integer** The page of results to return, starting at 1. **Example:** `1`
`per_page`	**integer** The number of notes per page (defaults to 30, max 500) **Example:** `30`
`filter[entity_uuid]`	**string (uuid)** Only records for the given entity.
`filter[event_uuid]`	**string (uuid)** Only notes attached to the given event.
`filter[program_uuid]`	**string (uuid)** Only records for entities in the given program.
`filter[created_by]`	**string** Filter to notes authored by any portal user (`user`) or any API key (`api_key`) on this account. **Enum:** `user` , `api_key`
`filter[created_by_user_uuid]`	**string (uuid)** Filter to notes authored by a specific portal user.
`filter[created_by_api_key_uuid]`	**string (uuid)** Filter to notes authored by a specific API key.
`filter[created_at_from]`	**string (date-time)** Inclusive lower bound on `created_at` (ISO 8601). **Example:** `2025-01-01T00:00:00Z`
`filter[created_at_to]`	**string (date-time)** Inclusive upper bound on `created_at` (ISO 8601). **Example:** `2025-12-31T23:59:59Z`
`sort`	**string** Field to sort by; prefix with `-` for descending order **Enum:** `created_at` , `-created_at` , `updated_at` , `-updated_at` **Example:** `-created_at`

archived	**boolean** Filter by archive status. `true` - only archived notes `false` - only non-archived notes Note: The API will accept the string 'true' or the number 1 for true and the string 'false' or the number 0 for false as well as the boolean values.

Responses

200 **Notes list response** (application/json)

Field	Description
data	**array of objects** An array of notes
data[]. uuid	**string (uuid)** The note's UUID **Example:** `7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data[]. note_number	**string** A short obfuscated public identifier for the note (e.g. "N-A1-B2C-3D4") **Example:** `N-A1-B2C-3D4`
data[]. note_text	**string** The free-text content of the note (max 5000 characters) **Example:** `Confirmed match against verified passport details.`
data[]. created_by	**object or null** Identifies who created the note. The `type` discriminator picks the shape of the remaining fields: `{ type: "user", uuid, username }` - a portal user created the note. The portal user (or an account manager) can edit/delete it through the portal. The API cannot edit or delete user-authored notes. `{ type: "api_key", uuid, label }` - an API key on this account created the note. The API can manage these notes via the PATCH and DELETE endpoints. The portal cannot edit or delete API-key-authored notes. `null` - rare legacy notes from before actor attribution was recorded; treat as read-only.
data[]. created_by. type	**string** Discriminator for the actor type. **Enum:** `user`, `api_key`

Field	Description
data[]. created_by. uuid	**string (uuid)** UUID of the user or API key that created the note.
data[]. created_by. username	**string** Username of the portal user. Present only when `type` is `user`.
data[]. created_by. label	**string** Label of the API key. Present only when `type` is `api_key`.
data[]. is_immutable	**boolean** When true, the note is an immutable audit-trail entry and cannot be edited or deleted by any actor through any channel. PATCH and DELETE requests against an immutable note return 403. **Example:** `false`
data[]. entity_uuid	**string (uuid)** UUID of the entity this note is attached to
data[]. program_uuid	**string (uuid)** UUID of the program the entity belongs to
data[]. event_uuid	**string (uuid) or null** UUID of the event this note is attached to, if any. Notes created via the Events PATCH endpoint with a `note` field carry an `event_uuid`; notes created via POST /v1/entities/{uuid}/notes do not.
data[]. archived_at	**string (date-time) or null** ISO 8601 timestamp at which the note was archived in the portal (null if not archived)
data[]. created_at	**string (date-time)** **Example:** `2025-01-01T00:00:00Z`
data[]. updated_at	**string (date-time)** **Example:** `2025-01-01T00:00:00Z`
meta	**object**
meta. current_page	**integer** **Example:** `1`
meta. per_page	**integer** **Example:** `30`
meta. total	**integer** **Example:** `1`

Field	Description
meta. last_page	integer Example: `1`
api_reference	string (uuid)

Sample response


```
{
    "data": [
        {
            "uuid": "7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
            "note_number": "N-A1-B2C-3D4",
            "note_text": "Confirmed match against verified passport details.",
            "created_by": {
                "type": "user",
                "uuid": "00000000-0000-0000-0000-000000000000",
                "username": "string",
                "label": "string"
            },
            "is_immutable": false,
            "entity_uuid": "00000000-0000-0000-0000-000000000000",
            "program_uuid": "00000000-0000-0000-0000-000000000000",
            "event_uuid": "00000000-0000-0000-0000-000000000000",
            "archived_at": "2024-01-01T00:00:00Z",
            "created_at": "2025-01-01T00:00:00Z",
            "updated_at": "2025-01-01T00:00:00Z"
        }
    ],
    "meta": {
        "current_page": 1,
        "per_page": 30,
        "total": 1,
        "last_page": 1
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 403, 429, 5XX (see Common error responses).

Create a note on an entity

POST `/entities/{entity_uuid}/notes`

Create a new note attached to the specified entity. The note inherits the entity's program and account.

Notes created via this endpoint are attributed to the calling API key (the response `created_by` carries `{ type: "api_key", uuid, label }`). They cannot be edited or deleted from the portal. Any API key on the same account can manage these notes via the PATCH and DELETE endpoints.

To attach a note to an event (rather than just to the entity) use the Events PATCH endpoint with a `note` field.

Idempotency

This endpoint accepts the `Idempotency-Key` header so that retries on network failure are safe. See the Idempotency section of the API guide for the full contract.

Path parameters

Field	Description
`entity_uuid` REQUIRED	**string (uuid)** The entity UUID

Header parameters

Field	Description
`Idempotency-Key`	**string [max 255 characters]** Optional client-generated key (typically a UUID) that lets you safely retry a write request without risk of duplicate work or duplicate billing. A retry with the same key and body returns the original response verbatim, with the `Idempotent-Replay: true` response header. Honoured on `POST` requests only. See the Idempotency section of the API guide for the full contract. **Example:** `8c4d3a18-2f63-4f9a-9f3a-9b1f5a7c2d4f`

Request body

The note details

Field	Description
`note_text` REQUIRED	**string [max 5000 characters]** Free-text content of the note. Max 5000 characters. **Example:** `Synced from CRM ticket`

Sample request

```
{
    "note_text": "Synced from CRM ticket"
}
```

Responses

201 **Note created** (application/json)

Field	Description
`data`	**object** A note attached to an entity, optionally also linked to an event. Deleting a note is a soft-delete: the record stays in the database for audit / compliance purposes but is no longer surfaced by the API. Subsequent show or list calls for a deleted note return `404`. Notes can be archived in the portal (the `archived_at` timestamp); the API surfaces this state for filtering but archive/unarchive operations are not currently exposed via the API.
`data.` `uuid`	**string (uuid)** The note's UUID **Example:** `7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
`data.` `note_number`	**string** A short obfuscated public identifier for the note (e.g. "N-A1-B2C-3D4") **Example:** `N-A1-B2C-3D4`
`data.` `note_text`	**string** The free-text content of the note (max 5000 characters) **Example:** `Confirmed match against verified passport details.`
`data.` `created_by`	**object or null** Identifies who created the note. The `type` discriminator picks the shape of the remaining fields: `{ type: "user", uuid, username }` - a portal user created the note. The portal user (or an account manager) can edit/delete it through the portal. The API cannot edit or delete user-authored notes. `{ type: "api_key", uuid, label }` - an API key on this account created the note. The API can manage these notes via the PATCH and DELETE endpoints. The portal cannot edit or delete API-key-authored notes. `null` - rare legacy notes from before actor attribution was recorded; treat as read-only.

`data.`
`created_by.`
`type`
	string
	Discriminator for the actor type.
	Enum: `user`, `api_key`

`data.`
`created_by.`
`uuid`
	string (uuid)
	UUID of the user or API key that created the note.

`data.`
`created_by.`
`username`
	string
	Username of the portal user. Present only when `type` is `user`.

Field	Description
data. created_by. label	**string** Label of the API key. Present only when `type` is `api_key`.
data. is_immutable	**boolean** When true, the note is an immutable audit-trail entry and cannot be edited or deleted by any actor through any channel. PATCH and DELETE requests against an immutable note return 403. **Example:** `false`
data. entity_uuid	**string (uuid)** UUID of the entity this note is attached to
data. program_uuid	**string (uuid)** UUID of the program the entity belongs to
data. event_uuid	**string (uuid) or null** UUID of the event this note is attached to, if any. Notes created via the Events PATCH endpoint with a `note` field carry an `event_uuid`; notes created via POST /v1/entities/{uuid}/notes do not.
data. archived_at	**string (date-time) or null** ISO 8601 timestamp at which the note was archived in the portal (null if not archived)
data. created_at	**string (date-time)** **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "note_number": "N-A1-B2C-3D4",
        "note_text": "Confirmed match against verified passport details.",
        "created_by": {
            "type": "user",
            "uuid": "00000000-0000-0000-0000-000000000000",
            "username": "string",
            "label": "string"
        },
        "is_immutable": false,
        "entity_uuid": "00000000-0000-0000-0000-000000000000",
        "program_uuid": "00000000-0000-0000-0000-000000000000",
```

```
        "event_uuid": "00000000-0000-0000-0000-000000000000",
        "archived_at": "2024-01-01T00:00:00Z",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


409 **Conflict - a request with this Idempotency-Key is currently being processed** (application/json)

Field	Description
message	string **Example:** `A request with this Idempotency-Key is already being processed.`

Sample response


```
{
    "message": "A request with this Idempotency-Key is already being processed."
}
```


422 **Idempotency-Key was reused with a different request body** (application/json)

Field	Description
message	string **Example:** `Idempotency-Key was reused with a different request body.`

Sample response


```
{
    "message": "Idempotency-Key was reused with a different request body."
}
```


Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

Show a note

GET `/notes/{note_uuid}`

Returns a single note by its UUID.

Path parameters

Field	Description

`note_uuid` REQUIRED	**string (uuid)** The note UUID

Responses

200 **Note response** (application/json)

Field	Description
`data`	**object** A note attached to an entity, optionally also linked to an event. Deleting a note is a soft-delete: the record stays in the database for audit / compliance purposes but is no longer surfaced by the API. Subsequent show or list calls for a deleted note return `404`. Notes can be archived in the portal (the `archived_at` timestamp); the API surfaces this state for filtering but archive/unarchive operations are not currently exposed via the API.
`data.` `uuid`	**string (uuid)** The note's UUID **Example:** `7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
`data.` `note_number`	**string** A short obfuscated public identifier for the note (e.g. "N-A1-B2C-3D4") **Example:** `N-A1-B2C-3D4`
`data.` `note_text`	**string** The free-text content of the note (max 5000 characters) **Example:** `Confirmed match against verified passport details.`
`data.` `created_by`	**object or null** Identifies who created the note. The `type` discriminator picks the shape of the remaining fields: `{ type: "user", uuid, username }` - a portal user created the note. The portal user (or an account manager) can edit/delete it through the portal. The API cannot edit or delete user-authored notes. `{ type: "api_key", uuid, label }` - an API key on this account created the note. The API can manage these notes via the PATCH and DELETE endpoints. The portal cannot edit or delete API-key-authored notes. `null` - rare legacy notes from before actor attribution was recorded; treat as read-only.

`data.`
`created_by.`
`type`
	string
	Discriminator for the actor type.
	Enum: `user`, `api_key`

Field	Description
data. created_by. uuid	**string (uuid)** UUID of the user or API key that created the note.
data. created_by. username	**string** Username of the portal user. Present only when `type` is `user`.
data. created_by. label	**string** Label of the API key. Present only when `type` is `api_key`.
data. is_immutable	**boolean** When true, the note is an immutable audit-trail entry and cannot be edited or deleted by any actor through any channel. PATCH and DELETE requests against an immutable note return 403. **Example:** `false`
data. entity_uuid	**string (uuid)** UUID of the entity this note is attached to
data. program_uuid	**string (uuid)** UUID of the program the entity belongs to
data. event_uuid	**string (uuid) or null** UUID of the event this note is attached to, if any. Notes created via the Events PATCH endpoint with a `note` field carry an `event_uuid`; notes created via POST /v1/entities/{uuid}/notes do not.
data. archived_at	**string (date-time) or null** ISO 8601 timestamp at which the note was archived in the portal (null if not archived)
data. created_at	**string (date-time)** **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "note_number": "N-A1-B2C-3D4",
        "note_text": "Confirmed match against verified passport details.",
        "created_by": {
            "type": "user",
```

```
            "uuid": "00000000-0000-0000-0000-000000000000",
            "username": "string",
            "label": "string"
        },
        "is_immutable": false,
        "entity_uuid": "00000000-0000-0000-0000-000000000000",
        "program_uuid": "00000000-0000-0000-0000-000000000000",
        "event_uuid": "00000000-0000-0000-0000-000000000000",
        "archived_at": "2024-01-01T00:00:00Z",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

Update a note

PATCH `/notes/{note_uuid}`

Update a note's text. Only `note_text` can be changed - the note's parent entity, parent event, author and archive state are all immutable from the API. To "move" a note to a different entity or event, delete it and re-create.

Only notes authored by an API key on this account can be updated through this endpoint:

- Notes authored by a portal user (`created_by.type` is `user`) return 404 from this endpoint, even if the note exists - it can only be edited via the portal by its author or an account manager.
- Notes with `is_immutable: true` (for example, the per-event audit-trail note left by a bulk update) return 403 because no actor on any channel can modify them.

Path parameters

Field	Description
`note_uuid` REQUIRED	string (uuid) The note UUID

Request body

The fields to update

Field	Description
`note_text` REQUIRED	string [max 5000 characters] Replacement text for the note. Max 5000 characters. **Example:** `Updated - customer's identity has now been independently verified.`

Sample request

```
{
    "note_text": "Updated - customer's identity has now been independently verified."
}
```

Responses

200 **Note updated** (application/json)

Field	Description
`data`	**object** A note attached to an entity, optionally also linked to an event. Deleting a note is a soft-delete: the record stays in the database for audit / compliance purposes but is no longer surfaced by the API. Subsequent show or list calls for a deleted note return `404`. Notes can be archived in the portal (the `archived_at` timestamp); the API surfaces this state for filtering but archive/unarchive operations are not currently exposed via the API.
`data.` `uuid`	**string (uuid)** The note's UUID **Example:** `7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
`data.` `note_number`	**string** A short obfuscated public identifier for the note (e.g. "N-A1-B2C-3D4") **Example:** `N-A1-B2C-3D4`
`data.` `note_text`	**string** The free-text content of the note (max 5000 characters) **Example:** `Confirmed match against verified passport details.`
`data.` `created_by`	**object or null** Identifies who created the note. The `type` discriminator picks the shape of the remaining fields: `{ type: "user", uuid, username }` - a portal user created the note. The portal user (or an account manager) can edit/delete it through the portal. The API cannot edit or delete user-authored notes. `{ type: "api_key", uuid, label }` - an API key on this account created the note. The API can manage these notes via the PATCH and DELETE endpoints. The portal cannot edit or delete API-key-authored notes. `null` - rare legacy notes from before actor attribution was recorded; treat as read-only.

Field	Description
data. created_by. type	**string** Discriminator for the actor type. **Enum:** `user`, `api_key`
data. created_by. uuid	**string (uuid)** UUID of the user or API key that created the note.
data. created_by. username	**string** Username of the portal user. Present only when `type` is `user`.
data. created_by. label	**string** Label of the API key. Present only when `type` is `api_key`.
data. is_immutable	**boolean** When true, the note is an immutable audit-trail entry and cannot be edited or deleted by any actor through any channel. PATCH and DELETE requests against an immutable note return 403. **Example:** `false`
data. entity_uuid	**string (uuid)** UUID of the entity this note is attached to
data. program_uuid	**string (uuid)** UUID of the program the entity belongs to
data. event_uuid	**string (uuid) or null** UUID of the event this note is attached to, if any. Notes created via the Events PATCH endpoint with a `note` field carry an `event_uuid`; notes created via POST /v1/entities/{uuid}/notes do not.
data. archived_at	**string (date-time) or null** ISO 8601 timestamp at which the note was archived in the portal (null if not archived)
data. created_at	**string (date-time)** **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response

```
{
    "data": {
```

```
        "uuid": "7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "note_number": "N-A1-B2C-3D4",
        "note_text": "Confirmed match against verified passport details.",
        "created_by": {
            "type": "user",
            "uuid": "00000000-0000-0000-0000-000000000000",
            "username": "string",
            "label": "string"
        },
        "is_immutable": false,
        "entity_uuid": "00000000-0000-0000-0000-000000000000",
        "program_uuid": "00000000-0000-0000-0000-000000000000",
        "event_uuid": "00000000-0000-0000-0000-000000000000",
        "archived_at": "2024-01-01T00:00:00Z",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

Delete a note

DELETE `/notes/{note_uuid}`

Soft-delete a note. The note remains in the database for audit / compliance purposes but is removed from list / show responses (a subsequent show or list call returns 404).

See the Soft Deletion section of the API guide for the full policy (30-day recovery window, portal-only restore).

If you need to keep the note accessible for compliance / audit purposes but no longer want it surfaced in day-to-day workflows, prefer archiving instead of deleting. Archiving a note is currently a portal-only action; the API surfaces the resulting `archived_at` timestamp and lets you filter by it. The deleted note record is returned in the response as confirmation.

Only notes authored by an API key on this account can be deleted through this endpoint:

- Notes authored by a portal user (`created_by.type` is `user`) return 404, mirroring the PATCH behaviour - portal-authored notes are managed through the portal.
- Notes with `is_immutable: true` return 403 because they record an audit-trail entry that no actor on any channel can remove.

Notes have no dependent records, so this is a single-record delete.

Path parameters

Field	Description
`note_uuid` REQUIRED	**string (uuid)** The note UUID

Responses

200 **Note deleted (soft-delete - returned as confirmation)** (application/json)

Field	Description
`data`	**object** A note attached to an entity, optionally also linked to an event. Deleting a note is a soft-delete: the record stays in the database for audit / compliance purposes but is no longer surfaced by the API. Subsequent show or list calls for a deleted note return `404`. Notes can be archived in the portal (the `archived_at` timestamp); the API surfaces this state for filtering but archive/unarchive operations are not currently exposed via the API.
`data.` `uuid`	**string (uuid)** The note's UUID **Example:** `7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
`data.` `note_number`	**string** A short obfuscated public identifier for the note (e.g. "N-A1-B2C-3D4") **Example:** `N-A1-B2C-3D4`
`data.` `note_text`	**string** The free-text content of the note (max 5000 characters) **Example:** `Confirmed match against verified passport details.`
`data.` `created_by`	**object or null** Identifies who created the note. The `type` discriminator picks the shape of the remaining fields: `{ type: "user", uuid, username }` - a portal user created the note. The portal user (or an account manager) can edit/delete it through the portal. The API cannot edit or delete user-authored notes. `{ type: "api_key", uuid, label }` - an API key on this account created the note. The API can manage these notes via the PATCH and DELETE endpoints. The portal cannot edit or delete API-key-authored notes. `null` - rare legacy notes from before actor attribution was recorded; treat as read-only.

`data.`
`created_by.`
`type`
	string
	Discriminator for the actor type.
	Enum: `user`, `api_key`

`data.`
`created_by.`
`uuid`
	string (uuid)
	UUID of the user or API key that created the note.

Field	Description
data. created_by. username	**string** Username of the portal user. Present only when `type` is `user`.
data. created_by. label	**string** Label of the API key. Present only when `type` is `api_key`.
data. is_immutable	**boolean** When true, the note is an immutable audit-trail entry and cannot be edited or deleted by any actor through any channel. PATCH and DELETE requests against an immutable note return 403. **Example:** `false`
data. entity_uuid	**string (uuid)** UUID of the entity this note is attached to
data. program_uuid	**string (uuid)** UUID of the program the entity belongs to
data. event_uuid	**string (uuid) or null** UUID of the event this note is attached to, if any. Notes created via the Events PATCH endpoint with a `note` field carry an `event_uuid`; notes created via POST /v1/entities/{uuid}/notes do not.
data. archived_at	**string (date-time) or null** ISO 8601 timestamp at which the note was archived in the portal (null if not archived)
data. created_at	**string (date-time)** **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "note_number": "N-A1-B2C-3D4",
        "note_text": "Confirmed match against verified passport details.",
        "created_by": {
            "type": "user",
            "uuid": "00000000-0000-0000-0000-000000000000",
            "username": "string",
            "label": "string"
        },
        "is_immutable": false,
```

```
        "entity_uuid": "00000000-0000-0000-0000-000000000000",
        "program_uuid": "00000000-0000-0000-0000-000000000000",
        "event_uuid": "00000000-0000-0000-0000-000000000000",
        "archived_at": "2024-01-01T00:00:00Z",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

Programs

Programs group the entities you monitor and the monitors that screen them.

List programs

GET `/programs`

Returns the programs belonging to the calling account. Soft-deleted programs are excluded. Results are paginated.

Filters

The following filter can be applied via the `filter[...]` query parameters:

- `status` - one of `active`, `paused`

Sorting

The `sort` query parameter accepts `created_at` (default: `-created_at`, newest first) or `program_name`. Prefix with `-` for descending order.

Query parameters

Field	Description
`page`	**integer** The page number to return **Example:** `1`
`per_page`	**integer** The number of programs per page (defaults to 30, max 500) **Example:** `30`

Field	Description
`filter[status]`	**string** Only records with the given status. The endpoint description lists the values. **Enum:** `active`, `paused`
`sort`	**string** Field to sort by; prefix with `-` for descending order **Enum:** `created_at`, `-created_at`, `program_name`, `-program_name` **Example:** `-created_at`

Responses

200 **Programs list response** (application/json)

Field	Description
`data`	**array of objects** An array of programs
`data[].` `uuid`	**string (uuid)** The program's UUID **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
`data[].` `program_number`	**string** A short obfuscated public identifier for the program (e.g. "P-123-456-789") **Example:** `P-123-456-789`
`data[].` `program_name`	**string** The display name of the program **Example:** `AML Onboarding Program`
`data[].` `status`	**string** Lifecycle status of the program: `active` - normal operation `paused` - monitors do not run; entities and checks remain accessible `archived` - read-only; the program is hidden from default lists but not deleted **Enum:** `active`, `paused`, `archived` **Example:** `active`
`data[].` `report_emails`	**array of strings** Email addresses (max 5) that receive event report notifications for this program

Field	Description
data[]. data_retention_months	**integer** How long, in months, to keep entity data for. Permitted values: `0` (no retention), `1`, `3`, `6`, `12`, `24`, `36`, `48`, `60`, `72`, `84`. **Example:** `12`
data[]. event_grouping	**string** Whether events from the same entity should be grouped into one event: `separate` - each event is created as a separate event (capped by `max_separate_events`) `grouped` - events are merged into a single event per entity **Enum:** `separate`, `grouped` **Example:** `separate`
data[]. max_separate_events	**integer or null** When `event_grouping` is `separate`, the maximum number of separate events to create for a single entity (5-100). Null when `event_grouping` is `grouped`. **Example:** `10`
data[]. default_risk_level	**string or null** The risk level pre-filled when a new entity is added to this program: `low` / `medium` / `high` - new entities pre-fill with this risk rating `null` - new entities start with no risk assigned; the rating may be left unassigned and set later (bulk file uploads assign `low`) **Enum:** `low`, `medium`, `high` **Example:** `low`
data[]. created_at	**string (date-time)** ISO 8601 timestamp at which the program was created **Example:** `2025-01-01T00:00:00Z`
data[]. updated_at	**string (date-time)** ISO 8601 timestamp at which the program was last updated **Example:** `2025-01-01T00:00:00Z`
meta	**object**
meta. current_page	**integer** **Example:** `1`
meta. per_page	**integer** **Example:** `30`

Field	Description
meta. total	integer **Example:** `1`
meta. last_page	integer **Example:** `1`
api_reference	string (uuid) A unique identifier for this request. This can be used to track the request in the logs and audit trail.

Sample response


```
{
    "data": [
        {
            "uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
            "program_number": "P-123-456-789",
            "program_name": "AML Onboarding Program",
            "status": "active",
            "report_emails": [
                "compliance@example.com"
            ],
            "data_retention_months": 12,
            "event_grouping": "separate",
            "max_separate_events": 10,
            "default_risk_level": "low",
            "created_at": "2025-01-01T00:00:00Z",
            "updated_at": "2025-01-01T00:00:00Z"
        }
    ],
    "meta": {
        "current_page": 1,
        "per_page": 30,
        "total": 1,
        "last_page": 1
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 402, 403, 429, 5XX (see Common error responses).

Create a program

POST `/programs`

Create a new program in the calling account. Only `program_name` is required; every other field has a sensible default.

Defaults

Fields you do not supply default to:

- `status` - `active`
- `data_retention_months` - `12`
- `event_grouping` - `separate` (with `max_separate_events` defaulting to 10)
- `report_emails` - empty
- `default_risk_level` - `low`

Idempotency

This endpoint accepts the `Idempotency-Key` header so that retries on network failure are safe. See the Idempotency section of the API guide for the full contract.

Header parameters

Field	Description
`Idempotency-Key`	**string [max 255 characters]** Optional client-generated key (typically a UUID) that lets you safely retry a write request without risk of duplicate work or duplicate billing. A retry with the same key and body returns the original response verbatim, with the `Idempotent-Replay: true` response header. Honoured on `POST` requests only. See the Idempotency section of the API guide for the full contract. **Example:** `8c4d3a18-2f63-4f9a-9f3a-9b1f5a7c2d4f`

Request body

The program details

Field	Description
`program_name` REQUIRED	**string [2..255 characters]** **Example:** `AML Onboarding Program`
`status`	**string** **Enum:** `active`, `paused` **Example:** `active`
`report_emails`	**array of strings [max 5 items] or null** Email addresses (max 5) that receive event report notifications

Field	Description
`data_retention_months`	**integer** **Enum:** `0` , `1` , `3` , `6` , `12` , `24` , `36` , `48` , `60` , `72` , `84` **Example:** `12`
`event_grouping`	**string** **Enum:** `separate` , `grouped` **Example:** `separate`
`max_separate_events`	**integer [5..100] or null** Required when `event_grouping` is `separate` ; ignored when `grouped` . Range 5-100. **Example:** `10`
`default_risk_level`	**string or null** The risk level pre-filled when a new entity is added to this program. When set to `null` (No risk assigned), new entities start with no risk rating, which may be left unassigned and set later; bulk file uploads assign `low` . **Enum:** `low` , `medium` , `high` **Example:** `low`

Sample request

```
{
    "program_name": "AML Onboarding Program",
    "status": "active",
    "report_emails": [
        "compliance@example.com"
    ],
    "data_retention_months": 12,
    "event_grouping": "separate",
    "max_separate_events": 10,
    "default_risk_level": "low"
}
```

Responses

201 **Program created** (application/json)

Field	Description
`data`	**object**
`data.` `uuid`	**string (uuid)** The program's UUID **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`

Field	Description
data. program_number	**string** A short obfuscated public identifier for the program (e.g. "P-123-456-789") **Example:** `P-123-456-789`
data. program_name	**string** The display name of the program **Example:** `AML Onboarding Program`
data. status	**string** Lifecycle status of the program: `active` - normal operation `paused` - monitors do not run; entities and checks remain accessible `archived` - read-only; the program is hidden from default lists but not deleted **Enum:** `active`, `paused`, `archived` **Example:** `active`
data. report_emails	**array of strings** Email addresses (max 5) that receive event report notifications for this program
data. data_retention_months	**integer** How long, in months, to keep entity data for. Permitted values: `0` (no retention), `1`, `3`, `6`, `12`, `24`, `36`, `48`, `60`, `72`, `84`. **Example:** `12`
data. event_grouping	**string** Whether events from the same entity should be grouped into one event: `separate` - each event is created as a separate event (capped by `max_separate_events`) `grouped` - events are merged into a single event per entity **Enum:** `separate`, `grouped` **Example:** `separate`
data. max_separate_events	**integer or null** When `event_grouping` is `separate`, the maximum number of separate events to create for a single entity (5-100). Null when `event_grouping` is `grouped`. **Example:** `10`

Field	Description
data. default_risk_level	**string or null** The risk level pre-filled when a new entity is added to this program: `low` / `medium` / `high` - new entities pre-fill with this risk rating `null` - new entities start with no risk assigned; the rating may be left unassigned and set later (bulk file uploads assign `low`) **Enum:** `low`, `medium`, `high` **Example:** `low`
data. created_at	**string (date-time)** ISO 8601 timestamp at which the program was created **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** ISO 8601 timestamp at which the program was last updated **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response

```
{
    "data": {
        "uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "program_number": "P-123-456-789",
        "program_name": "AML Onboarding Program",
        "status": "active",
        "report_emails": [
            "compliance@example.com"
        ],
        "data_retention_months": 12,
        "event_grouping": "separate",
        "max_separate_events": 10,
        "default_risk_level": "low",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```

409 **Conflict - a request with this Idempotency-Key is currently being processed** (application/json)

Field	Description
message	**string** **Example:** `A request with this Idempotency-Key is already being processed.`

Sample response

```
{
    "message": "A request with this Idempotency-Key is already being processed."
}
```

422 **Idempotency-Key was reused with a different request body** (application/json)

Field	Description
`message`	**string** **Example:** `Idempotency-Key was reused with a different request body.`

Sample response

```
{
    "message": "Idempotency-Key was reused with a different request body."
}
```

Standard error responses: 400, 401, 402, 403, 429, 5XX (see Common error responses).

Show a program

GET `/programs/{program_uuid}`

Returns a single program by its UUID. Unknown UUIDs return `404`.

Path parameters

Field	Description
`program_uuid` REQUIRED	**string (uuid)** The program UUID

Responses

200 **Program response** (application/json)

Field	Description
`data`	**object**
`data.` `uuid`	**string (uuid)** The program's UUID **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`

Field	Description
data. program_number	**string** A short obfuscated public identifier for the program (e.g. "P-123-456-789") **Example:** `P-123-456-789`
data. program_name	**string** The display name of the program **Example:** `AML Onboarding Program`
data. status	**string** Lifecycle status of the program: `active` - normal operation `paused` - monitors do not run; entities and checks remain accessible `archived` - read-only; the program is hidden from default lists but not deleted **Enum:** `active`, `paused`, `archived` **Example:** `active`
data. report_emails	**array of strings** Email addresses (max 5) that receive event report notifications for this program
data. data_retention_months	**integer** How long, in months, to keep entity data for. Permitted values: `0` (no retention), `1`, `3`, `6`, `12`, `24`, `36`, `48`, `60`, `72`, `84`. **Example:** `12`
data. event_grouping	**string** Whether events from the same entity should be grouped into one event: `separate` - each event is created as a separate event (capped by `max_separate_events`) `grouped` - events are merged into a single event per entity **Enum:** `separate`, `grouped` **Example:** `separate`
data. max_separate_events	**integer or null** When `event_grouping` is `separate`, the maximum number of separate events to create for a single entity (5-100). Null when `event_grouping` is `grouped`. **Example:** `10`

Field	Description
data. default_risk_level	**string or null** The risk level pre-filled when a new entity is added to this program: `low` / `medium` / `high` - new entities pre-fill with this risk rating `null` - new entities start with no risk assigned; the rating may be left unassigned and set later (bulk file uploads assign `low`) **Enum:** `low`, `medium`, `high` **Example:** `low`
data. created_at	**string (date-time)** ISO 8601 timestamp at which the program was created **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** ISO 8601 timestamp at which the program was last updated **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "program_number": "P-123-456-789",
        "program_name": "AML Onboarding Program",
        "status": "active",
        "report_emails": [
            "compliance@example.com"
        ],
        "data_retention_months": 12,
        "event_grouping": "separate",
        "max_separate_events": 10,
        "default_risk_level": "low",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 402, 403, 404, 429, 5XX (see Common error responses).

Update a program

PATCH `/programs/{program_uuid}`

Update an existing program using PATCH semantics: only the fields you supply are changed, everything else is left as-is. Pass `null` to clear an optional field.

Note: when changing `event_grouping` to `separate` you must also supply `max_separate_events`. Conversely, when changing it to `grouped`, any supplied `max_separate_events` is ignored and the stored value is cleared.

Path parameters

Field	Description
`program_uuid` REQUIRED	**string (uuid)** The program UUID

Request body

The program fields to update. All fields are optional.

Field	Description
`program_name`	**string [2..255 characters]**
`status`	**string** **Enum:** `active`, `paused`
`report_emails`	**array of strings [max 5 items] or null**
`data_retention_months`	**integer** **Enum:** `0`, `1`, `3`, `6`, `12`, `24`, `36`, `48`, `60`, `72`, `84`
`event_grouping`	**string** **Enum:** `separate`, `grouped`
`max_separate_events`	**integer [5..100] or null**
`default_risk_level`	**string or null** The risk level pre-filled when a new entity is added to this program. When set to `null` (No risk assigned), new entities start with no risk rating, which may be left unassigned and set later; bulk file uploads assign `low`. **Enum:** `low`, `medium`, `high`

Sample request

```
{
    "program_name": "string",
    "status": "active",
    "report_emails": [
        "user@example.com"
    ],
    "data_retention_months": 0,
```

```
    "event_grouping": "separate",
    "max_separate_events": 0,
    "default_risk_level": "low"
}
```

Responses

200 **Program updated** (application/json)

Field	Description
`data`	**object**
`data.` `uuid`	**string (uuid)** The program's UUID **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
`data.` `program_number`	**string** A short obfuscated public identifier for the program (e.g. "P-123-456-789") **Example:** `P-123-456-789`
`data.` `program_name`	**string** The display name of the program **Example:** `AML Onboarding Program`
`data.` `status`	**string** Lifecycle status of the program: `active` - normal operation `paused` - monitors do not run; entities and checks remain accessible `archived` - read-only; the program is hidden from default lists but not deleted **Enum:** `active`, `paused`, `archived` **Example:** `active`
`data.` `report_emails`	**array of strings** Email addresses (max 5) that receive event report notifications for this program
`data.` `data_retention_months`	**integer** How long, in months, to keep entity data for. Permitted values: `0` (no retention), `1`, `3`, `6`, `12`, `24`, `36`, `48`, `60`, `72`, `84`. **Example:** `12`

Field	Description
data. event_grouping	**string** Whether events from the same entity should be grouped into one event: `separate` - each event is created as a separate event (capped by `max_separate_events`) `grouped` - events are merged into a single event per entity **Enum:** `separate`, `grouped` **Example:** `separate`
data. max_separate_events	**integer or null** When `event_grouping` is `separate`, the maximum number of separate events to create for a single entity (5-100). Null when `event_grouping` is `grouped`. **Example:** `10`
data. default_risk_level	**string or null** The risk level pre-filled when a new entity is added to this program: `low` / `medium` / `high` - new entities pre-fill with this risk rating `null` - new entities start with no risk assigned; the rating may be left unassigned and set later (bulk file uploads assign `low`) **Enum:** `low`, `medium`, `high` **Example:** `low`
data. created_at	**string (date-time)** ISO 8601 timestamp at which the program was created **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** ISO 8601 timestamp at which the program was last updated **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "program_number": "P-123-456-789",
        "program_name": "AML Onboarding Program",
        "status": "active",
        "report_emails": [
            "compliance@example.com"
        ],
        "data_retention_months": 12,
        "event_grouping": "separate",
```

```
        "max_separate_events": 10,
        "default_risk_level": "low",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 402, 403, 404, 429, 5XX (see Common error responses).

Delete a program

DELETE `/programs/{program_uuid}`

Soft-deletes the program and every entity in it (and that entity's checks, ID checks, reports, events, notes and crossmatches). The records are no longer returned by list/show endpoints but are retained for compliance and audit purposes. The deleted program record is returned in the response as confirmation; a subsequent show or list call for it will return `404`.

See the Soft Deletion section of the API guide for the full policy (30-day recovery window, portal-only restore, what gets cascaded).

Path parameters

Field	Description
`program_uuid` REQUIRED	**string (uuid)** The program UUID

Responses

200 **Program deleted** (application/json)

Field	Description
`data`	**object**
`data.` `uuid`	**string (uuid)** The program's UUID **Example:** `9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
`data.` `program_number`	**string** A short obfuscated public identifier for the program (e.g. "P-123-456-789") **Example:** `P-123-456-789`
`data.` `program_name`	**string** The display name of the program **Example:** `AML Onboarding Program`

Field	Description
data. status	**string** Lifecycle status of the program: `active` - normal operation `paused` - monitors do not run; entities and checks remain accessible `archived` - read-only; the program is hidden from default lists but not deleted **Enum:** `active`, `paused`, `archived` **Example:** `active`
data. report_emails	**array of strings** Email addresses (max 5) that receive event report notifications for this program
data. data_retention_months	**integer** How long, in months, to keep entity data for. Permitted values: `0` (no retention), `1`, `3`, `6`, `12`, `24`, `36`, `48`, `60`, `72`, `84`. **Example:** `12`
data. event_grouping	**string** Whether events from the same entity should be grouped into one event: `separate` - each event is created as a separate event (capped by `max_separate_events`) `grouped` - events are merged into a single event per entity **Enum:** `separate`, `grouped` **Example:** `separate`
data. max_separate_events	**integer or null** When `event_grouping` is `separate`, the maximum number of separate events to create for a single entity (5-100). Null when `event_grouping` is `grouped`. **Example:** `10`
data. default_risk_level	**string or null** The risk level pre-filled when a new entity is added to this program: `low` / `medium` / `high` - new entities pre-fill with this risk rating `null` - new entities start with no risk assigned; the rating may be left unassigned and set later (bulk file uploads assign `low`) **Enum:** `low`, `medium`, `high` **Example:** `low`
data. created_at	**string (date-time)** ISO 8601 timestamp at which the program was created **Example:** `2025-01-01T00:00:00Z`

Field	Description
data. updated_at	**string (date-time)** ISO 8601 timestamp at which the program was last updated **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response

```
{
    "data": {
        "uuid": "9c3e0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "program_number": "P-123-456-789",
        "program_name": "AML Onboarding Program",
        "status": "active",
        "report_emails": [
            "compliance@example.com"
        ],
        "data_retention_months": 12,
        "event_grouping": "separate",
        "max_separate_events": 10,
        "default_risk_level": "low",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```

Standard error responses: 400, 401, 402, 403, 404, 429, 5XX (see Common error responses).

Relationships

Links between entities, including ultimate beneficial owners.

List an entity's relationships

GET `/entities/{entity_uuid}/relationships`

Returns the relationships involving the specified entity, where the entity is either the `from` or the `to` side of the link. The related entity can be in any program on your account. Results are paginated.

Each relationship is returned once with both `forward_label` and `inverse_label`, so you can present it from either entity's point of view. See the relationship list endpoint for the full shape.

Sorting

The `sort` query parameter accepts `created_at` (default: `-created_at`, newest first) or `updated_at`. Prefix with `-` for descending order.

Path parameters

Field	Description
`entity_uuid` REQUIRED	**string (uuid)** The entity UUID

Query parameters

Field	Description
`page`	**integer** The page of results to return, starting at 1. **Example:** `1`
`per_page`	**integer** The number of relationships per page (defaults to 30, max 500) **Example:** `30`
`sort`	**string** Field to sort by; prefix with `-` for descending order **Enum:** `created_at`, `-created_at`, `updated_at`, `-updated_at` **Example:** `-created_at`

Responses

200 **Relationships list response** (application/json)

Field	Description
`data`	**array of objects** An array of relationships
`data[].` `uuid`	**string (uuid)** The relationship's UUID **Example:** `7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`

Field	Description
data[]. relationship_type	**string** The type of relationship. **Enum:** `spouse_of`, `director_of`, `shareholder_of`, `trustee_of`, `beneficiary_of`, `parent_of`, `sibling_of`, `employee_of`, `officer_of`, `partner_in`, `owner_of`, `buyer_of`, `seller_of`, `associated_with` **Example:** `director_of`
data[]. forward_label	**string** Human-readable label when read from the `from` entity (e.g. "Director of"). **Example:** `Director of`
data[]. inverse_label	**string** Human-readable label when read from the `to` entity (e.g. "Has director"). **Example:** `Has director`
data[]. symmetric	**boolean** When `true` the relationship reads the same from both sides (e.g. "Spouse / de facto / domestic partner") and `forward_label` equals `inverse_label`. **Example:** `false`
data[]. from_entity_uuid	**string (uuid)** UUID of the entity the relationship is recorded from.
data[]. to_entity_uuid	**string (uuid)** UUID of the entity the relationship points to.
data[]. percentage_ownership	**number (float) or null** Ownership percentage (0-100) for relationship types that support it (such as `shareholder_of`, `partner_in` and `owner_of`). Null for other types, or when an ownership relationship's percentage is unknown. Drives ultimate beneficial owner tracing. **Example:** `40`
data[]. source	**string** How the relationship was recorded (e.g. `manual`, `workflow`, `report`, `kyb_discovery`, `import`). **Example:** `manual`
data[]. effective_from	**string (date) or null** Optional date from which the relationship is effective (YYYY-MM-DD). **Example:** `2024-01-01`

Field	Description
data[]. effective_to	**string (date) or null** Optional date the relationship is effective until (YYYY-MM-DD). Always null for point-in-time relationship types (`buyer_of` and `seller_of`), which record a single dated event in `effective_from` .
data[]. comment	**string or null** Optional free-text comment describing the relationship. **Example:** `Appointed at the 2024 AGM.`
data[]. created_at	**string (date-time)** **Example:** `2025-01-01T00:00:00Z`
data[]. updated_at	**string (date-time)** **Example:** `2025-01-01T00:00:00Z`
meta	**object**
meta. current_page	**integer** **Example:** `1`
meta. per_page	**integer** **Example:** `30`
meta. total	**integer** **Example:** `1`
meta. last_page	**integer** **Example:** `1`
api_reference	**string (uuid)**

Sample response


```
{
    "data": [
        {
            "uuid": "7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
            "relationship_type": "director_of",
            "forward_label": "Director of",
            "inverse_label": "Has director",
            "symmetric": false,
            "from_entity_uuid": "00000000-0000-0000-0000-000000000000",
            "to_entity_uuid": "00000000-0000-0000-0000-000000000000",
            "percentage_ownership": 40,
            "source": "manual",
            "effective_from": "2024-01-01",
            "effective_to": null,
            "comment": "Appointed at the 2024 AGM.",
```

```
            "created_at": "2025-01-01T00:00:00Z",
            "updated_at": "2025-01-01T00:00:00Z"
        }
    ],
    "meta": {
        "current_page": 1,
        "per_page": 30,
        "total": 1,
        "last_page": 1
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

Create a relationship on an entity

POST `/entities/{entity_uuid}/relationships`

Create a relationship FROM the entity in the path TO another entity on your account. The two entities can belong to different programs.

Direction is set by the order: the path entity becomes `from_entity_uuid` and the `to_entity_uuid` in the body becomes the `to` side. To record the relationship the other way around, send the request to the other entity instead.

A relationship of a given type is recorded once between two entities. For symmetric types (such as `spouse_of`) the same pair counts as one relationship regardless of direction; a repeat request returns `409`.

Idempotency

This endpoint accepts the `Idempotency-Key` header so that retries on network failure are safe. See the Idempotency section of the API guide for the full contract.

Path parameters

Field	Description
`entity_uuid` REQUIRED	**string (uuid)** The entity UUID (the `from` side of the relationship)

Header parameters

Field	Description

`Idempotency-Key`	**string [max 255 characters]** Optional client-generated key (typically a UUID) that lets you safely retry a write request without risk of duplicate work or duplicate billing. A retry with the same key and body returns the original response verbatim, with the `Idempotent-Replay: true` response header. Honoured on `POST` requests only. See the Idempotency section of the API guide for the full contract. **Example:** `8c4d3a18-2f63-4f9a-9f3a-9b1f5a7c2d4f`

Request body

The relationship details

Field	Description
`to_entity_uuid` REQUIRED	**string (uuid)** UUID of the entity the relationship points to. Must be on your account and different from the path entity.
`relationship_type` REQUIRED	**string** The type of relationship. Each type permits only certain entity types on the `from` and `to` sides (for example `director_of` links an individual to a business); a request that pairs incompatible entity types is rejected with a 422. Some types (such as `associated_with`) require a `comment` . **Enum:** `spouse_of` , `director_of` , `shareholder_of` , `trustee_of` , `beneficiary_of` , `parent_of` , `sibling_of` , `employee_of` , `officer_of` , `partner_in` , `owner_of` , `buyer_of` , `seller_of` , `associated_with` **Example:** `director_of`
`percentage_ownership`	**number (float)** Optional ownership percentage (0-100). Only meaningful for relationship types that support ownership (such as `shareholder_of` , `partner_in` , `owner_of`); ignored for other types. Drives ultimate beneficial owner tracing. **Example:** `40`
`effective_from`	**string (date)** Optional date the relationship is effective from (YYYY-MM-DD). **Example:** `2024-01-01`

Field	Description
effective_to	**string (date)** Optional date the relationship is effective until (YYYY-MM-DD). Must be on or after effective_from. Ignored (stored as null) for point-in-time relationship types (`buyer_of` and `seller_of`), which record a single dated event in `effective_from` . **Example:** `2025-01-01`
comment	**string [max 255 characters]** Free-text comment. Max 255 characters. Optional for most relationship types, but required for some (such as `associated_with`). **Example:** `Appointed at the 2024 AGM.`

Sample request

```
{
    "to_entity_uuid": "00000000-0000-0000-0000-000000000000",
    "relationship_type": "director_of",
    "percentage_ownership": 40,
    "effective_from": "2024-01-01",
    "effective_to": "2025-01-01",
    "comment": "Appointed at the 2024 AGM."
}
```

Responses

201 **Relationship created** (application/json)

Field	Description
data	**object** A directed link between two entities on your account - for example a person who is a director of a company, the spouse of another person, or the beneficial owner of a trust. The two entities can belong to any program on your account. A relationship is stored once, in the direction it was created (`from_entity_uuid` -> `to_entity_uuid`). Both labels are returned so you can present the link from either entity's point of view: read `forward_label` when looking from the `from` entity and `inverse_label` when looking from the `to` entity. When `symmetric` is `true` the two labels are identical and the direction carries no extra meaning. Deleting a relationship is a soft-delete: the record is kept for audit / compliance purposes and a subsequent list call no longer returns it.
data. uuid	**string (uuid)** The relationship's UUID **Example:** `7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`

Field	Description
data. relationship_type	**string** The type of relationship. **Enum:** `spouse_of`, `director_of`, `shareholder_of`, `trustee_of`, `beneficiary_of`, `parent_of`, `sibling_of`, `employee_of`, `officer_of`, `partner_in`, `owner_of`, `buyer_of`, `seller_of`, `associated_with` **Example:** `director_of`
data. forward_label	**string** Human-readable label when read from the `from` entity (e.g. "Director of"). **Example:** `Director of`
data. inverse_label	**string** Human-readable label when read from the `to` entity (e.g. "Has director"). **Example:** `Has director`
data. symmetric	**boolean** When `true` the relationship reads the same from both sides (e.g. "Spouse / de facto / domestic partner") and `forward_label` equals `inverse_label`. **Example:** `false`
data. from_entity_uuid	**string (uuid)** UUID of the entity the relationship is recorded from.
data. to_entity_uuid	**string (uuid)** UUID of the entity the relationship points to.
data. percentage_ownership	**number (float) or null** Ownership percentage (0-100) for relationship types that support it (such as `shareholder_of`, `partner_in` and `owner_of`). Null for other types, or when an ownership relationship's percentage is unknown. Drives ultimate beneficial owner tracing. **Example:** `40`
data. source	**string** How the relationship was recorded (e.g. `manual`, `workflow`, `report`, `kyb_discovery`, `import`). **Example:** `manual`
data. effective_from	**string (date) or null** Optional date from which the relationship is effective (YYYY-MM-DD). **Example:** `2024-01-01`

Field	Description
data. effective_to	**string (date) or null** Optional date the relationship is effective until (YYYY-MM-DD). Always null for point-in-time relationship types (`buyer_of` and `seller_of`), which record a single dated event in `effective_from` .
data. comment	**string or null** Optional free-text comment describing the relationship. **Example:** `Appointed at the 2024 AGM.`
data. created_at	**string (date-time)** **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "relationship_type": "director_of",
        "forward_label": "Director of",
        "inverse_label": "Has director",
        "symmetric": false,
        "from_entity_uuid": "00000000-0000-0000-0000-000000000000",
        "to_entity_uuid": "00000000-0000-0000-0000-000000000000",
        "percentage_ownership": 40,
        "source": "manual",
        "effective_from": "2024-01-01",
        "effective_to": null,
        "comment": "Appointed at the 2024 AGM.",
        "created_at": "2025-01-01T00:00:00Z",
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


409 **Conflict - either a request with this Idempotency-Key is currently being processed, or a relationship of this type already exists between these entities.** (application/json)

Field	Description
message	**string** **Example:** `A relationship of this type already exists between these entities.`

Sample response

```
{
    "message": "A relationship of this type already exists between these entities."
}
```

422 **Idempotency-Key was reused with a different request body** (application/json)

Field	Description
`message`	**string** **Example:** `Idempotency-Key was reused with a different request body.`

Sample response

```
{
    "message": "Idempotency-Key was reused with a different request body."
}
```

Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

Resolve an entity's ultimate beneficial owners

GET `/entities/{entity_uuid}/ubos`

Resolves the ultimate beneficial owners (UBOs) of a business entity by tracing its shareholding relationships (the relationship types that carry an ownership percentage) out to the natural persons behind it. Indirect ownership through other companies is followed and each person's stake is aggregated across every chain before the `threshold` is applied.

Every business that could not be fully traced - because it has no recorded shareholders, a shareholder with an unknown percentage, or recorded shareholdings that do not add up to 100% - is returned in `data_gaps` regardless of its stake, so the disclosure is complete. The `threshold` only filters the confirmed `ubos` list.

Only `business` entities can be resolved; for any other entity type `applicable` is `false` and both lists are empty.

Path parameters

Field	Description
`entity_uuid` REQUIRED	**string (uuid)** The entity UUID

Query parameters

Field	Description

`threshold`	**number (float)** The minimum ownership percentage for an owner to be returned (0-100, default 25). **Example:** `25`
`as_of`	**string (date)** Evaluate ownership as at this date (YYYY-MM-DD, defaults to today). **Example:** `2025-01-01`

Responses

200 **Ultimate beneficial owner analysis** (application/json)

Field	Description
`data`	**object** The ultimate beneficial owner (UBO) analysis for a business entity. Ownership is traced through the entity's shareholding relationships out to the natural persons behind it, with each person's stake aggregated across every chain before the threshold is applied. Every business that could not be fully traced is reported in `data_gaps`, regardless of its stake.
`data.` `applicable`	**boolean** False when the entity is not a business (UBO tracing only applies to businesses). **Example:** `true`
`data.` `threshold`	**number (float)** The minimum ownership percentage used to filter the `ubos` list. **Example:** `25`
`data.` `as_of`	**string (date)** The date the ownership was evaluated as at (YYYY-MM-DD). **Example:** `2025-01-01`
`data.` `fully_traced`	**boolean** True when there are no data-quality gaps. **Example:** `false`
`data.` `circular_detected`	**boolean** True when a loop was found in the ownership structure. **Example:** `false`
`data.` `ubos`	**array of objects** Confirmed beneficial owners at or above the threshold.

Field	Description
data. ubos[]. uuid	**string (uuid)** UUID of the beneficial owner entity. **Example:** `7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
data. ubos[]. name	**string** Name of the beneficial owner. **Example:** `Alice Smith`
data. ubos[]. entity_type	**string** The owner's entity type (typically `individual`). **Example:** `individual`
data. ubos[]. percentage	**number (float)** Aggregated direct and indirect ownership of the subject entity. **Example:** `30`
data. ubos[]. paths	**array of objects** The ownership chains contributing to this owner's stake.
data. ubos[]. paths[]. percentage	**number (float)** The ownership contributed by this chain. **Example:** `30`
data. ubos[]. paths[]. via	**array of objects** The intermediary businesses on this chain, from the subject outwards.
data. ubos[]. paths[]. via[]. uuid	**string (uuid)**
data. ubos[]. paths[]. via[]. name	**string** **Example:** `Coral Bay Investments Pty Ltd`
data. data_gaps	**array of objects** Businesses whose ownership could not be fully traced, regardless of stake.
data. data_gaps[]. uuid	**string (uuid)** UUID of the business whose ownership could not be fully traced. **Example:** `7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`

Field	Description
data. data_gaps[]. name	**string** Name of the business. **Example:** `Coral Bay Investments Pty Ltd`
data. data_gaps[]. percentage	**number (float)** The effective ownership of the subject entity that flows through this business (its total stake, traced or not). **Example:** `100`
data. data_gaps[]. untraced_percentage	**number (float)** The effective ownership of the subject entity that is actually unaccounted for via this business - the stake reaching it multiplied by its unresolved fraction. This is the figure shown in the UI and the most meaningful measure of the gap. **Example:** `40`
data. data_gaps[]. reason	**string** Why the business could not be fully traced - one of `no_ownership_data`, `unknown_percentage`, `incomplete_ownership`, `circular_ownership`, `max_depth_reached`. **Example:** `no_ownership_data`
data. data_gaps[]. remainder	**number (float) or null** The unresolved fraction (percentage) of this business's own ownership, where known. **Example:** `40`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "applicable": true,
        "threshold": 25,
        "as_of": "2025-01-01",
        "fully_traced": false,
        "circular_detected": false,
        "ubos": [
            {
                "uuid": "7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
                "name": "Alice Smith",
                "entity_type": "individual",
                "percentage": 30,
                "paths": [
                    {
                        "percentage": 30,
                        "via": []
                    }
```

```
                ]
            }
        ],
        "data_gaps": [
            {
                "uuid": "7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
                "name": "Coral Bay Investments Pty Ltd",
                "percentage": 100,
                "untraced_percentage": 40,
                "reason": "no_ownership_data",
                "remainder": 40
            }
        ]
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

Delete a relationship

DELETE `/relationships/{relationship_uuid}`

Soft-delete a relationship by its UUID. The record is hidden from the API (it is no longer returned by the list endpoint) but remains in the database during a recovery window. The deleted relationship is returned in the response as confirmation.

Soft deletion is not a long-term retention mechanism: a soft-deleted relationship can be restored within **30 days** (a portal-only action), after which it is eligible for permanent removal and may be purged without notice. If you need to retain a relationship for audit or compliance, do not delete it.

Any API key on your account can delete a relationship on the account. An unknown relationship UUID returns `404`.

See the Soft Deletion section of the API guide for the full policy (recovery window, portal-only restore).

Path parameters

Field	Description
`relationship_uuid` REQUIRED	string (uuid) The relationship UUID

Responses

200 **Relationship deleted (soft-delete - returned as confirmation)** (application/json)

Field	Description

`data`	**object** A directed link between two entities on your account - for example a person who is a director of a company, the spouse of another person, or the beneficial owner of a trust. The two entities can belong to any program on your account. A relationship is stored once, in the direction it was created (`from_entity_uuid` -> `to_entity_uuid`). Both labels are returned so you can present the link from either entity's point of view: read `forward_label` when looking from the `from` entity and `inverse_label` when looking from the `to` entity. When `symmetric` is `true` the two labels are identical and the direction carries no extra meaning. Deleting a relationship is a soft-delete: the record is kept for audit / compliance purposes and a subsequent list call no longer returns it.
`data.` `uuid`	**string (uuid)** The relationship's UUID **Example:** `7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f`
`data.` `relationship_type`	**string** The type of relationship. **Enum:** `spouse_of` , `director_of` , `shareholder_of` , `trustee_of` , `beneficiary_of` , `parent_of` , `sibling_of` , `employee_of` , `officer_of` , `partner_in` , `owner_of` , `buyer_of` , `seller_of` , `associated_with` **Example:** `director_of`
`data.` `forward_label`	**string** Human-readable label when read from the `from` entity (e.g. "Director of"). **Example:** `Director of`
`data.` `inverse_label`	**string** Human-readable label when read from the `to` entity (e.g. "Has director"). **Example:** `Has director`
`data.` `symmetric`	**boolean** When `true` the relationship reads the same from both sides (e.g. "Spouse / de facto / domestic partner") and `forward_label` equals `inverse_label` . **Example:** `false`
`data.` `from_entity_uuid`	**string (uuid)** UUID of the entity the relationship is recorded from.
`data.` `to_entity_uuid`	**string (uuid)** UUID of the entity the relationship points to.

data. percentage_ownership	**number (float) or null** Ownership percentage (0-100) for relationship types that support it (such as `shareholder_of`, `partner_in` and `owner_of`). Null for other types, or when an ownership relationship's percentage is unknown. Drives ultimate beneficial owner tracing. **Example:** `40`
data. source	**string** How the relationship was recorded (e.g. `manual`, `workflow`, `report`, `kyb_discovery`, `import`). **Example:** `manual`
data. effective_from	**string (date) or null** Optional date from which the relationship is effective (YYYY-MM-DD). **Example:** `2024-01-01`
data. effective_to	**string (date) or null** Optional date the relationship is effective until (YYYY-MM-DD). Always null for point-in-time relationship types (`buyer_of` and `seller_of`), which record a single dated event in `effective_from`.
data. comment	**string or null** Optional free-text comment describing the relationship. **Example:** `Appointed at the 2024 AGM.`
data. created_at	**string (date-time)** **Example:** `2025-01-01T00:00:00Z`
data. updated_at	**string (date-time)** **Example:** `2025-01-01T00:00:00Z`
api_reference	**string (uuid)**

Sample response


```
{
    "data": {
        "uuid": "7c1c0a8e-2b9f-4f0e-8d1a-1e2b3c4d5e6f",
        "relationship_type": "director_of",
        "forward_label": "Director of",
        "inverse_label": "Has director",
        "symmetric": false,
        "from_entity_uuid": "00000000-0000-0000-0000-000000000000",
        "to_entity_uuid": "00000000-0000-0000-0000-000000000000",
        "percentage_ownership": 40,
        "source": "manual",
        "effective_from": "2024-01-01",
        "effective_to": null,
        "comment": "Appointed at the 2024 AGM.",
        "created_at": "2025-01-01T00:00:00Z",
```

```
        "updated_at": "2025-01-01T00:00:00Z"
    },
    "api_reference": "00000000-0000-0000-0000-000000000000"
}
```


Standard error responses: 400, 401, 403, 404, 429, 5XX (see Common error responses).

Test

A connectivity check that confirms your credentials and base URL are correct.

Test endpoint

GET `/test`

Test endpoint which just returns a message indicating the API is connected and working.

Responses

200 **Test response** (application/json)

Field	Description
`message`	**string** A message indicating the result of the request. This will be `Ok` if the request was successful. **Example:** `Ok`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs and audit trail. **Example:** `123e4567-e89b-12d3-a456-426614174000`

Sample response


```
{
    "message": "Ok",
    "api_reference": "123e4567-e89b-12d3-a456-426614174000"
}
```


Standard error responses: 400, 401, 402, 403, 429, 5XX (see Common error responses).

Users

The portal users on the calling account.

List users

GET `/users`

List all users on the account.

Filters

The following filters can be applied to the users list:

enabled: Whether the user is enabled

Query parameters

Field	Description
`page`	**integer** The page number to return **Example:** `1`
`per_page`	**integer** The number of users per page (defaults to 30, max 500) **Example:** `30`
`enabled`	**boolean** Only return enabled users `true` - Only return enabled users `false` - Only return disabled users Note: The API will accept the string 'true' or the number 1 for true and the string 'false' or the number 0 for false as well as the boolean values. **Example:** `true`

Responses

200 **Users list response** (application/json)

Field	Description
`data`	**array of objects** An array of users
`data[].` `uuid`	**string (uuid)** The user's UUID **Example:** `123e4567-e89b-12d3-a456-426614174000`

Field	Description
data[]. username	**string** The user's username **Example:** `jenny.doe@example`
data[]. first_name	**string** The user's first name **Example:** `Jenny`
data[]. last_name	**string** The user's last name **Example:** `Doe`
data[]. email	**string** The user's email address **Example:** `jenny.doe@example.com`
data[]. phone	**string** The user's phone number **Example:** `0412345678`
data[]. enabled	**boolean** Whether the user is enabled **Example:** `true`
data[]. roles	**array of strings** The user's roles `account_manager` - Account Manager `security_manager` - Security Functions (update SAML settings and related security settings) `vpn_access` - VPN Access (Permits access from known VPN connections) Note: This will be an empty array if the user has no special roles assigned to them.
data[]. last_login_at	**string (date-time) or null** The user's last login timestamp (ISO 8601 format) **Example:** `2021-01-01T00:00:00Z`
data[]. created_at	**string (date-time)** The user's creation timestamp (ISO 8601 format) **Example:** `2021-01-01T00:00:00Z`

Field	Description
data[]. updated_at	**string (date-time)** The user's last update timestamp (ISO 8601 format) **Example:** `2021-01-01T00:00:00Z`
meta	**object** Metadata about the users list
meta. current_page	**integer** The current page number **Example:** `1`
meta. per_page	**integer** The number of users per page **Example:** `10`
meta. total	**integer** The total number of users **Example:** `1`
meta. last_page	**integer** The last page number **Example:** `10`
api_reference	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs and audit trail. **Example:** `123e4567-e89b-12d3-a456-426614174000`

Sample response


```
{
    "data": [
        {
            "uuid": "123e4567-e89b-12d3-a456-426614174000",
            "username": "jenny.doe@example",
            "first_name": "Jenny",
            "last_name": "Doe",
            "email": "jenny.doe@example.com",
            "phone": "0412345678",
            "enabled": true,
            "roles": [
                "account_manager"
            ],
            "last_login_at": "2021-01-01T00:00:00Z",
            "created_at": "2021-01-01T00:00:00Z",
            "updated_at": "2021-01-01T00:00:00Z"
```

```
        }
    ],
    "meta": {
        "current_page": 1,
        "per_page": 10,
        "total": 1,
        "last_page": 10
    },
    "api_reference": "123e4567-e89b-12d3-a456-426614174000"
}
```

Standard error responses: 400, 401, 402, 403, 429, 5XX (see Common error responses).

Show user

GET `/users/{user_uuid}`

Return a single user by UUID.

Path parameters

Field	Description
`user_uuid`	**string (uuid)** The user UUID **Example:** `123e4567-e89b-12d3-a456-426614174000`

Responses

200 **User response** (application/json)

Field	Description
`data`	**object** The user details
`data.` `uuid`	**string (uuid)** The user's UUID **Example:** `123e4567-e89b-12d3-a456-426614174000`
`data.` `username`	**string** The user's username **Example:** `jenny.doe@example`

Field	Description
data. first_name	**string** The user's first name **Example:** `Jenny`
data. last_name	**string** The user's last name **Example:** `Doe`
data. email	**string** The user's email address **Example:** `jenny.doe@example.com`
data. phone	**string** The user's phone number **Example:** `0412345678`
data. enabled	**boolean** Whether the user is enabled **Example:** `true`
data. roles	**array of strings** The user's roles `account_manager` - Account Manager `security_manager` - Security Functions (update SAML settings and related security settings) `vpn_access` - VPN Access (Permits access from known VPN connections) Note: This will be an empty array if the user has no special roles assigned to them.
data. last_login_at	**string (date-time) or null** The user's last login timestamp (ISO 8601 format) **Example:** `2021-01-01T00:00:00Z`
data. created_at	**string (date-time)** The user's creation timestamp (ISO 8601 format) **Example:** `2021-01-01T00:00:00Z`
data. updated_at	**string (date-time)** The user's last update timestamp (ISO 8601 format) **Example:** `2021-01-01T00:00:00Z`

Field	Description
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs and audit trail. **Example:** `123e4567-e89b-12d3-a456-426614174000`

Sample response


```
{
    "data": {
        "uuid": "123e4567-e89b-12d3-a456-426614174000",
        "username": "jenny.doe@example",
        "first_name": "Jenny",
        "last_name": "Doe",
        "email": "jenny.doe@example.com",
        "phone": "0412345678",
        "enabled": true,
        "roles": [
            "account_manager"
        ],
        "last_login_at": "2021-01-01T00:00:00Z",
        "created_at": "2021-01-01T00:00:00Z",
        "updated_at": "2021-01-01T00:00:00Z"
    },
    "api_reference": "123e4567-e89b-12d3-a456-426614174000"
}
```


Standard error responses: 400, 401, 402, 403, 429, 5XX (see Common error responses).

Update user

PATCH `/users/{user_uuid}`

Update a single user by UUID.

The update user record is returned in the response.

Parameters

The following parameters can be updated:

enabled: Whether the user is enabled

Path parameters

Field	Description

`user_uuid`	**string (uuid)** The user UUID **Example:** `123e4567-e89b-12d3-a456-426614174000`

Request body

The user details to update

Field	Description
`enabled`	**boolean** Update the user's enabled status `true` - The user is enabled `false` - The user is disabled Note: The API will accept the string 'true' or the number 1 for true and the string 'false' or the number 0 for false as well as the boolean values. **Example:** `true`

Sample request

```
{
    "enabled": true
}
```

Responses

200 **User response** (application/json)

Field	Description
`data`	**object** The user details after the update
`data.` `uuid`	**string (uuid)** The user's UUID **Example:** `123e4567-e89b-12d3-a456-426614174000`
`data.` `username`	**string** The user's username **Example:** `jenny.doe@example`
`data.` `first_name`	**string** The user's first name **Example:** `Jenny`

Field	Description
data. last_name	**string** The user's last name **Example:** `Doe`
data. email	**string** The user's email address **Example:** `jenny.doe@example.com`
data. phone	**string** The user's phone number **Example:** `0412345678`
data. enabled	**boolean** Whether the user is enabled **Example:** `true`
data. roles	**array of strings** The user's roles `account_manager` - Account Manager `security_manager` - Security Functions (update SAML settings and related security settings) `vpn_access` - VPN Access (Permits access from known VPN connections) Note: This will be an empty array if the user has no special roles assigned to them.
data. last_login_at	**string (date-time) or null** The user's last login timestamp (ISO 8601 format) **Example:** `2021-01-01T00:00:00Z`
data. created_at	**string (date-time)** The user's creation timestamp (ISO 8601 format) **Example:** `2021-01-01T00:00:00Z`
data. updated_at	**string (date-time)** The user's last update timestamp (ISO 8601 format) **Example:** `2021-01-01T00:00:00Z`
api_reference	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs and audit trail. **Example:** `123e4567-e89b-12d3-a456-426614174000`

Sample response

```
{
    "data": {
        "uuid": "123e4567-e89b-12d3-a456-426614174000",
        "username": "jenny.doe@example",
        "first_name": "Jenny",
        "last_name": "Doe",
        "email": "jenny.doe@example.com",
        "phone": "0412345678",
        "enabled": true,
        "roles": [
            "account_manager"
        ],
        "last_login_at": "2021-01-01T00:00:00Z",
        "created_at": "2021-01-01T00:00:00Z",
        "updated_at": "2021-01-01T00:00:00Z"
    },
    "api_reference": "123e4567-e89b-12d3-a456-426614174000"
}
```


Standard error responses: 400, 401, 402, 403, 429, 5XX (see Common error responses).

Common error responses

The following error responses are shared across all endpoints.

All error responses use the same JSON envelope: the `message` shown below, plus an `api_reference` UUID that uniquely identifies the request (quote it when contacting support). Validation failures (`400`) additionally include an `errors` object keyed by field name.

Code	Description	Message
`400`	Bad Request - The request is invalid	`Bad Request`
`401`	Unauthorized - the API key or account is inactive, expired, or not permitted from this IP address	`API Key Inactive`
`402`	Insufficient Credit - Insufficient credit to process this request	`Insufficient Credit`
`403`	No access - No permission to process this request	`No access`
`404`	Not Found - The requested resource does not exist or is not accessible by this account	`Entity not found`
`429`	Rate limited - The request rate limit has been reached	`Rate limited`
`5XX`	Internal error - Internal system error	`Internal error`

Rate limit headers

Header	Type	Description
`X-RateLimit-Limit`	integer	The maximum number of requests allowed in the current 60-second window.
`X-RateLimit-Remaining`	integer	The number of requests still available before the limit is reached. `0` on a 429.

Header	Type	Description
`Retry-After`	integer	Number of seconds to wait before the next request will be accepted. Use this value to schedule your retry rather than a fixed sleep. Only present on `429` responses.
`X-RateLimit-Reset`	integer	Unix timestamp (seconds since the epoch) at which the current rate-limit window resets. Only present on `429` responses.