Global Data API Documentation

Integration guides and endpoint reference for the Global Data API

Generated 8 September 2026

Table of Contents

API Documentation

- Globaldata Api
- Globaldata Check
- Asic
- Companies House
- Company Risk
- Dvs
- Idpass
- Idpass Validation
- Idpass Workflow
- Id Verification Custom Workflow

API Reference

- Address
- Adverse Media
- ASIC
- Business Check
- Business Intelligence
- Caspar
- Company Risk
- Court Check
- Deceased Check
- DVS
- Email
- Global Data Check
- ID Document
- ID Pass
- Likeness Check
- Liveness Check
- NZ Identity Verification
- Payroll and Super Check
- Person
- Phone
- Sanctions
- Social
- Test
- UK Companies House
- Common error responses
- Rate limit headers

API Documentation

Global Data API System

The Global Data API allows external programmatic access to the full functionality of the Global Data suite of validation, verification and search functions.

You will need to contact Global Data to obtain an API account and access to the API system. Once you have an account you will be able to access the API web interface and create API Keys for programmatic access.

The GDAPI portal

The Global Data API comes in two parts: the API itself, which your systems call, and the GDAPI portal at https://gdapi.globaldata.net.au, which is the web interface where you manage your account. This documentation site is public and does not require a login; the portal does.

The portal is where you:

- **Create and manage API keys.** Each key is tied to either the sandbox or the live environment, and can be restricted to particular IP addresses or subnets. Keys can be rotated or revoked at any time.
- **Manage users.** Add colleagues to your account, control what each of them can do, and remove them when they leave.
- **Review activity.** The audit log records what was called, when, by which key and user, and with what result, so you can trace any request your integration made.
- **Track usage and billing.** See how many calls each function has consumed, download invoices, and keep your billing details current.
- **Download customer-only material.** Some resources, such as the DVS sample identity documents used for sandbox testing, are provided in the portal rather than on this site.

You do not need the portal to make API calls. Once you have a key, everything your integration needs is in the API itself and in this documentation. Think of the portal as the place you go to set things up, and to look into what happened afterwards.

GLOBAL DATA

Access Overview

The API system is implemented as two parts. The first part is a web system, the GDAPI portal, which gives access to API key, user, billing and audit functions. The second part is the API itself which is a RESTful API interface with JSON payloads and uses standard HTTP methods and response codes.

API Web Access

The API web interface can be accessed from https://gdapi.globaldata.net.au. From here you will be able to manage your API account, keys and users, and review your usage and audit logs. Access to the web interface requires a valid API account.

Once logged in you will be able to access functions to:

- Manage your account
- Create additional users
- Create API access Keys
- View or update your billing details
- View your API usage and access logs

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.

Sandbox Access

You will initially provided with a sandbox access. This sandbox environment utilizes a separate test dataset which can be used for development and integration testing. The sandbox environment is identical to the production data environment and allows access to the API but does not incur any usage costs.

Sandbox API Endpoint

When using the sandbox environment, you will need to use the sandbox API endpoint: `https://sandbox-gdapi.globaldata.net.au/api/v2` . Attempting to use the production API endpoint will return a `HTTP/1.0 401 Unauthorized` status.

> **Important Note**: The sandbox environment does not contain real data! The test data consists of randomly generated names, addresses and associated details. Data from the sandbox environment must not be used in a production!

Authentication and Authorization

Overview

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

Generating an API Key

1. Visit the Global Data API web portal: https://gdapi.globaldata.net.au
2. Log in with your registered credentials.
3. Navigate to the **API Keys** section.
4. From here, you can generate one or multiple API keys as per your requirement.

Wen generating an API key, you can choose to use the sandbox environment or the live environment. Note that your account may initially be limited to the sandbox environment. Please contact Global Data to request access to the live environment.

Note: For added security, you have the option to restrict each API key's usage to specific IP addresses or subnets.

Using Your API Key

Once you have your API Key, include it in the request header when making API calls. The `Authorization` header format is as follows:

```
Authorization: Bearer <your-api-token>
```

Replace `<your-api-token>` with your actual API Key.

Handling Authentication Failures

If a request is made with an invalid, expired, or missing token, the API will respond with a `HTTP/1.0 401 Unauthorized` status. This indicates that the server understands the request but refuses to authorize it. When you receive such a response:

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

Access Endpoints

The API uses different endpoints for the sandbox and live environments.

Environment	Endpoint
Live	`https://gdapi.globaldata.net.au/api/v2`
Sandbox	`https://sandbox-gdapi.globaldata.net.au/api/v2`

> **Important Note**: Using the wrong endpoint (eg attempting to access the live environment with a sandbox key) will return a `HTTP/1.0 401 Unauthorized` status.

Billing and Account Management

Overview

The Global Data API operates on a pay-per-call model, 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 API 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 Global Data web portal.

Credit and Call Limits

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

1. **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.

2. **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 `HTTP/1.0 402 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.

Payment Failures and Resolution

In the event that a charge fails due to reasons like expired payment methods or insufficient funds, you will be notified. It is essential to resolve these issues promptly to prevent service disruptions.

If you encounter a `HTTP/1.0 402 Payment Required` response and believe it to be an error, please contact customer support or your account manager for further assistance.

Best Practices

- **Monitor API usage**: Stay informed about which endpoints are being accessed most frequently and consider optimizing your application to minimize costs.

- **Plan Ahead**: If you anticipate a spike in API usage, notify your Global Data account manager in advance to discuss potential solutions or rate adjustments.

Following these guidelines will ensure seamless interaction with the Global Data API and help in managing your costs effectively.

Rate Limiting

Overview

To ensure equitable access and maintain performance for all users, the Global Data 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 set to **600 requests per minute**. This is calculated using a rolling average method, where the number of requests is averaged over a 1-minute interval.

Adjusting Your Rate Limit

If there are operational needs that necessitate 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:

1. Contact Global Data support.
2. 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 surpassed, the API will respond with a `HTTP/1.0 429 Too Many Requests` status. In such scenarios, applications should implement a back-off strategy, waiting for a reasonable duration before retrying.

The following response headers provide additional details on rate limits:

- **x-ratelimit-limit**: The maximum number of allowable requests in the current time window (e.g., per minute).
- **x-ratelimit-remaining**: Indicates the number of requests left within the current time window.

For example:

```
x-ratelimit-limit: 600
x-ratelimit-remaining: 599
```

The above headers indicate a limit of 600 requests per minute, with 599 requests still available for use.

Best Practices

- **Implement Rate Limit Handling**: Always incorporate rate limit error handling in your application logic. This can be done by checking the HTTP status and parsing the provided headers to decide when to retry.

- **Spread Out Requests**: If feasible, distribute your API requests evenly over time to avoid hitting rate limits during peak usage times.

- **Monitor Usage**: Use the provided rate limit headers to monitor your application's API usage and adjust its behavior accordingly.

Following these guidelines will ensure a smooth interaction with the Global Data API and prevent unnecessary disruptions due to rate limiting.

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 Global Data 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://gdapi.globaldata.net.au/api/v2/resource-endpoint
```

In the above example, `v2` indicates that the API endpoint belongs to version 2.

Changes & Backward Compatibility

1. **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.

2. **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 2**. Ensure to check the version in the endpoint URL to guarantee you're accessing the correct version.

Version Support Lifecycle

Every version of the Global Data 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 3 is launched, version 2 will continue to be supported for at least another 12 months.

Transitioning Between Versions

It's advisable for developers to:

1. Regularly check for announcements on new API versions.
2. Test their applications against the new version during the support overlap period.
3. 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 Global Data API and easily navigate between versions.

Data Retention and Footprints

Introduction

The Global Data API prioritizes data privacy and protection. This section details our policies regarding the retention, storage, and access of the data communicated through our API.

> **Important Note**: No Personally Identifiable Information (PII) sent to Global Data as part of an API request is stored or retained unless it's explicitly necessary for processing the request.

Data Retention Policy

What We Retain:

1. **Query Data**: This encompasses the data necessary to process the requested query. For instance, a "Person Search" might involve PII like names, addresses, and phone numbers.
2. **Return Data**: Data sent back in response to a query, which might also contain PII.
3. **Local Data**: Details from the local web browser or software agent making the request. Examples include browser versions, operating systems, and screen resolutions.
4. **Network Data**: Captured from internet connections to our API services, primarily the IP address of the requester.

While we retain various data types, PII within Query Data is never stored or retained.

How We Store Data:

- **Local and Network Data**: Stored in server logs and our service audit database.
- **Query Data**: Stripped of any PII and then stored in the service audit database.
- **Return Data**: Contains PII already known to Global Data and stored in our service audit database.

Security Measures:

- All data, both in transit and at rest, is encrypted using industry-standard methodologies.
- Access to stored data adheres strictly to the Global Data IT policy, following the principle of least privilege.

Why We Store Data:

Data storage primarily caters to:

1. **Historic Review**: Enables users to revisit their past API queries or searches.
2. **Audit Purposes**: For refining service performance or addressing service hitches.
3. **Regulatory Compliance**: Meeting requirements like the Australian Privacy Act and Australian Privacy Principles.
4. **Specific Processing**: Particularly when data is added for future access purposes, e.g., in datawashes.

Data Retention Duration

The period for which data is retained varies:

- **Query Data**: Retained indefinitely from the capture time, stripped of any PII.
- **Return Data**: Retained indefinitely post initial capture.
- **Network and Local Data**: Kept indefinitely if they're devoid of PII.

Data Erasure

Once the retention period is fulfilled, data is systematically purged and obliterated, aligning with industry best practices and the Global Data's Data Lifecycle policy.

Access Footprint Policy

It's crucial to highlight that Global Data is not a credit bureau or assessment entity. Therefore, we don't create access footprints or access records on a data record that's discernible to end-users or record proprietors, beyond the scope described in this documentation.

Maintenance Windows

To ensure that the Global Data 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 adhere to the AEST timezone (UTC+10).

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:

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

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

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

Recommendations for API Users:

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

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

3. **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.

Sandbox Responses

In the Global Data API, the sandbox environment is a separate test dataset that can be used for development and integration testing. The test account is identical to the production data account and allows access to the API web portal and API but does not incur any usage costs.

PII Handling

Some of the Global Data API endpoints return PII (Personally Identifiable Information). This data is sensitive and as such requires adherence to the Global Data Privacy Policy and terms of service.

You will need to ensure that you have the appropriate privacy policy and PII handling processes in place before being granted access to these API endpoints.

IP Whitelisting for Webhooks

Some API endpoints send webhooks, HTTP callbacks that notify your application in real time when certain events occur (for example, when a verification completes or a status changes). To ensure your system can receive these webhook requests, you may need to whitelist the source IP addresses in your firewall or network configuration.

All webhook requests will originate from one of the following IP addresses:

- 13.236.17.73
- 3.24.198.127
- 52.65.32.72

Make sure your application accepts incoming HTTPS requests from these addresses to avoid missing any webhook notifications.

Global Data Check

Global Data Check matches a person against the **Global Data Universe** and returns a per-field breakdown of how well the supplied details line up with one or more candidate records.

It is a single endpoint - `POST /globaldata_check` - that supports a wide range of identity verification, contact validation, and lookup workflows. The formal request and response schema (with every field, default, and enum) lives in the Global Data Check API reference; this guide focuses on **how to use it** and walks through worked sandbox examples for each common use case.

The check runs against the Global Data Universe, an in-house dataset of person and contact records aggregated from many sources.

The response is **match information only - no PII is returned.** Every field in `match_results` is a status flag (`match` , `no_match` , `match_year` , etc.) describing how the supplied input compares to a candidate. The candidate's underlying details (their actual DOB, phone, address, etc.) are never sent back. This makes the check a clean choice for confirmation flows where you already hold the data you are checking and just need a yes/no/partial signal.

The check is also **very fast** and is suitable for high-volume bulk processing - for example, scoring an entire customer database overnight or running real-time checks at signup throughput:

Percentile	Response time
p50	180 ms
p75	550 ms

Per-request times vary with how much narrowing the supplied identifiers provide - a request with a phone, email, DOB, or full address resolves on the fast path; broad searches against very common last names with no other discriminator make up the slower tail.

Common things you can do with it:

- Confirm a person exists with a given name and date of birth (KYC / identity verification).
- Confirm that a phone number or email belongs to a particular person (signup flows, fraud checks).
- Verify someone lives at a given address, with full or partial address inputs.
- Find candidate matches for a partially-known identity to support manual review queues.
- Match against typos, nicknames, and reversed dates of birth without a separate fuzzy-matching pipeline.
- Score the match confidence and use it to drive auto-approve / manual review / reject routing.

How the check works

Request flow

The minimum request supplies a `last_name` . Everything else is optional - first name, middle name, date of birth, address, up to four phone numbers, and up to four email addresses. Whichever fields you supply are evaluated against each candidate; the

response tells you which ones matched.

The high-level flow is:

1. Filter the universe on `last_name`; supplied identifiers (DOB, address, phone, email) seed additional indexed lookups so candidates that match on a strong identifier are still considered even if their last-name spelling differs from the supplied one.
2. Score every candidate that comes back against the supplied fields.
3. Return either the single best candidate, or every candidate ordered best-first if `return_multiple_candidates: true` was supplied.

Response shape

`match_results` is **always an array**. By default it contains the single best candidate (length 0 if no candidate was found, length 1 otherwise). Set `return_multiple_candidates: true` to receive multiple candidates ordered best-first.

When `return_multiple_candidates: true` is set, the response is capped at the top **5 candidates by score**. If your search matches more than 5 records, only the strongest 5 are returned; the weaker tail is dropped. Narrow your request (add a DOB, a phone, or an address) to lift the strongest candidates above the cap.

Two universe records that produce **identical per-field outcomes** against your request collapse into a single row, so the count of rows in `match_results` reflects the number of distinct outcome maps, not the count of underlying records considered. See the Multi-candidate review use case for details.

Each candidate row only includes the fields that were actually evaluated. Fields the caller did not supply are **omitted from the row entirely** (they are not returned as `null` or any placeholder value). The address block is treated as a single unit: when no address was supplied, the `address` rollup and the four `street_address` / `suburb` / `state` / `postcode` components are all omitted together.

Match outcomes

First name and middle name

For **first_name** and **middle_name** the response uses a graded ladder:

Label	Meaning
`match`	Exact equality after normalisation (case- and diacritic-insensitive).
`alias_match`	The two names share a known nickname / alias group (Bob/Robert, Jenny/Jennifer, Tony/Antonio).
`partial_match`	One side is exactly a single-letter initial that matches the leading letter of the other side (for example `P` vs `Peter`). Modelled on universe records that only have a first-letter initial captured for the first name. Longer-form abbreviations like `Pete` vs `Peter` are not partial matches - they are handled by the `alias_match` ladder instead.
`fuzzy_match`	The two names share a phonetic (metaphone) code and the same leading letter.
`no_match`	None of the above.
`no_record`	(Middle name only.) The caller supplied a middle name but the candidate has none on file.
`not_used`	The caller supplied this field but it was dropped because it failed validation and `ignore_errors=true`.

Which match types are accepted is controlled per field via `first_name_matching`, `middle_name_matching`, and `last_name_matching` (each defaults to `["exact"]`).

Last name

`last_name` uses a smaller subset of the ladder:

Label	Meaning
`match`	Exact equality after normalisation.
`fuzzy_match`	The two last names share a metaphone code and the same leading letter. Only returned when `last_name_matching: ["exact","fuzzy"]` is set.
`no_match`	Neither of the above.
`not_used`	Dropped because it failed validation and `ignore_errors=true`.

Date of birth

`dob` has its own ladder of partial outcomes for cases where the supplied date is close but not identical to the candidate's:

Label	Meaning
`match`	Exact match on year, month, and day.
`match_day_month_reversal`	Year matches; the supplied day and month are the same as the candidate's month and day swapped (e.g. supplied `1989-12-08` vs candidate `1989-08-12`).
`match_year`	Only the year matches; day and month differ and are not a reversal.
`no_record`	The candidate has no DOB on file.
`no_match`	Year does not match.
`not_used`	Dropped because it failed validation and `ignore_errors=true`.

Address

`address` is a rollup summary derived from the four supplied address components (`street_address`, `suburb`, `state`, `postcode`). Each component returns `match` / `no_match` independently and they are returned alongside the rollup so you can see exactly which parts contributed.

Label	Meaning
`match`	All four supplied address components match the candidate.
`match_street`	Same street number on the same street name (street-type variations like `ST` vs `RD` are allowed) but at least one of suburb / state / postcode does not match.
`match_locality`	The street did not match, but state matches (when supplied) and at least one of suburb or postcode also matches.
`no_match`	None of the above.
`not_used`	Dropped because it failed validation and `ignore_errors=true`.

Whichever address input form was used (`gnaf_id` , `full_address` , or the four address parts), the address is resolved to its canonical components before evaluation, so the same rollup applies regardless of how it was supplied.

A person commonly has more than one address on file (a current address plus past addresses), but `match_results` only ever returns **one row per person**. Internally, every address on file is scored against the supplied address and the **best-matching** address is the one whose outcome appears in the response. So if a person has a current address that is an exact `match` and an older address that is only a `match_locality` , you will see `address: match` in the response - never both. Ties on rollup outcome are broken by preferring the most recently active address (largest `date_end`).

Phone and email

Phone and email are evaluated **per slot**. Each supplied slot (`phone` , `phone2` , `phone3` , `phone4` , and the same four `email` slots) returns its own outcome in the response. Slots that were not supplied are omitted.

Label	Meaning
`match`	The supplied value for this slot matches at least one of the candidate's phones / emails on file.
`no_match`	The supplied value for this slot does not match any of the candidate's phones / emails on file.
`not_used`	Dropped because it failed validation and `ignore_errors=true` .

So if a request supplies `phone` and `phone3` , the response carries two independent outcomes (`phone` and `phone3`); `phone2` and `phone4` are omitted. Each slot is evaluated against the candidate's full phones (or emails) list, so the slot order in the request does not matter for matching.

Best-candidate selection

Candidates are ranked in three steps, applied in order.

1. **Strong-discriminator hits rank first.** A candidate that produces an exact match against any of `phone` , `email` , full `dob` , or the full `address` (the rollup at `match`) outranks any candidate that does not, regardless of how well the other candidate scores on names alone. `dob: match_day_month_reversal` also counts as a strong-discriminator hit, **but only when the candidate also has `first_name: match` (or `alias_match`) AND `last_name: match`** - without name corroboration, a reversed DOB is treated as a numeric coincidence and is not enough to elevate the candidate into the strong-discriminator group.
2. **Exact last-name wins within the strong-discriminator group.** Among candidates that tied on step 1, candidates with `last_name: match` rank above candidates with `last_name: fuzzy_match` - a fuzzy last-name candidate never displaces an exact last-name candidate for the top slot.
3. **Weighted score breaks remaining ties.** Per-field weights are summed and the highest sum wins.

Field	match	partial outcomes
`first_name`	20	`alias_match` 16, `partial_match` 10, `fuzzy_match` 6
`middle_name`	1	`alias_match` 1, `partial_match` 1
`last_name`	12	`fuzzy_match` 8
`dob`	10	`match_day_month_reversal` 7, `match_year` 4
`address`	8	`match_street` 5, `match_locality` 3
`phone`	4	-

Field	match	partial outcomes
`email`	2	-

This calibration deliberately lets multiple identifiers outrank a stronger single name match. For example, within the strong-discriminator group, a candidate with `first_name: alias_match` (16) + `dob: match` (10) + `phone: match` (4) = 30 will beat a candidate with `first_name: match` (20) + `phone: match` (4) = 24.

Sandbox data

Every example in the rest of this guide uses the following sandbox records, available in the sandbox environment for free testing:

First Name	Middle Name	Last Name	Birth Date	Address	Phone	Email
JOHN	ANDREW	SMITH	1998-03-21	20 HARDY ST, LILYDALE 3140 VIC	0399999999, 0491222111	jsmith1998@example.com
ROBERT	PATRICK	BROWN	1971-03-18	8 WALTHAM ST, RICHMOND 3121 VIC	0399998888	
MARY	SALLY	JONES	1989-08-12	35 YARALLA ST, CONCORD WEST 2138 NSW	0299995000, 0491222444	mary_jones89@example.com
MARION		FILEWOOD	1955-09-07	375 ARGENT ST, BROKEN HILL 2880 NSW	0412024869, 0880885999	m47b7@rev-zone.net
AARON	ALBERT	FILEWOOD	1949-03-11	375 ARGENT ST, BROKEN HILL 2880 NSW	0880885999	
RUTH		WOLFE	2000-05-20	375 ARGENT ST, BROKEN HILL 2880 NSW	0455963579	
MATTHEW		SMITH	1999-10-21	(no address on file)	(no phone on file)	(no email on file)
JOAN		EVANS	1996-07-06	(no address on file)	(no phone on file)	(no email on file)
ROBERT		BROWN	(no DOB on file)	6 WATERVIEW CL, PORT MACQUARIE 2444 NSW	0491222333	
R		BROWN	(no DOB on file)	22 BRABYN ST, WINDSOR 2756 NSW	(no phone on file)	RP_BROWN71@EXAMPLE.COM

Sandbox calls are free and do not affect billing.

Use cases

Identity verification (KYC-light)

The most common use case: confirm a person exists with the supplied first name, last name, and date of birth. Often used as a low-touch identity check on its own, or as a pre-screen before launching a full DVS document verification.

Request:

```
{
  "first_name": "John",
  "middle_name": "Andrew",
  "last_name": "Smith",
  "birth_date": "1998-03-21"
}
```

Response:

```
{
  "message": "Ok",
  "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
  "match_results": [
    {
      "first_name": "match",
      "middle_name": "match",
      "last_name": "match",
      "dob": "match"
    }
  ]
}
```

What this tells you: there is a person named John Andrew Smith with date of birth 21 March 1998 in our universe. All four supplied fields matched cleanly. If the user had got the DOB wrong, `dob` would come back as `no_match` (or `match_year` / `match_day_month_reversal` if it was close - see the DOB partial matches example).

Phone tied to person (signup confirmation)

During signup, you've collected a name and a phone number from the user. You want to confirm that the phone number is actually associated with that person in our universe before trusting it for SMS-based 2FA or password recovery.

Request:

```
{
  "first_name": "John",
  "last_name": "Smith",
  "phone": "0399999999"
}
```

Response:

```
{
  "message": "Ok",
  "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
```

```
  "match_results": [
    {
      "first_name": "match",
      "last_name": "match",
      "phone": "match"
    }
  ]
}
```

If the phone did not belong to a John Smith on file, the best candidate would still come back, but with `phone: no_match`. That means the name was found but the phone is not associated with it - a useful signal for fraud-prevention workflows.

Phone numbers are accepted in either Australian `0NSN` form (10 digits starting with `0`) or the equivalent E.164 form (`+61` followed by 9 digits). The leading `+` on the E.164 form is optional - `61400123456` is treated the same as `+61400123456`, which avoids spurious rejections of numbers that have round-tripped through Excel / CSV imports that silently strip the `+`. All three shapes are normalised internally before comparison, so it does not matter which one your application stores.

Email tied to person (signup confirmation)

The same pattern as the phone use case, but using an email address instead.

Request:

```
{
  "first_name": "Mary",
  "last_name": "Jones",
  "email": "mary_jones89@example.com"
}
```

Response:

```
{
  "message": "Ok",
  "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
  "match_results": [
    {
      "first_name": "match",
      "last_name": "match",
      "email": "match"
    }
  ]
}
```

Multi-contact verification

Customers often have several contact points on file (current and old phone numbers, work and personal email addresses). You can supply up to four phones (`phone` , `phone2` , `phone3` , `phone4`) and up to four emails (`email` , `email2` , `email3` , `email4`) on a single request.

Each slot is evaluated **independently** against the candidate's full set of phones / emails, so the response tells you exactly which of your supplied values matched. Slots you did not supply are omitted from the row.

Request:

```
{
  "first_name": "John",
  "last_name": "Smith",
  "birth_date": "1998-03-21",
  "phone": "0399999999",
  "phone3": "0491222111",
  "email": "jsmith1998@example.com"
}
```

Response:

```
{
  "message": "Ok",
  "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
  "match_results": [
    {
      "first_name": "match",
      "last_name": "match",
      "dob": "match",
      "phone": "match",
      "phone3": "match",
      "email": "match"
    }
  ]
}
```

Notice that the request used `phone` and `phone3` (skipping `phone2`); the response mirrors that and contains a `phone` outcome and a `phone3` outcome with no `phone2` or `phone4` keys. The order does not matter for matching - each slot is checked against the candidate's complete phones list - so it is fine to use whichever slots best line up with how your application stores its data.

If a slot does not match (for example, the customer gave you an old phone number that is no longer on file), that slot will return `no_match` while the other slots can still return `match`.

Address verification (full match)

Confirm that a person lives at a specific address. Address can be supplied in **any one** of three forms (mutually exclusive):

- **Address parts** - `street_address` + `suburb` + `state` + `postcode`, all required together.
- **Full address** - a single `full_address` string that we parse for you.
- **GNAF id** - the canonical `gnaf_id` for the property.

Whichever form you use, the address is resolved to its canonical components before evaluation, so the same `address` rollup logic applies.

Using address parts

Request:

```
{
  "first_name": "John",
  "last_name": "Smith",
  "street_address": "20 Hardy St",
  "suburb": "Lilydale",
  "state": "VIC",
```

```
  "postcode": "3140"
}
```

Response:

```
{
  "message": "Ok",
  "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
  "match_results": [
    {
      "first_name": "match",
      "last_name": "match",
      "address": "match",
      "street_address": "match",
      "suburb": "match",
      "state": "match",
      "postcode": "match"
    }
  ]
}
```

Using full_address

Request:

```
{
  "first_name": "John",
  "last_name": "Smith",
  "full_address": "20 Hardy St, Lilydale VIC 3140"
}
```

The response shape is identical - the `full_address` is parsed and resolved to the same canonical components, then matched against the candidate.

Using gnaf_id

Request:

```
{
  "first_name": "John",
  "last_name": "Smith",
  "gnaf_id": "GAVIC421647320"
}
```

Same response shape. Use this form when you already hold the canonical GNAF id (for example from a prior address validation call).

GNAF persistent identifiers come in two shapes depending on the state of issue. Some states (ACT, NSW, VIC, QLD, WA, TAS) emit them with no separator between the state prefix and the digit run (`GAVIC421647320` , `GANSW710277749`). NT and SA emit them with a single underscore between the prefix and the digits (`GANT_717247498` , `GASA_414917272`). Both forms are accepted - keep the underscore as it appears in the source id rather than pre-stripping it.

Partial address lookup

Sometimes you only have part of an address, or the user has supplied a slightly different address than what is on file. The `address` rollup tells you at a glance how good the match is, and the four component fields tell you exactly which parts agreed.

"Same street, different suburb" - `match_street`

Suppose your records show John Smith at `20 Hardy Street, Doncaster VIC 3108` but our universe has him at Lilydale 3140. The street number and name match, but everything else is wrong.

Request:

```
{
  "first_name": "John",
  "last_name": "Smith",
  "street_address": "20 Hardy St",
  "suburb": "Doncaster",
  "state": "VIC",
  "postcode": "3108"
}
```

Response:

```
{
  "message": "Ok",
  "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
  "match_results": [
    {
      "first_name": "match",
      "last_name": "match",
      "address": "match_street",
      "street_address": "match",
      "suburb": "no_match",
      "state": "match",
      "postcode": "no_match"
    }
  ]
}
```

The rollup `match_street` says "we found someone of this name on the same street, but not at the same locality" - useful for nudging a user to double-check their address details rather than rejecting outright.

"Is there a John Smith in postcode 3140?" - `match_locality`

If you only know the locality, supply just the suburb / state / postcode. Even with a placeholder street address, the rollup will tell you whether anyone of that name lives in that locality.

Request:

```
{
  "first_name": "John",
  "last_name": "Smith",
  "street_address": "1 Unknown St",
  "suburb": "Lilydale",
  "state": "VIC",
  "postcode": "3140"
```

```
}
```

Response:

```
{
  "message": "Ok",
  "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
  "match_results": [
    {
      "first_name": "match",
      "last_name": "match",
      "address": "match_locality",
      "street_address": "no_match",
      "suburb": "match",
      "state": "match",
      "postcode": "match"
    }
  ]
}
```

`match_locality` requires `state` to match (when supplied) and at least one of `suburb` or `postcode` to also match. Use this when you want a "yes there's a John Smith in this area" confidence signal rather than an exact-address check.

Multi-candidate review

Set `return_multiple_candidates: true` to receive up to 5 candidates ordered best-first.

When to use it

Default single-candidate mode is the right choice for high-confidence yes/no checks: "is this person in the universe?", "does this phone belong to this person?". The single best candidate carries the per-field outcomes you need to drive an auto-approve / manual-review / reject decision, and the response stays small and easy to consume.

Switch to multi-candidate mode when one of the following applies:

- **Building a manual review queue.** Reviewers benefit from seeing several plausible candidates side-by-side so they can pick the right one rather than blindly trusting the top pick.
- **Building a "did you mean...?" disambiguation step.** When a user has typed something that could match more than one record (a common surname with a nickname, a postcode-only locality lookup), surfacing the alternates lets the user resolve the ambiguity themselves.
- **Hedging against ranker disagreement.** When the request supplies several identifiers that point in different directions (the user's phone belongs to Person A, but their name and DOB match Person B), the top candidate reflects the ranker's calibration. Multi-candidate mode lets you see both Person A and Person B and apply your own business rules to choose between them.
- **Investigating an unexpected outcome.** When a downstream workflow flagged a record and you want to understand "why didn't the right person win?", inspecting the full top-5 is the fastest way to see what else was considered.
- **Auditing or sampling.** When validating a new data feed or comparing universe coverage against an internal source, seeing all distinct candidates per query gives a clearer picture than the single best.

For straight pass/fail verification flows, leave it off - the single-candidate response is faster to handle and conveys the same yes/no signal.

Response shape and dedupe behaviour

`match_results` always returns **one row per matched person** - it never duplicates the same person once per address on file. A person with three known addresses still appears as a single row, with the address outcome reflecting the strongest match across all of them (see Address above for the best-address selection rules).

A subtle but important behaviour: **identical rows are collapsed.** If two underlying universe records produce the same per-field outcome map against your request (for example, two Roberts at the same address with the same DOB, both evaluated against your supplied `first_name + last_name + birth_date`), they appear in the response as a single row, not two. The number of rows in `match_results` therefore depends on how many *distinct* outcome maps your request produced, not on how many universe records were considered.

The practical implication: a request like `{ "last_name": "Brown", "return_multiple_candidates": true }` returns just a single row with `last_name: match`, because every Brown candidate evaluates identically against a request that only supplies a last name. The response cannot tell you how many Browns are in the universe; it can only tell you that at least one exists.

To get multiple distinct rows, supply additional identifiers so that the candidates differ in their outcomes. For example, supplying a `first_name` lets candidates with different first names produce different `first_name` outcomes (`match` vs `no_match`), so they appear as separate rows.

Request:

```
{
  "first_name": "Robert",
  "last_name": "Brown",
  "first_name_matching": ["exact", "alias", "partial"],
  "return_multiple_candidates": true
}
```

Response (the sandbox has Robert Patrick Brown, a Robert Brown with no DOB, and an `R BROWN` record):

```
{
  "message": "Ok",
  "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
  "match_results": [
    {
      "first_name": "match",
      "last_name": "match"
    },
    {
      "first_name": "partial_match",
      "last_name": "match"
    }
  ]
}
```

The two `Robert Brown` records (with and without a DOB on file) collapse into the first row because they evaluate identically against this request; the `R BROWN` record produces a distinct second row because its first name only matches partially. Each row in `match_results` is independent and self-describing, so you can route them straight into a review UI.

Typo tolerance with fuzzy matching

When your data may contain spelling variations of names (Mathew/Matthew, Smith/Smyth, Stephen/Steven), opt in to fuzzy matching for the affected name field. Fuzzy matching uses a phonetic (metaphone) algorithm and additionally requires the same

leading letter, so it is robust against typos while still being conservative.

Request:

```
{
  "first_name": "Mathew",
  "last_name": "Smith",
  "birth_date": "1999-10-21",
  "first_name_matching": ["exact", "fuzzy"]
}
```

Response:

```
{
  "message": "Ok",
  "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
  "match_results": [
    {
      "first_name": "fuzzy_match",
      "last_name": "match",
      "dob": "match"
    }
  ]
}
```

`fuzzy_match` was returned for `first_name` because Mathew and Matthew share the same metaphone code (`M0`) and start with the same letter. The DOB and last name still matched exactly, so this is a strong overall match across all three supplied identifiers.

To allow spelling variations on the last name as well, add `last_name_matching: ["exact", "fuzzy"]` . This also widens the underlying candidate search so that records with a fuzzy last name are considered, not just records with the exact supplied last name.

Nickname / alias tolerance

Customers may identify themselves by their preferred name (Bob, Jenny, Tony) when their record is on file under a formal name (Robert, Jennifer, Antonio). Opt in to alias matching to allow the comparison.

Request:

```
{
  "first_name": "Bob",
  "last_name": "Brown",
  "birth_date": "1971-03-18",
  "first_name_matching": ["exact", "alias"]
}
```

Response:

```
{
  "message": "Ok",
  "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
  "match_results": [
    {
      "first_name": "alias_match",
```

```
      "last_name": "match",
      "dob": "match"
    }
  ]
}
```

The alias group catalogue includes hundreds of common nickname pairings. You can combine `["exact", "alias", "fuzzy"]` to tolerate both nicknames and typos, or add `"partial"` to also accept single-initial matches like `"R" -> "Robert"`.

DOB partial matches

Date of birth has its own ladder of partial outcomes for cases where the customer has the year right but not the exact date.

Day / month reversal

Mary Jones is on file with a DOB of 12 August 1989 (`1989-08-12` in `YYYY-MM-DD`). Upstream systems frequently mix up the regional date conventions `DD/MM/YYYY` and `MM/DD/YYYY` . For example, a customer who handwrites `12/08/1989` on a form (intending 12 August) might have it entered into a US-style system as 8 December 1989 - the day and month silently swap. When that incorrect date later makes its way into a request to this API, the supplied value is `1989-12-08` instead of `1989-08-12` . The check spots that the year matches and the day and month are swapped, and reports the outcome explicitly.

> The API itself always requires `birth_date` in `YYYY-MM-DD` format - the day/month confusion described above happens upstream of the API, in the system that originally collected the date. This check just helps you recover when that has already happened.

Request:

```
{
  "first_name": "Mary",
  "last_name": "Jones",
  "birth_date": "1989-12-08"
}
```

Response:

```
{
  "message": "Ok",
  "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
  "match_results": [
    {
      "first_name": "match",
      "last_name": "match",
      "dob": "match_day_month_reversal"
    }
  ]
}
```

Year-only

If only the year matches but the day and month do not (and they are not a reversal), the response returns `match_year` :

Request:

```
{
  "first_name": "Mary",
  "last_name": "Jones",
  "birth_date": "1989-05-15"
}
```

Response:

```
{
  "message": "Ok",
  "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
  "match_results": [
    {
      "first_name": "match",
      "last_name": "match",
      "dob": "match_year"
    }
  ]
}
```

No DOB on file

Some universe records have no DOB on file at all. When that is the case, the candidate returns `dob: no_record` rather than `no_match` - a useful distinction because it tells you "we matched this person, but we have no DOB to compare against" instead of "we matched this person and their DOB is different to yours".

The sandbox includes a Robert Brown record with phone `0491222333` and no DOB on file (distinct from the Robert Patrick Brown record born 1971-03-18). Searching for him by phone, with any DOB, shows the outcome:

Request:

```
{
  "first_name": "Robert",
  "last_name": "Brown",
  "phone": "0491222333",
  "birth_date": "1985-06-04"
}
```

Response:

```
{
  "message": "Ok",
  "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
  "match_results": [
    {
      "first_name": "match",
      "last_name": "match",
      "dob": "no_record",
      "phone": "match"
    }
  ]
}
```

`dob: no_record` is a signal that the candidate is on file but does not have a date of birth recorded, so the supplied DOB cannot be confirmed or refuted from our universe. It is not a positive match, but it is also not a `no_match` either - decide per workflow whether the absence of a DOB on file is acceptable, or whether you want to fall back to another verification path.

If your supplied DOB happens to match a different candidate (for example, supplying `1971-03-18` would match the Robert Patrick Brown record), that other candidate's `dob: match` will outscore this candidate's `dob: no_record` and become the top result. Set `return_multiple_candidates: true` if you want to see both rows.

The other partial DOB outcomes (`match_year` , `match_day_month_reversal`) are useful when DOB is collected from messy sources (handwritten forms, customer entry on a phone keypad).

Dirty input with `ignore_errors`

When you are checking data collected from imperfect sources - a customer-facing signup form, a CSV import, a third-party feed - some individual fields will inevitably be malformed. By default, an invalid phone number or DOB will cause the whole call to fail with a `400` . With `ignore_errors: true` , the endpoint silently drops any individual field that fails validation and runs the match against whatever valid input remains. Dropped fields come back in the response as `not_used` so you can see exactly what was ignored.

Request:

```
{
  "first_name": "John",
  "last_name": "Smith",
  "birth_date": "1987666",
  "phone": "1234",
  "email": "jsmith1998@example.com",
  "ignore_errors": true
}
```

Response:

```
{
  "message": "Ok",
  "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
  "match_results": [
    {
      "first_name": "match",
      "last_name": "match",
      "dob": "not_used",
      "phone": "not_used",
      "email": "match"
    }
  ]
}
```

Use `ignore_errors: true` whenever you cannot guarantee the cleanliness of the inputs and you'd rather get a partial result than re-architect a retry-with-bad-fields-omitted loop on your side. Note that the request still has to be **structurally** well-formed (correct types, `last_name` present); only individual field-value problems are soft-ignored.

Confidence-based routing

Each candidate row reports the per-field outcomes directly. To drive a downstream routing rule, branch on the specific outcomes that matter for your use case rather than on a single aggregate score. For example:

- exact `last_name` plus exact `dob` plus any `phone` match ? auto-approve.
- exact `last_name` plus a partial DOB outcome (`match_day_month_reversal` / `match_year`) ? manual review queue.
- everything else ? reject or trigger a more thorough verification (such as a DVS document check or an ID Pass).

The exact thresholds depend on your risk appetite and the field mix you are sending; the Best-candidate selection table tells you how strongly each outcome weighs in the candidate ranking, which is a useful starting point when calibrating.

Common gotchas and tips

- **Last name is the only required field**, but supplying more identifiers materially improves both the candidate pool and the per-row confidence. A request with just a last name will often return many candidates with low scores; adding a DOB or address narrows it dramatically.
- **Each phone and email slot is evaluated independently.** Supplying `phone` and `phone3` produces a `phone` outcome and a `phone3` outcome in the response; the slot order in the request has no effect on matching. Slots you did not supply are omitted from the row.
- **The address block is all-or-nothing.** When no address was supplied, the `address` rollup and the four component fields are all omitted from the row. When the address was supplied but could not be resolved, you get a `400` (or a `not_used` block if `ignore_errors: true` is set).
- **`include_deceased: true` adds deceased records but the response carries no deceased indicator.** Most use cases (KYC, identity verification, contact validation) should leave this off; only enable it when you have a specific need to match against deceased records and your downstream workflow handles them appropriately.
- **`return_multiple_candidates: true` does not change the row schema.** Single-candidate mode just returns the top row; multi-candidate mode returns up to 5 candidates ordered best-first using the same per-row layout.
- **Identical candidate rows are collapsed.** Two universe records that produce the exact same per-field outcome map against your request appear in the response as a single row. The row count therefore reflects distinct outcome maps, not the count of underlying universe records. Add identifiers to your request if you want candidates to surface as distinct rows.
- **Score ties are broken deterministically** so that repeating the same request always produces the same ordering, even when several candidates have an identical score. Treat the order in `match_results` as authoritative for picking a "best" record.

ASIC Extracts

The ASIC Extract API allows you to generate official ASIC extracts for companies and individuals. Extracts can be returned as **JSON** for system integration or **PDF** for human-readable reports.

What is an ASIC Extract?

An ASIC extract is an official record containing company or personal information sourced from the Australian Securities and Investments Commission (ASIC). The API enables you to request these extracts on demand and retrieve them programmatically.

ASIC extracts play an important role in due-diligence, compliance, and operational workflows. This guide provides an overview of how to request and retrieve these extracts using our API in a consistent and reliable way. Once familiar with the extract types and request process, your teams can automate information gathering and incorporate ASIC data directly into your business processes.

Extract Types Available

A number of different extracts are available from the ASIC API. These can be returned as JSON or PDF.

- Company Current Extract
- Company Historical Extract
- Company Relational Extract
- Person Current Extract
- Person Historical Extract

When to Use Each Extract Type

Extract Type	Description	Typical Use Case
Company Current	Current registration details only.	Confirm up-to-date company details and status.
Company Historical	Full history of changes over time.	Compliance checks, due diligence, historical analysis.
Company Relational	Relationships between a company and other entities.	Group structure, related entities, and relationship mapping.
Person Current	A person’s current roles and shareholdings.	Check current directorships and officeholder roles.
Person Historical	A person’s current and past roles, licences, and company associations.	Deep background checks and historical involvement analysis.

Extract Types

Company Current Extract

Provides a snapshot of a company's **current registration details**, including officeholders, registered address, share structure (where applicable), and status at the time of the extract.

- Example Company Current Extract PDF
- Example Company Current Extract JSON

Company Historical Extract

Provides a **full history** of a company's registration details, including current and past names, former officeholders, past addresses, share-structure changes, and historic status information.

- Example Company Historical Extract PDF
- Example Company Historical Extract JSON

Company Relational Extract

Provides a **relationship focused** view of how a company relates to other entities, including:

- Its ultimate holding company
- Entities in which it holds interests, licences, or roles
- Membership or directorship links.
- Example Company Relational Extract PDF
- Example Company Relational Extract JSON

Person Current Extract

Provides a person's **current involvement** with companies, including their officeholder roles, directorships, or shareholdings as recorded by ASIC.

- Example Person Current Extract PDF
- Example Person Current Extract JSON

Person Historical Extract

Provides a person's **current and past** corporate roles, shareholdings, licences and other company-related associations as recorded by ASIC.

- Example Person Historical Extract PDF
- Example Person Historical Extract JSON

API Usage

There are four API endpoints used to perform ASIC extracts for companies and individuals. Two are used for searching, and two are used for requesting and retrieving extract results.

Endpoint Name	Description
ASIC Extract Person Search	Searches the ASIC officeholder/shareholder register for a person using their name and date of birth.
ASIC Extract Company Search	Searches the ASIC company register for a company using the company's name, ABN, or ACN.
ASIC Extract	Requests a current, historical, or relational ASIC extract for a company or individual.
Retrieve ASIC Extract	Retrieves the results of an ASIC extract request.

Quick Start

The following sections show the typical workflow for requesting and retrieving ASIC extracts for companies and individuals.

Company Extract (Current, Historical or Relational)

1. (Optional) Use **ASIC Extract Company Search** to find a company by name, partial name, ABN, or ACN.
2. Identify the company of interest and note its `abn` or `acn`.
3. Use the **ASIC Extract** endpoint to request the extract for the company, providing the `abn` or `acn`.
4. Use **Retrieve ASIC Extract** with the returned `api_reference` to poll for the extract and retrieve it in JSON or PDF format.

Person Extract (Current or Historical)

1. Use **ASIC Extract Person Search** to search for a person using their name and date of birth.
2. From the matches, select the person of interest and note the `person_id`, `search_id`, and any `additional_person_ids`.
3. Use the **ASIC Extract** endpoint to request the extract for the person, providing the `person_id`, `search_id`, and any `additional_person_ids`.
4. Use **Retrieve ASIC Extract** with the `api_reference` returned by the **ASIC Extract** endpoint to obtain the extract.

Processing Time

The system begins generating the extract as soon as the ASIC Extract endpoint is called.

- Extract generation normally completes within a few seconds.
- During busy periods, processing can take **up to 10 minutes**.
- The **Retrieve ASIC Extract** endpoint can be used to check the status and retrieve the extract once it is ready.

Billing

> Note: Calls to the ASIC Extract endpoint outside of the sandbox environment will be billed. Refer to your commercial agreement or pricing documentation for details.

API Endpoints

More technical details on the API endpoints (including full request/response schemas, status codes, and error formats) are available in the API Reference section of the documentation.

ASIC Extract Person Search

The ASIC Extract Person Search endpoint is used to search the ASIC officeholder/shareholder register for a person using their name and date of birth.

This search returns a list of matching ASIC identities, including the `person_id` and `search_id` you must supply to the **ASIC Extract** endpoint.

The search may also return **additional person IDs** that are linked to the person of interest. These should be included in the `additional_person_ids` parameter when requesting the extract to ensure all linked identities are captured.

Example Search

```
{
    "first_name": "John",
    "middle_name": "Michael",
    "last_name": "Doe",
    "birth_date": "1980-05-15",
    "extract_type": "historical"
}
```

Example Response

```
{
    "message": "Ok",
    "function": "asic_extract_person_search",
    "api_reference": "0653317d-4d78-4e39-bd08-f7edbc30a656",
    "matches": {
        "person_id": "123456780",
        "search_id": "11223344",
        "name": "JOHN MICHAEL DOE",
        "dob": "1980-05-15",
        "state": "NSW",
        "suburb": "SYDNEY",
        "additional": [
            {
                "person_id": "123456781",
                "search_id": "11223344",
                "name": "JOHN MICHAEL DOE",
                "dob": null,
                "state": null,
                "suburb": null,
                "additional": []
            }
        ]
    }
}
```

> Important: The `person_id`, `search_id`, and any `additional_person_ids` are only valid for 24 hours. If you need to request an extract for a person after this time, you must perform a new person search.

The data returned from the ASIC Extract Person Search endpoint can then be passed to the ASIC Extract endpoint to request the extract for that person.

See the ASIC Extract Person Search API documentation for detailed information on the request and response.

ASIC Extract Company Search

The ASIC Extract Company Search endpoint is used to search the ASIC company register for a company using the company's name, ABN, or ACN.

This search returns a list of matching ASIC companies, including the `abn` and `acn` you must supply to the **ASIC Extract** endpoint.

This function is provided as a convenience to obtain the ABN/ACN for a company. Unlike the ASIC Extract Person Search process, there is **no requirement** to perform a company search before requesting an extract, as only the `abn` or `acn` is required by the extract endpoint.

Example Search

```
{
    "search": "Tech Innovators",
    "status": "registered"
}
```

Example Response

```
{
    "message": "Ok",
    "function": "asic_extract_company_search",
    "api_reference": "c3c74e96-ead1-482d-bfb5-5b902090cf0c",
    "matches": [
        {
            "name": {
                "name": "TECH INNOVATORS LTD"
            },
            "identifier": {
                "numberHeading": "ACN",
                "number": 111111114
            },
            "abrEntity": {
                "abn": "32111111114",
                "entityName": "TECH INNOVATORS LTD",
                "entityType": "PRV",
                "effectiveDate": "2018-03-15"
            },
            ...
        }
    ],
    "more_results": false
}
```

See the ASIC Extract Company Search API documentation for detailed information on the request and response.

ASIC Extract

The ASIC Extract endpoint is used to request the generation of a **current**, **historical**, or **relational** ASIC extract for a company or individual.

> Note: Calls to this endpoint outside of the sandbox will be billed.

Example Request

```
{
    "type": "company",
    "abn": "32111111114",
    "extract_type": "current"
}
```

Example Response

```
{
    "message": "Ok",
    "function": "asic_extract",
    "api_reference": "0653317d-4d78-4e39-bd08-f7edbc30a656"
}
```

The endpoint returns immediately with an `api_reference`, and the system begins generating the extract in the background.

The returned `api_reference` can then be used to retrieve the results of the extract using the **Retrieve ASIC Extract** endpoint.

ABN validation

Company extracts are only available for companies. When an `abn` is supplied, it is first resolved against the company registers; if the ABN belongs to an individual (for example a sole trader) or cannot be matched to a company, the request is rejected with a `400` response:

```
{
    "message": "There is no ACN associated with this ABN. Company extracts require the entity to have an ACN."
}
```

In the sandbox environment, any valid ABN that does not belong to one of the sandbox companies (for example `51824753556`) can be used to reproduce this response.

See the ASIC Extract API documentation for detailed information on the request and response.

Retrieve ASIC Extract

The Retrieve ASIC Extract endpoint is used to retrieve the results of an ASIC extract request.

Both JSON and PDF formats are available. To return a PDF, append `?pdf=true` to the URL.

Example Request

```
GET /asic_extract/0653317d-4d78-4e39-bd08-f7edbc30a656
```

Example Response

```
{
    "message": "Ok",
    "function": "asic_extract_show",
```

```
    "api_reference": "9a2c2264-2e36-4b00-97a0-686a89f306c2",
    "extract": {
        "id": "e947f82f-d259-4510-8870-fab2eda4d562",
        "entity_type": "company",
        "cases": [],
        "entity": {
            "abn": "33234567894",
            "acn": "234567894",
            "name": "Creative Media Agency",
            ...
        }
        ...
    }
    ...
}
```

Example PDF Request

```
GET /asic_extract/9a2c2264-2e36-4b00-97a0-686a89f306c2?pdf=true
```

This returns the extract as a PDF document in the response body.

Waiting for the extract to be ready

As noted previously, generation is normally completed within a few seconds but can take up to 10 minutes if the ASIC service is busy.

When the extract is not yet complete, the endpoint may wait for up to 30 seconds for the extract to be ready. If the extract is still not ready, the endpoint will return a `202` status code and the response will be:

```
{
    "message": "Extract is still being processed. Please try again later."
}
```

You can then repeat the request with the same `api_reference` until the extract is returned.

UK Companies House

The UK Companies House API allows you to search for and retrieve detailed information about companies and officers registered with Companies House in the United Kingdom. Reports can be returned as **JSON** for system integration or **PDF** for human-readable documents.

What is Companies House?

Companies House is the United Kingdom's registrar of companies. It maintains the official register of all incorporated companies in England, Wales, Scotland, and Northern Ireland. The register includes information about company registration details, directors, secretaries, persons with significant control (PSCs), and filed documents.

The Companies House API enables you to search the register, retrieve company reports, officer reports, and filed documents on demand. This data plays an important role in due-diligence, KYB (Know Your Business), compliance, and operational workflows.

Available Functions

The UK Companies House API provides the following functions:

- **Company Search** - Search for companies by name or company number
- **Officer Search** - Search for officers (directors, secretaries) by name
- **Company Report** - Generate comprehensive company reports with officers, PSCs, and filing history
- **Officer Report** - Generate reports on individual officers including their appointments
- **Document Download** - Retrieve filed documents (annual accounts, confirmation statements, etc.)

API Usage

There are seven API endpoints used to interact with UK Companies House data:

Endpoint Name	Description
Company House Search	Searches for companies by name or company number
Company House Officer Search	Searches for officers by name
Company House Report	Requests a comprehensive company report
Retrieve Company House Report	Retrieves the results of a company report request
Company House Officer Report	Requests a comprehensive officer report
Retrieve Company House Officer Report	Retrieves the results of an officer report request
Company House Document	Downloads a filed document by document ID

Quick Start

The following sections show the typical workflow for using the UK Companies House API.

Company Report Workflow

1. Use **Company House Search** to find a company by name or company number.
2. Identify the company of interest and note its `company_number`.
3. Use the **Company House Report** endpoint to request the report, providing the `company_number`.
4. Use **Retrieve Company House Report** with the returned `api_reference` to poll for the report and retrieve it in JSON or PDF format.

Officer Report Workflow

1. Use **Company House Officer Search** to search for an officer by name.
2. From the matches, select the officer of interest and note the `officer_id`.
3. Use the **Company House Officer Report** endpoint to request the report, providing the `officer_id`.
4. Use **Retrieve Company House Officer Report** with the returned `api_reference` to poll for the report and retrieve it in JSON or PDF format.

Document Download Workflow

1. Retrieve a company report to obtain the filing history.
2. Identify the document of interest and note its `document_id`.
3. Use the **Company House Document** endpoint to download the document as a PDF.

Processing Time

The system begins generating reports as soon as the request endpoint is called.

- Report generation normally completes within a few seconds.
- During busy periods, processing can take longer.
- The **Retrieve** endpoints can be used to check the status and retrieve the report once it is ready.
- The service polls for up to 30 seconds. If the report is still not ready, a `202` status code is returned and you should continue polling.

Sandbox Environment

The sandbox environment uses deterministic test data and does not return live data. Use the sample company numbers, officer IDs, and document IDs below to receive predictable responses.

Sandbox Companies

Company Name	Company Number	Status	Notes
TEST COMPANY LTD	12345678	Active	Standard company with 2 officers
SAMPLE HOLDINGS PLC	87654321	Active	PLC with multiple officers including corporate secretary
DORMANT SERVICES LTD	11223344	Dormant	Dormant company

Company Name	Company Number	Status	Notes
DISSOLVED EXAMPLE LTD	99887766	Dissolved	Dissolved company with previous names
SCOTTISH ENTERPRISE SC	SC654321	Active	Scottish company
DELAYED REPORT LTD	55667788	Active	Returns 202 on first poll (simulates async processing)

Sandbox Officers

Name	Officer ID	Type	DOB (Y-M)	Appointments
John David SMITH	ABC123DEF456	Natural	1975-06	3
Jane Elizabeth DOE	XYZ789GHI012	Natural	1980-03	1
Robert James JOHNSON	JKL345MNO678	Natural	1968-11	5
Christopher Mark TAYLOR	DEL789AYE012	Natural	1990-05	1 (delayed report)
CORPORATE SECRETARIES LIMITED	VWX567YZA890	Corporate	N/A	25
David DISQUALIFIED	DIS123QUA456	Natural (Disqualified)	1970-01	0

Sandbox Documents

Document IDs in the sandbox must use the `sandbox-document-*` prefix format. Examples from company filing histories include:

Document ID	Description
sandbox-document-001	TEST COMPANY LTD annual accounts
sandbox-document-002	TEST COMPANY LTD confirmation statement
sandbox-document-003	SAMPLE HOLDINGS PLC annual accounts

Any document ID starting with `sandbox-document-` will return a valid placeholder PDF.

Billing

> Note: Calls to the Company House Report and Officer Report endpoints outside of the sandbox environment will be billed. Refer to your commercial agreement or pricing documentation for details.

API Endpoints

More technical details on the API endpoints (including full request/response schemas, status codes, and error formats) are available in the API Reference section of the documentation.

Company House Search

The Company House Search endpoint searches the UK Companies House register for companies matching the provided search term.

The search is intelligent and will detect if the search term matches a company number format:

- **Company number search**: If the search term matches the 8-character company number format, a direct lookup is performed.
- **Name search**: Otherwise, a fuzzy name search is performed against registered company names.

Example Request

```
{
    "company": "Test Company",
    "is_operational": true,
    "max_results": 50
}
```

Example Response

```
{
    "message": "Ok",
    "function": "company_house_search",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "records": [
        {
            "company_number": "12345678",
            "company_name": "TEST COMPANY LTD",
            "company_status": "active",
            "company_type": "ltd",
            "date_of_creation": "2020-01-15",
            "registered_office_address": {
                "address_line_1": "123 Test Street",
                "address_line_2": "Test District",
                "locality": "London",
                "postal_code": "EC1A 1BB",
                "country": "United Kingdom"
            }
        }
    ],
    "records_returned": 1,
    "total_records_available": 1
}
```

See the Company House Search API documentation for detailed information on the request and response.

Company House Officer Search

The Company House Officer Search endpoint searches the UK Companies House register for officers matching the provided name.

An optional `date_of_birth` parameter can be provided to narrow results. When supplied, only officers whose date of birth matches the given year and month are returned. The day component is accepted for convenience but is not used for matching, as Companies House only provides month and year of birth for most officers.

Example Request

```
{
    "name": "John Smith",
    "date_of_birth": "1975-06-15",
```

```
    "max_results": 50
}
```

Example Response

```
{
    "message": "Ok",
    "function": "company_house_officer_search",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "records": [
        {
            "officer_id": "ABC123DEF456",
            "title": "John David SMITH",
            "date_of_birth": "1975-06",
            "address": {
                "address_line_1": "Test Street",
                "address_line_2": "Test District",
                "locality": "London",
                "postal_code": "EC1A 1BB",
                "country": "United Kingdom"
            },
            "address_snippet": "123 Test Street, Test District, London, United Kingdom, EC1A 1BB",
            "appointment_count": 3,
            "description": "Total number of appointments 3 - Born June 1975"
        }
    ],
    "records_returned": 1,
    "more_results": false
}
```

See the Company House Officer Search API documentation for detailed information on the request and response.

Company House Report

The Company House Report endpoint requests a comprehensive company report. The response confirms the job was queued and returns an `api_reference` for polling.

> Note: Calls to this endpoint outside of the sandbox will be billed.

Example Request

```
{
    "company_number": "12345678"
}
```

Example Response

```
{
    "message": "Ok",
    "function": "company_house_report",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95"
}
```

The endpoint returns immediately with an `api_reference`, and the system begins generating the report in the background.

See the Company House Report API documentation for detailed information on the request and response.

Retrieve Company House Report

The Retrieve Company House Report endpoint retrieves the results of a company report request.

The report contains the company's profile information (name, status, type, registered address, accounts, confirmation statement), an `officers` array with all current and former officers, a `filing_history` object containing the most recent filings (up to 200), and a `persons_with_significant_control` array. Each section is only included when the corresponding data exists at Companies House.

Both JSON and PDF formats are available. To return a PDF, append `?pdf=true` to the URL.

Example Request

```
GET /company_house_report/fe4291ca-d831-4760-96df-c9cb03b3cd95
```

Example Response


```
{
    "message": "Ok",
    "function": "company_house_report_show",
    "api_reference": "7cbb094c-0f27-42f4-9f36-9cf0b2f6d5ff",
    "report": {
        "company_name": "TEST COMPANY LTD",
        "company_number": "12345678",
        "company_status": "active",
        "type": "ltd",
        "jurisdiction": "england-wales",
        "date_of_creation": "2020-01-15",
        "can_file": true,
        "report_date": "2026-02-03T10:00:00Z",
        "registered_office_address": {
            "address_line_1": "123 Test Street",
            "address_line_2": "Test District",
            "locality": "London",
            "postal_code": "EC1A 1BB",
            "country": "United Kingdom"
        },
        "accounts": {
            "overdue": false,
            "next_due": "2026-10-15",
            "last_accounts": {
                "type": "full",
                "made_up_to": "2025-01-15"
            },
            "accounting_reference_date": {
                "day": "15",
                "month": "01"
            }
        },
        "confirmation_statement": {
            "overdue": false,
            "next_due": "2026-07-15",
            "last_made_up_to": "2025-06-15"
        },
        "sic_codes": [
            {
                "code": "62020",
                "description": "Information technology consultancy activities"
            }
```

```
        ],
        "officers": [
            {
                "name": "SMITH, John David",
                "officer_id": "ABC123DEF456",
                "officer_role": "director",
                "appointed_on": "2020-01-15",
                "date_of_birth": "1975-06",
                "nationality": "British",
                "country_of_residence": "England",
                "address": {
                    "premises": "123",
                    "address_line_1": "Test Street",
                    "locality": "London",
                    "postal_code": "EC1A 1BB",
                    "country": "United Kingdom"
                }
            },
            {
                "name": "DOE, Jane Elizabeth",
                "officer_id": "XYZ789GHI012",
                "officer_role": "secretary",
                "appointed_on": "2020-01-15",
                "date_of_birth": "1980-03",
                "nationality": "British",
                "country_of_residence": "England",
                "address": {
                    "premises": "456",
                    "address_line_1": "Sample Road",
                    "locality": "London",
                    "postal_code": "EC1A 2CC",
                    "country": "United Kingdom"
                }
            }
        ],
        "filing_history": {
            "total_files": 5,
            "returned_files": 5,
            "files": [
                {
                    "date": "2025-10-01",
                    "type": "AA",
                    "pages": 15,
                    "category": "accounts",
                    "description": "accounts-with-accounts-type-full",
                    "document_id": "sandbox-document-001",
                    "transaction_id": "SANDBOX00001"
                }
            ]
        },
        "persons_with_significant_control": [
            {
                "name": "Mr John David Smith",
                "kind": "individual-person-with-significant-control",
                "natures_of_control": [
                    "ownership-of-shares-75-to-100-percent",
                    "voting-rights-75-to-100-percent"
                ],
                "notified_on": "2020-01-15",
                "country_of_residence": "England",
                "nationality": "British"
            }
```

```
        ]
    }
}
```

Waiting for the report to be ready

When the report is not yet complete, the endpoint may wait for up to 30 seconds. If the report is still not ready, the endpoint will return a `202` status code:

```
{
    "message": "Extract is still being processed. Please try again later."
}
```

You can then repeat the request with the same `api_reference` until the report is returned.

See the Retrieve Company House Report API documentation for detailed information.

Company House Officer Report

The Company House Officer Report endpoint requests a comprehensive officer report. The response confirms the job was queued and returns an `api_reference` for polling.

> Note: Calls to this endpoint outside of the sandbox will be billed.

Example Request

```
{
    "officer_id": "ABC123DEF456",
    "disqualified": false,
    "type": "natural"
}
```

Example Response

```
{
    "message": "Ok",
    "function": "company_house_officer_report",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95"
}
```

See the Company House Officer Report API documentation for detailed information.

Retrieve Company House Officer Report

The Retrieve Company House Officer Report endpoint retrieves the results of an officer report request.

The report contains the officer's personal details and an `items` array listing every appointment (role) they hold or have held across different companies. Each item includes the company details (`appointed_to`), the role (`officer_role`), appointment date, and correspondence address. Active appointments have no `resigned_on` field; former appointments include `resigned_on` .

For disqualified officers (requested with `disqualified=true`), the report includes a `disqualifications` array instead of appointment items.

Both JSON and PDF formats are available. To return a PDF, append `?pdf=true` to the URL.

Example Request

```
GET /company_house_officer_report/fe4291ca-d831-4760-96df-c9cb03b3cd95
```

Example Response


```
{
    "message": "Ok",
    "function": "company_house_officer_show",
    "api_reference": "7cbb094c-0f27-42f4-9f36-9cf0b2f6d5ff",
    "report": {
        "officer_id": "XYZ789GHI012",
        "name": "Jane Elizabeth DOE",
        "date_of_birth": "1980-03",
        "etag": "sandbox-etag-002",
        "kind": "personal-appointment",
        "type": "natural",
        "is_corporate_officer": false,
        "disqualified": false,
        "report_date": "2026-02-03T10:00:00Z",
        "total_results": 1,
        "items_per_page": 50,
        "start_index": 0,
        "items": [
            {
                "name": "Jane Elizabeth DOE",
                "officer_role": "secretary",
                "appointed_on": "2020-01-15",
                "appointed_to": {
                    "company_name": "TEST COMPANY LTD",
                    "company_number": "12345678",
                    "company_status": "active"
                },
                "address": {
                    "premises": "456",
                    "address_line_1": "Sample Road",
                    "locality": "London",
                    "country": "United Kingdom",
                    "postal_code": "EC1A 2CC"
                },
                "nationality": "British",
                "name_elements": {
                    "title": "Ms",
                    "forename": "Jane",
                    "other_forenames": "Elizabeth",
                    "surname": "DOE"
                },
                "country_of_residence": "England",
                "links": {
                    "company": "/company/12345678"
                },
                "is_pre_1992_appointment": false
            }
        ]
    }
}
```

Waiting for the report to be ready

When the report is not yet complete, the endpoint may wait for up to 30 seconds. If the report is still not ready, the endpoint will return a `202` status code:

```
{
    "message": "Extract is still being processed. Please try again later."
}
```

You can then repeat the request with the same `api_reference` until the report is returned.

See the Retrieve Company House Officer Report API documentation for detailed information.

Company House Document

The Company House Document endpoint retrieves a filed document from Companies House. Documents are returned as PDF files.

Example Request

```
GET /company_house_document/sandbox-document-001
```

This returns the document as a PDF file in the response body.

> **Sandbox Note:** Document IDs must use the `sandbox-document-*` prefix in the sandbox environment. Valid sandbox document IDs can be found in the `filing_history` section of company reports (e.g., `sandbox-document-001`, `sandbox-document-002`).

Error Responses

If the document is not available in PDF format, a `415 Unsupported Media Type` error is returned:

```
{
    "message": "Document media type is not supported"
}
```

See the Company House Document API documentation for detailed information.

Company Risk Report

A comprehensive risk report that combines various data sources to provide insights into a company's financial health, legal standing, and potential risks associated with business dealings.

API Usage

The Company Risk Report uses three endpoints:

Endpoint Name	Description
ASIC Company Search	Search ASIC to find a company by name, ABN, or ACN.
Company Risk	Request generation of a Company Risk Report using an ABN or ACN.
Retrieve Company Risk Report	Retrieve the report results (JSON) or download a PDF once it is ready.

Quick Start

1. (Optional) Use **ASIC Company Search** to locate the company and note its `abn` or `acn`.
2. Use **Company Risk** with the `abn` or `acn` to start the report generation.
3. Use **Retrieve Company Risk Report** with the returned `api_reference` to poll for completion and retrieve the report.
4. Append `?pdf=true` to the show endpoint to return the report as a PDF.

Processing Time

- Report generation usually completes within a few seconds.
- The show endpoint waits up to 30 seconds before returning a `202` response if the report is not ready.

Billing

> Note: Calls to the Company Risk endpoint outside of the sandbox environment will be billed. Refer to your commercial agreement or pricing documentation for details.

API Endpoints

ASIC Company Search

The ASIC Company Search endpoint is used to search the ASIC company register for a company using the company's name, ABN, or ACN.

This search returns a list of matching ASIC companies, including the `abn` and `acn` you must supply to the **Company Risk Report Request** endpoint.

This function is provided as a convenience to obtain the ABN/ACN for a company. Unlike the ASIC Extract Person Search process, there is **no requirement** to perform a company search before requesting an extract, as only the `abn` or `acn` is required by the extract endpoint.

See the ASIC Company Search API documentation for detailed information on the request and response.

Example Search

```
{
    "search": "Marble Kookaburra Interiors",
    "status": "registered"
}
```

Example Response

```
{
    "message": "Ok",
    "function": "asic_company_search",
    "api_reference": "c3c74e96-ead1-482d-bfb5-5b902090cf0c",
    "matches": [
        {
            "identifier": {
                "numberHeading": "ACN",
                "number": 655375652
            },
            "name": {
                "name": "MARBLE KOOKABURRA INTERIORS PTY LTD"
            },
            "abrEntity": {
                "abn": "84655375652",
                "entityName": "MARBLE KOOKABURRA INTERIORS PTY LTD",
                "entityType": "PRV",
                "effectiveDate": "2021-07-19"
            },
            ...
        }
    ],
    "more_results": false
}
```

Company Risk

Request generation of a Company Risk Report. Supply **either** `abn` or `acn` (not both).

Example Request

```
{
    "abn": "84655375652"
}
```

Example Response

```
{
    "message": "Ok",
    "function": "company_risk",
    "api_reference": "0653317d-4d78-4e39-bd08-f7edbc30a656"
```

```
}
```

The endpoint returns immediately with an `api_reference`, and the system begins generating the extract in the background.

The returned `api_reference` can then be used to retrieve the results of the extract using the **Retrieve Company Risk Report** endpoint.

See the Company Risk API documentation for detailed information on the request and response.

Retrieve Company Risk Report

Retrieve the report after requesting it.

Example Request

```
GET /company_risk/0653317d-4d78-4e39-bd08-f7edbc30a656
```

Example Response

```
{
    "message": "Ok",
    "function": "company_risk_show",
    "api_reference": "9a2c2264-2e36-4b00-97a0-686a89f306c2",
    "report": {
        "cases": [
            {
                "name": "MARBLE KOOKABURRA INTERIORS PTY LTD",
                "type": "Court Case",
                "uuid": "1e3f9b52-5485-4c5e-8a0b-72e8792cf222",
                "state": "VIC",
                "suburb": "Melbourne",
                "parties": [
                    {
                        "abn": "84655375652",
                        "acn": "655375652",
                        "fax": "",
                        "name": "MARBLE KOOKABURRA INTERIORS PTY LTD",
                        "role": "Respondent",
                        "type": "Company",
                        "phone": "",
                        "address": "",
                        "representative_firm": "",
                        "representative_name": ""
                    },
                    ...
                ],
                ...
            },
            ...
        ],
        ...
    }
}
```

Example PDF Request

```
GET /company_risk/0653317d-4d78-4e39-bd08-f7edbc30a656?pdf=true
```

Waiting for the report to be ready

If the report is not yet complete, the endpoint will return a `202` response:

```
{
    "message": "Report is still being processed. Please try again later."
}
```

You can then repeat the request with the same `api_reference` until the report is returned.

See the Retrieve Company Risk Report API documentation for detailed information on the request and response.

DVS Responses

The response from a DVS request consists of the following attributes in the data object:

- **response_type**: The type of response returned from DVS.
- **activity_id**: The unique identifier for the response generated by DVS.
- **verification_request_number**: A unique identifier assigned to the request that is matched to a DVS response.
- **verification_result_code**: The result of the verification request. The following codes are returned:
  - **Y**: The data in the request matches data held by the issuer.
  - **N**: The data in the request does not match data held by the issuer.
  - **D**: A data error occurred at the issuer - Eg: The document itself is invalid or not electronically held.

Expanded Responses

In the case of N or D results, some responses include an expanded message that provides additional information about the result.

The expanded response will be listed in an **additional_information** array. The **additional_information** array will contain one or more objects with the following attributes:

- **code**: An ID code for the expanded response.
- **message**: The expanded response message.

DVS System Error

It is also possible that the DVS returns a system error in the event that there is a problem at either the DVSHub or the Document Issuer. In these cases, a 503 HTTP status code will be returned, with the message 'DVS returned a system error. Please try again later.'.

Depending on the problem, it may be worth retrying the request after a short delay.

Malicious Activity Protection Solution Lockout

Individual documents may be locked out from further verification attempts if a high number of verification requests are made within a short period of time. The number of attempts before lockout ranges from 4 to 10, depending on the document type.

If a document is locked out, an `N` result will be returned with an error code of `DVS-001` and the message `Document Temporarily Locked`, indicating that the document has been temporarily locked out from further verification attempts. This lockout will expire after 20 minutes if no further attempts are made on the document. The lockout timeny further attempts during the lockout period will restart the lockout timer.

Sandbox Responses

In the Global Data API, the sandbox environment is a separate test dataset that can be used for development and integration testing. The test account is identical to the production data account and allows access to the API web portal and API but does not incur any usage costs.

The DVS sandbox environment will produce a random verification result code - Y, D, N, S for any valid request. Specific results can be generated, depending on the fields passed in the request. The values and results are outlined in the sections pertaining to the specific DVS endpoint.

The activity ID for the response will be prefixed with the word 'sandbox'. Eg: sandbox-612aca83-4844-4f42-bf0d-fcf3f1986d64

An expanded response which elaborates on an N or D result will randomly be returned if such a response exists. Specific expanded responses can be simulated by passing specific values as outlined in each of the endpoints below.

System Error

A DVS System Error can be simulated by passing the value '**error**' as the **lastName** or the **fullName** attribute, depending on the DVS endpoint.

Lockout

A DVS Lockout Response can be simulated by passing the value '**locked**' as the **lastName** or the **fullName** attribute, depending on the DVS endpoint.

DVS address

The result code for a DVS address check will be determined by a range of street numbers:

Street Number	Result
101 - 200	Y
201 - 300	D
301 - 400	N

Specific expanded responses will be determined by the suburb:

Suburb	Result	Code	Response
Sampleville	N	EC-001	Address not known or invalid.
Testville	N	EC-002	Name and/or Date of Birth not found at Address provided.

DVS ASIC/MSIC

The result code for a DVS ASIC/MSIC check will be determined by a range of card numbers:

Card Number	Result

IC1000000 - IC1999999	Y
IC2000000 - IC2999999	D
IC3000000 - IC3999999	N

Specific expanded responses will be determined by the card number:

Card Number	Result	Code	Response
IC9999990	D	SI-101	Document invalid
IC8888880	N	SI-001	Name on Card does not match
IC8888881	N	SI-002	Date of Birth does not match
IC8888882	N	SI-003	Card Number does not match
IC8888883	N	SI-004	Expiry Date does not match

DVS birth certificate

The result code for a DVS birth certificate check will be determined by a range of certificate numbers.

N.B. The certificate number requirements vary between states.

State	Certificate Number	Result
ACT	1000000 - 1999999	Y
NSW	1000000 - 1999999	Y
NT	1000000 - 1999999	Y
QLD	1000000000 - 2000000000	Y
SA	1000000 - 1999999	Y
TAS	10000000000 - 19999999999	Y
VIC	1000000 - 1999999	Y
WA	10000000000 - 19999999999	Y
ACT	2000000 - 2999999	D
NSW	2000000 - 2999999	D
NT	2000000 - 2999999	D
QLD	2000000000 - 2999999999	D
SA	2000000 - 2999999	D
TAS	20000000000 - 29999999999	D
VIC	2000000 - 2999999	D

State	Certificate Number	Result
WA	20000000000 - 29999999999	D
ACT	3000000 - 3999999	N
NSW	3000000 - 3999999	N
NT	3000000 - 3999999	N
QLD	3000000000 - 39999999999	N
SA	3000000 - 3999999	N
TAS	30000000000 - 39999999999	N
VIC	3000000 - 3999999	N
WA	30000000000 - 39999999999	N

Specific expanded responses will be determined by the certificate number:

State	Certificate Number	Result	Code	Response
ACT	9999990	N	BC-001	Family Name does not match.
ACT	9999991	N	BC-002	Registration number does not match.
ACT	9999992	N	BC-003	Certificate number does not match.
ACT	9999993	N	BC-004	Date of birth does not match.
ACT	9999994	N	BC-005	Registration number does not match.
NSW	9999990	N	BC-006	Family Name does not match.
NSW	9999991	N	BC-007	Given Name does not match.
NSW	9999992	N	BC-008	Date of Birth does not match.
NSW	9999993	N	BC-009	Registration Number does not match.
NSW	9999994	N	BC-010	Certificate Number invalid.
NT	9999990	N	BC-003	Certificate number does not match.
NT	9999991	N	BC-005	Registration number does not match.
NT	9999992	N	BC-004	Date of birth does not match.
NT	9999993	N	BC-001	Family Name does not match.
NT	9999994	N	BC-002	Registration number does not match.
QLD	9999999990	N	BC-011	Registration date does not match.

State	Certificate Number	Result	Code	Response
QLD	9999999991	N	BC-003	Certificate number does not match.
QLD	9999999992	N	BC-004	Date of birth does not match.
QLD	9999999993	N	BC-012	Date of birth does not match.
QLD	9999999994	N	BC-002	Registration number does not match.
SA	9999990	N	BC-003	Certificate number does not match.
SA	9999991	N	BC-002	Registration number does not match.
SA	9999992	N	BC-005	Registration number does not match.
SA	9999993	N	BC-001	Family Name does not match.
SA	9999994	N	BC-004	Date of birth does not match.
TAS	99999999990	N	BC-011	Registration date does not match.
TAS	99999999991	N	BC-003	Certificate number does not match.
TAS	99999999992	N	BC-018	Certificate number does not match.
TAS	99999999993	N	BC-012	Given Name/s does not match.
TAS	99999999994	N	BC-001	Family Name does not match.
TAS	99999999995	N	BC-002	Registration number does not match.
VIC	9999990	N	BC-017	Document not found.
VIC	9999991	N	BC-018	Certificate Number does not match.
VIC	9999992	N	BC-019	Registration Number does not match.
VIC	9999993	N	BC-020	Given Name does not match.
VIC	9999994	N	BC-021	Both the Given and Family Name do not match.
WA	99999999990	N	BC-013	Registration number not found.
WA	99999999991	N	BC-012	Given Name/s does not match.
WA	99999999992	N	BC-014	Family Name does not match
WA	99999999993	N	BC-015	Date of birth does not match.
WA	99999999994	N	BC-016	Certificate number not found.
WA	99999999995	N	BC-022	Registration not found.

DVS centrelink

The result code for a DVS centrelink check will be determined by a range of CRNs.

CRN	Result
100000000A - 100000000Z	Y
200000000A - 200000000Z	D
300000000A - 300000000Z	N

Specific expanded responses will be determined by the CRN:

CRN	Result	Code	Response
999999999A	N	CO-001	Name does not match.
999999999B	N	CO-002	Date of Birth does not match.
999999999C	N	CO-003	CRN is invalid
999999999D	N	CO-004	Expiry Date does not match
999999999E	N	CO-005	Card Type does not match.
999999999F	N	CO-006	Input Card type not matched
999999999G	N	CO-007	Input Name mismatch

DVS change of name certificate

The result code for a DVS change of name certificate check will be determined by a range of certificate numbers.

N.B. The certificate number requirements vary between states.

State	Certificate Number	Result
ACT	1000000 - 1999999	Y
NSW	1000000 - 1999999	Y
NT	1000000 - 1999999	Y
QLD	1000000000 - 2000000000	Y
SA	1000000 - 1999999	Y
TAS	10000000000 - 19999999999	Y
VIC	1000000 - 1999999	Y
WA	10000000000 - 19999999999	Y
ACT	2000000 - 2999999	D
NSW	2000000 - 2999999	D
NT	2000000 - 2999999	D
QLD	2000000000 - 2999999999	D

State	Certificate Number	Result
SA	2000000 - 2999999	D
TAS	20000000000 - 29999999999	D
VIC	2000000 - 2999999	D
WA	20000000000 - 29999999999	D
ACT	3000000 - 1999999	N
NSW	3000000 - 1999999	N
NT	3000000 - 1999999	N
QLD	3000000000 - 39999999999	N
SA	3000000 - 1999999	N
TAS	30000000000 - 39999999999	N
VIC	3000000 - 1999999	N
WA	30000000000 - 39999999999	N

Specific expanded responses will be determined by the certificate number:

State	Certificate Number	Result	Code	Response
ACT	9999990	N	NC-001	Registration number does not match.
ACT	9999991	N	NC-002	Registration number does not match.
ACT	9999992	N	NC-003	Certificate number does not match.
ACT	9999993	N	NC-004	Date of birth does not match.
ACT	9999994	N	NC-005	Family Name does not match.
ACT	9999995	N	NC-022	Given Name does not match.
NSW	9999990	N	NC-006	Date of Birth does not match.
NSW	9999991	N	NC-007	Certificate Number invalid.
NSW	9999992	N	NC-008	Registration Number does not match.
NSW	9999993	N	NC-009	Given Name does not match.
NSW	9999994	N	NC-010	Family Name does not match.
NSW	9999995	N	NC-020	Both the Given and Family Name do not match.
NT	9999990	N	NC-001	Registration number does not match.
NT	9999991	N	NC-003	Certificate number does not match.

State	Certificate Number	Result	Code	Response
NT	9999992	N	NC-005	Family Name does not match.
NT	9999993	N	NC-004	Date of birth does not match.
NT	9999994	N	NC-002	Registration number does not match.
NT	9999995	N	NC-021	Incorrect certificate type.
QLD	9999999990	N	NC-001	Registration number does not match.
QLD	9999999991	N	NC-003	Certificate number does not match.
QLD	9999999992	N	NC-005	Family Name does not match.
QLD	9999999993	N	NC-011	Given Name/s does not match.
QLD	9999999994	N	NC-012	Date of event does not match.
SA	9999990	N	NC-001	Registration number does not match.
SA	9999991	N	NC-005	Family Name does not match.
SA	9999992	N	NC-004	Date of birth does not match.
SA	9999993	N	NC-003	Certificate number does not match.
SA	9999994	N	NC-002	Registration number does not match.
TAS	99999999990	N	NC-001	Registration number does not match.
TAS	99999999991	N	NC-005	Family Name does not match.
TAS	99999999992	N	NC-011	Given Name/s does not match.
TAS	99999999993	N	NC-012	Date of event does not match.
TAS	99999999994	N	NC-003	Certificate number does not match.
VIC	9999990	N	NC-013	Certificate Number does not match.
VIC	9999991	N	NC-014	Document not found.
VIC	9999992	N	NC-015	Registration Number does not match.
WA	99999999990	N	NC-016	Registration number not found.
WA	99999999991	N	NC-017	Family Name does not match
WA	99999999992	N	NC-011	Given Name/s does not match
WA	99999999993	N	NC-018	Date of birth does not match.
WA	99999999994	N	NC-019	Certificate number not found.

DVS citizenship certificate

The result code for a DVS citizenship certificate check will be determined by a range of stock numbers.

Stock Number	Result
10000000000 - 19999999999	Y
20000000000 - 29999999999	D
30000000000 - 39999999999	N

Specific expanded responses will be determined by the stock number:

Stock Number	Result	Code	Response
99999999990	N	CC-001	Name or Date of Birth does not match.
99999999991	N	CC-002	Document not found.

DVS death certificate

The result code for a DVS death certificate check will be determined by a range of certificate numbers.

N.B. The certificate number requirements vary between states.

State	Certificate Number	Result
ACT	1000000 - 1999999	Y
NSW	1000000 - 1999999	Y
NT	1000000 - 1999999	Y
QLD	1000000000 - 2000000000	Y
SA	1000000 - 1999999	Y
TAS	10000000000 - 19999999999	Y
VIC	1000000 - 1999999	Y
WA	10000000000 - 19999999999	Y
ACT	2000000 - 2999999	D
NSW	2000000 - 2999999	D
NT	2000000 - 2999999	D
QLD	2000000000 - 2999999999	D
SA	2000000 - 2999999	D
TAS	20000000000 - 29999999999	D
VIC	2000000 - 2999999	D

State	Certificate Number	Result
WA	20000000000 - 29999999999	D
ACT	3000000 - 1999999	N
NSW	3000000 - 1999999	N
NT	3000000 - 1999999	N
QLD	3000000000 - 39999999999	N
SA	3000000 - 1999999	N
TAS	30000000000 - 39999999999	N
VIC	3000000 - 1999999	N
WA	30000000000 - 39999999999	N

Specific expanded responses will be determined by the certificate number:

State	Certificate Number	Result	Code	Response
ACT	9999990	N	DC-001	Given Name/s does not match.
ACT	9999991	N	DC-002	Family Name does not match.
ACT	9999992	N	DC-003	Date of death does not match.
ACT	9999993	N	DC-004	Certificate number does not match.
ACT	9999994	N	DC-005	Registration number does not match.
NSW	9999990	N	DC-007	Given Name/s does not match.
NSW	9999991	N	DC-008	Family Name does not match.
NSW	9999992	N	DC-009	Date of Death does not match.
NSW	9999993	N	DC-010	Certificate Number invalid.
NSW	9999994	N	DC-011	Registration Number does not match.
NSW	8888880	D	DC-101	Certificate Number invalid.
NT	9999990	N	DC-001	Given Name/s does not match.
NT	9999991	N	DC-002	Family Name does not match.
NT	9999992	N	DC-003	Date of death does not match.
NT	9999993	N	DC-012	Incorrect certificate type.
NT	9999994	N	DC-005	Registration number does not match.

State	Certificate Number	Result	Code	Response
QLD	9999999990	N	DC-001	Given Name/s does not match.
QLD	9999999991	N	DC-002	Family Name does not match.
QLD	9999999992	N	DC-013	Date of death does not match.
QLD	9999999993	N	DC-014	Certificate number does not match.
QLD	9999999994	N	DC-015	Registration date does not match.
SA	9999990	N	DC-001	Given Name/s does not match.
SA	9999991	N	DC-002	Family Name does not match.
SA	9999992	N	DC-003	Date of death does not match.
SA	9999993	N	DC-004	Certificate number does not match.
SA	9999994	N	DC-005	Registration number does not match.
TAS	99999999990	N	DC-006	Registration number does not match.
TAS	99999999991	N	DC-014	Certificate number does not match.
VIC	9999990	N	DC-016	Document not found.
VIC	9999991	N	DC-017	Certificate Number does not match.
VIC	9999992	N	DC-018	Registration Number does not match.
VIC	9999993	N	DC-025	Given Name does not match.
WA	99999999990	N	DC-023	Given name does not match.
WA	99999999991	N	DC-019	Family Name does not match
WA	99999999992	N	DC-021	Date of event does not match.
WA	99999999993	N	DC-020	Certificate number not found.
WA	99999999994	N	DC-024	Registration number not found.
WA	99999999995	N	DC-026	Registration not found.

DVS drivers licence

The result code for a DVS driver licence check will be determined by a range of card numbers:

N.B. It is expected that the state is VIC for these specific responses.

Card Number	Result
10000000 - 19999999	Y

Card Number	Result
20000000 - 29999999	D
30000000 - 39999999	N

Specific expanded responses will be determined by the card number:

Card Number	Result	Code	Response
88888880	N	DL-001	Given Name/s does not match.
88888881	N	DL-002	Middle name does not match.
88888882	N	DL-003	Surname does not match.
88888883	N	DL-004	Date of birth does not match.
88888884	N	DL-005	Licence number does not match.
88888885	N	DL-006	Card number not matched.
88888886	N	DL-007	Given name does not match.
99999990	D	DL-101	Document Invalid.
77777770	Y	DL-201	Successful Match - Card Number has not been checked.

DVS immicard

The result code for a DVS immicard check will be determined by a range of card numbers:

Card Number	Result
ABC100000 - ABC199999	Y
ABC200000 - ABC299999	D
ABC300000 - ABC399999	N

Specific expanded responses will be determined by the card number:

Card Number	Result	Code	Response
ABC888880	N	IM-001	Travel document could not be found.
ABC888881	N	IM-002	Date of birth does not match.
ABC888882	N	IM-003	Family Name does not match.
ABC888883	N	IM-004	Document Invalid
ABC888884	N	IM-005	Both the Given and Family Name do not match.

DVS marriage certificate

The result code for a DVS marriage certificate check will be determined by a range of certificate numbers.

N.B. The certificate number requirements vary between states.

N.B. The registration field is state-specific. For **QLD**, the `registration_date` is used; for all **other states**, the `registration_number` is used. If you supply the field that does not apply to the requested state (e.g. a `registration_date` for a non-QLD certificate, or a `registration_number` for a QLD certificate), it is silently ignored and not forwarded to DVS. Note that a supplied `registration_date` must still be in `YYYY-MM-DD` format even when it is going to be ignored, otherwise the request is rejected.

State	Certificate Number	Result
ACT	1000000 - 1999999	Y
NSW	1000000 - 1999999	Y
NT	1000000 - 1999999	Y
QLD	1000000000 - 2000000000	Y
SA	1000000 - 1999999	Y
TAS	10000000000 - 19999999999	Y
VIC	1000000 - 1999999	Y
WA	10000000000 - 19999999999	Y
ACT	2000000 - 2999999	D
NSW	2000000 - 2999999	D
NT	2000000 - 2999999	D
QLD	2000000000 - 2999999999	D
SA	2000000 - 2999999	D
TAS	20000000000 - 29999999999	D
VIC	2000000 - 2999999	D
WA	20000000000 - 29999999999	D
ACT	3000000 - 1999999	N
NSW	3000000 - 1999999	N
NT	3000000 - 1999999	N
QLD	3000000000 - 39999999999	N
SA	3000000 - 1999999	N
TAS	30000000000 - 39999999999	N

State	Certificate Number	Result
VIC	3000000 - 1999999	N
WA	30000000000 - 39999999999	N

Specific expanded responses will be determined by the certificate number:

State	Certificate Number	Result	Code	Response
ACT	9999990	N	MC-001	Family Name does not match.
ACT	9999991	N	MC-002	Date of marriage does not match.
ACT	9999992	N	MC-007	Certificate number does not match.
ACT	9999993	N	MC-003	Registration number does not match.
ACT	9999994	N	MC-004	Registration number does not match.
ACT	9999995	N	MC-017	Both the Given and Family Name do not match.
ACT	9999996	N	MC-022	Given Name does not match.
NSW	9999990	N	MC-005	Date of Marriage does not match.
NSW	9999991	N	MC-020	Certificate Number invalid.
NSW	9999992	N	MC-021	Given Name does not match.
NSW	9999993	N	MC-023	Family Name does not match.
NSW	9999994	N	MC-024	Registration not found.
NT	9999990	N	MC-001	Family Name does not match.
NT	9999991	N	MC-002	Date of marriage does not match.
NT	9999992	N	MC-015	Incorrect certificate type.
NT	9999993	N	MC-003	Registration number does not match.
NT	9999994	N	MC-004	Registration number does not match.
QLD	9999999990	N	MC-004	Registration number does not match.
QLD	9999999991	N	MC-006	Given Name/s does not match.
QLD	9999999992	N	MC-007	Certificate number does not match.
SA	9999990	N	MC-004	Registration number does not match.
TAS	99999999990	N	MC-007	Certificate number does not match.
TAS	99999999991	N	MC-001	Family Name does not match.

State	Certificate Number	Result	Code	Response
TAS	99999999992	N	MC-002	Date of marriage does not match.
TAS	99999999993	N	MC-003	Registration number does not match.
TAS	99999999994	N	MC-004	Registration number does not match.
TAS	99999999995	N	MC-001	Family Name does not match.
TAS	99999999996	N	MC-006	Given Name/s does not match.
TAS	99999999997	N	MC-008	Date of marriage does not match.
VIC	9999990	N	MC-016	Document not found.
VIC	9999991	N	MC-010	Registration Number does not match.
VIC	9999992	N	MC-009	Certificate Number does not match.
VIC	9999993	N	MC-018	Given Name for Person 1 or Person 2 does not match.
VIC	9999994	N	MC-019	Family Name for Person 1 or Person 2 does not match.
WA	99999999990	N	MC-011	Registration number not found.
WA	99999999991	N	MC-012	Certificate number not found.
WA	99999999992	N	MC-013	Family Name does not match
WA	99999999993	N	MC-006	Given Name/s does not match.
WA	99999999994	N	MC-014	Date of event does not match.

DVS medicare

The result code for a DVS medicare check will be determined by a range of card numbers:

Card Number	Result
2000000020 - 2999999939	Y
3000000030 - 3999999949	D
4000000040 - 4999999959	N

Specific expanded responses will be determined by the card number:

Card Number	Result	Code	Response
6000000060	N	MD-001	Name does not match.
6000000061	N	MD-002	Date of Birth does not match.
6000000062	N	MD-003	Card Number does not match.

Card Number	Result	Code	Response
6000000063	N	MD-004	Expiry Date does not match.
6000000064	N	MD-005	Card Number does not match.
6000000065	N	MD-006	Card Type does not match.

DVS passport

The result code for a DVS passport check will be determined by a range of travel document numbers:

Travel Document Number	Result
M1000000 - M1999999	Y
M2000000 - M2999999	D
M3000000 - M3999999	N

Specific expanded responses will be determined by the travel document number:

Travel Document Number	Result	Code	Response
M9999990	D	PP-101	Document Invalid.
M9999991	D	PP-102	Document number does not match.
M9999992	N	PP-001	Gender does not match.
M9999993	N	PP-002	Date of birth does not match.
M9999994	N	PP-003	Family name does not match.
M9999995	N	PP-004	Given Name/s does not match
M9999996	N	PP-005	Given Name does not match.

DVS registration by descent certificate

The result code for a DVS registration by descent certificate check will be determined by a range of stock numbers.

Stock Number	Result
10000000000 - 19999999999	Y
20000000000 - 29999999999	D
30000000000 - 39999999999	N

Specific expanded responses will be determined by the stock number:

Stock Number	Result	Code	Response
99999999990	N	RD-001	Name or Date of Birth does not match.

Stock Number	Result	Code	Response
99999999991	N	RD-002	Document not found.

DVS visa

The result code for a DVS visa check will be determined by a range of passport numbers:

Passport Number	Result
M1000000 - M1999999	Y
M2000000 - M2999999	D
M3000000 - M3999999	N

Specific expanded responses will be determined by the passport number:

Passport Number	Result	Code	Response
M9999990	N	VI-001	Travel document could not be found.
M9999991	N	VI-002	Date of birth does not match.
M9999992	N	VI-003	Family Name does not match.
M9999993	N	VI-004	Both the Given and Family Name do not match.

Out of Scope Documents

Some documents cannot be checked using the DVS system. This can be due to several reasons:

- The documents are too old
- The documents are part of a set which has not been digitised by the responsible jurisdiction.
- Some quirk of the name or address formatting.

The list below outlines the current known out of scope documents by document type and jurisdiction.

These are guidance, not rules the API enforces. They describe what each jurisdiction typically holds in a form the DVS can check, and are useful for understanding why a document came back unverified. They are not applied as validation: a request for a document in one of the ranges below is still sent to the DVS, because a record from outside the usual range may have been captured electronically and will match. Equally, a document outside these ranges is not guaranteed to verify. Treat the list as the likely explanation for a no-match rather than as a pre-flight check - rejecting a document on these ranges alone would turn away documents the DVS would have verified.

Birth Certificates

All Jurisdictions:

- Births registered in Papua New Guinea - The Territory of Papua and New Guinea was considered an external territory of Australia before becoming an independent nation in 1975

ACT

- Birth certificates issued (printed) prior to 1 January 1930 - Certificates issued prior to this were issued by NSW Registry of Births, Deaths and Marriages and could be obtained from there.

NSW

- Birth certificates prior to 1 January 1914
- Birth records occurring prior to 1952 that contain a district number only

NT

- Birth certificates issued (printed) prior to 1 January 1870

QLD

- Birth certificates where the birth occurred prior to 1 January 1941 and the certificate has not been re-issued since 27 June 2016

SA

- Birth certificates issued (printed) prior to 1 January 1917
- District certificates issued for births only by/in the following districts:
    - District of Adelaide
    - District of Angaston
    - District of Burra
    - District of Clare
    - District of Daly
    - District of Encounter Bay
    - District of Flinders
    - District of Frome
    - District of Gawler
    - District of Gilbert
    - District of Grey
    - District of Hindmarsh
    - District of Kapunda
    - District of Mount Barker
    - District of Murray
    - District of Norwood
    - District of Pirie
    - District of Port Adelaide
    - District of Robe
    - District of Talunga
    - District of Wilunga

TAS

- Birth certificates where the birth was registered prior to 1 January 1965 and the certificate has not been re-issued since 1 January 2000
- Single name birth certificates

VIC

- Birth certificates issued (printed) prior to 1 January 1929
- Birth certificates where a change of name, correction or legitimation occurred prior to and during 1993 - these records were re-registered and were allocated a different number and registration year and will not verify through the DVS.

WA

- Birth certificates issued prior to 1 January 1841
- Birth certificates registered in the Cocos Islands prior to 1977
- Birth certificates issued by/in Christmas Island prior to 1 July 1993

ID Pass Overview

ID Pass is a secure, configurable identity verification service for Australian KYC checks.

You create a one-time verification link, your customer completes a guided online check, and you retrieve a clear pass/fail result plus the verified identity details.

This guide explains how ID Pass works, what your customers will experience, and how to integrate it at a high level.

- For full field-by-field definitions, see the ID Pass API Reference.
- For a customer journey walkthrough, see the ID Pass User Journey.

Document verification strategies

ID Pass supports three document verification strategies, selected via the `document_verification` parameter when creating an ID Pass:

- **DVS** (default) - Full verification of Australian documents via the Australian Document Verification Service (DVS), with the DVS result codes and details returned to you. A New Zealand Driver Licence under this strategy is verified with the New Zealand licence issuer instead, and its result is returned in the same shape.
- **IDSP** - Document verification is performed by Global Data on your behalf. An **identity opinion** (valid / not valid) is returned; DVS result codes and details are not returned in the API response.
- **Basic** - Format and checksum validation only. No formal document verification is performed.

See the dedicated section for each mode below for requirements, behaviour, and what the API returns.

DVS mode

In `dvs` mode (the default), each Australian identity document is verified directly against the Australian Document Verification Service (DVS). The full DVS result - including the result code and any mismatch detail - is returned to you on the response. The one exception is the New Zealand Driver Licence, which is verified with the New Zealand licence issuer rather than DVS; its result is returned in the same shape (see New Zealand Driver Licence).

This is the right mode for most KYC use cases where you want full visibility into the DVS outcome.

Requirements

- A DVS Identity (OAC) on your account for live environments. Registration is a free process available for Australian businesses - contact your Global Data account manager to register for the DVS.
- No DVS Identity is required in sandbox.

What is verified

Each document is checked against the authoritative issuer database. See Biographic Data Verification for the list of authorities and what fields are submitted.

What the API returns

The default response includes the top-level `verification_status` (`passed` when all documents match and cross-document identity data is consistent, `failed` otherwise), a `verification_description` , and a decrypted `validation_summary` with strategy-agnostic per-document results (`biographic_validated` , `biometric_validated` , `validated_fields` , `changes` , and similar).

To retrieve the raw DVS detail per document, pass `return_documents: true` on the details request. The response then also contains a `documents` object, and each `documents.document_N.validation_result` includes:

- `strategy` - `dvs`
- `verification_result_code` - DVS result: `Y` (match), `N` (no match), or `D` (document invalid)
- `verification_request_number` - DVS request identifier, required for any official queries on the result
- `additional_information` - detail about any mismatch (e.g. "Card number did not match")
- `originating_agency_code` - the OAC used for the request
- `checked_at` - timestamp of the DVS call

See the ID Pass - DVS mode quick start for example payloads, or the ID Pass API Reference for the full response schema.

IDSP mode

In `idsp` mode, Global Data returns an **identity opinion** - a single valid / not valid assessment produced by the combined checks of Global Data's Document Validation System and DVS. Global Data runs the document verification on your behalf and provides you with an identity opinion on whether the documents are valid or not.

This mode is intended for customers who are not registered with DVS but need a compliant identity verification service. Global Data acts as an Identity Service Provider (IDSP) on your behalf, and the identity opinion returned is Global Data's own assessment, not a DVS response.

Requirements

- Liveness detection must be enabled (`check_liveness` set to `true`). This ensures every verification is based on multiple independent checks - liveness, biometric face match (where a photo ID is supplied), and document verification - which together form the basis of a identity opinion.
- Your account must be enrolled as an IDSP Service Client. Contact your Global Data account manager to arrange enrolment.
- No DVS registration is required to use IDSP mode.

Multi-stage document validation

In `idsp` mode, each document is put through a number of checks, including Global Data's Document Validation System and a DVS check. The combined results of these checks determine the verification result for the document, which in turn contributes to Global Data's overall identity opinion for the ID Pass.

What the API returns

The default response includes the top-level `verification_status` , which carries Global Data's identity opinion: `passed` (valid) or `failed` (not valid). It also includes a `verification_description` and a decrypted `validation_summary` with per-document `biographic_validated` , `biometric_validated` , `validated_fields` , and similar friendly fields.

To retrieve the per-document verification boolean, pass `return_documents: true` on the details request. The response then also contains a `documents` object, and each `documents.document_N.validation_result` includes:

- `strategy` - `idsp`
- `verification_passed` - boolean (valid / not valid) for that document
- `checked_at` - timestamp

DVS result codes, DVS extended result codes, DVS error codes, DVS request identifiers, and the OAC used are not returned in the API response in any mode.

See the ID Pass - IDSP mode quick start for example payloads.

Basic mode

In `basic` mode, ID Pass performs format and checksum validation on the supplied document fields but does not perform any external verification. No DVS call is made, in either live or sandbox.

This mode is useful when you need OCR extraction, liveness, and biometric face match but intend to perform document verification yourself, or when DVS verification is not required for your use case.

Requirements

- No DVS Identity or special enrolment is required.

What is validated

Validation rules are applied to the fields supplied (or extracted by OCR) for each document:

- Document number formats (licence number patterns, passport number patterns, visa number patterns, and similar)
- Card number patterns and checksums, per jurisdiction where applicable
- Medicare card checksums and expiry-date formatting
- Centrelink CRN format
- Name field formats and lengths
- Date of birth format

What the API returns

The default response includes the top-level `verification_status` (`passed` when all documents pass format validation and cross-document identity data is consistent, `failed` otherwise), a `verification_description`, and a decrypted `validation_summary` with per-document `biographic_validated` and `biometric_validated` flags.

To retrieve the per-field validation errors for a failed document, pass `return_documents: true` on the details request. The response then also contains a `documents` object, and each `documents.document_N.validation_result` includes:

- `strategy` - `basic`
- `basic_validation_passed` - boolean
- `errors` - per-field validation errors (empty when validation passed)
- `checked_at` - timestamp

See the ID Pass - Basic mode quick start for example payloads.

What ID Pass verifies

The ID Pass service performs a number of verification steps to ensure the identity of the person is verified.

Liveness

Liveness detection ensures the person verifying their identity is physically present.

This step involves asking the customer to use their mobile camera or webcam to capture a short video of their face. This video is reviewed by an AI-driven liveness detector to ensure that the video is a real-time video of a human face.

The detector can not be fooled by presenting a previously recorded video of a face, a photo, screen, or mask, etc.

Identity Documents

The customer is asked to take a photo or upload a copy of their identity document.

ID Pass currently supports the following identity documents:

- Australian Driver's Licence
- Australian Passport
- Medicare Card
- Foreign Passport (for Australian Visa holders)
- Centrelink Card (manual entry, no OCR)
- Australian Birth Certificate (manual entry, no OCR)
- New Zealand Driver Licence

The most common use case is to verify two documents where at least one is a photo ID. For example:

- Document 1: Drivers Licence, Passport or Visa
- Document 2: Drivers Licence, Passport, Visa, Medicare card, Centrelink Card, Birth Certificate or New Zealand Driver Licence

Note: A document which has already been verified cannot be used again for a subsequent verification. So if the customer has already used their licence as the first document, then they cannot use it again for the second document.

When more than one document is verified, ID Pass also cross-checks that the name and date of birth are consistent across documents and fails the verification if they do not match. See the Name Matching section for how names are compared.

Biographic Data Verification

For strategies that perform DVS verification (`dvs` and `idsp`), each document is checked against the authoritative issuer database. This provides a source of truth for the biographic information on the document.

- **Passports** are verified against the **Department of Foreign Affairs and Trade** (DFAT) database.
- **Australian driver licences** are verified against the **National Exchange of Vehicle and Driver Information System** (NEVDIS). A New Zealand Driver Licence is verified with the New Zealand licence issuer.
- **Medicare cards** are verified against the **Services Australia** database.
- **Visas** are verified against the **Department of Home Affairs** database.
- **Centrelink cards** are verified against the **Services Australia** database.
- **Birth certificates** are verified against the **Registry of Births, Deaths and Marriages** for the state or territory the birth was registered in.

In `basic` mode, no external verification is performed - only document field formats are validated.

Biometric Face Verification

During the liveness check, an image of the customers face (the probe image) is captured and compared to the image of the face on the identity document (the ID photo) using a biometric face verification algorithm.

This step stops a person from presenting another person's identity document by comparing the probe image with the photo on the identity document (eg the licence or passport photo) to ensure they are the same person.

How it works

1. Create an ID Pass link

You call `idpass/register` and choose how many documents to collect, whether to run a liveness check, link expiry, and other options. You receive a secure link, an ID, and a cipher key.

2. Send the link to your customer

If you are inlining the ID Pass process then you can just redirect the customer to the link, otherwise you can send the link via SMS or email. The customer opens it on mobile (recommended) or desktop. If they start on desktop, they'll be prompted to scan a QR code to continue on mobile.

3. Customer completes the verification

ID Pass guides them through consent, selfie liveness (if enabled), and 1, 2 or 3 document checks. They can review and correct OCR-read details before submission (for centrelink, details are entered manually).

4. You receive webhook updates

As the customer progresses, ID Pass can notify you of status changes like `opened`, `in_progress`, `complete`, or `failed`.

5. Retrieve the result

Call `idpass/details` any time (usually after complete) to get the final pass/fail decision and verified identity details.

What your customer experiences

ID Pass is a step-by-step flow optimised for mobile devices, but it will work on desktop with a web camera.

- **Introduction screen** shows what they need to do (liveness, number of documents, optional ID photo).
- **Consent screen** includes your company name and privacy policy link.
- **Liveness selfie video** (if enabled) verifies they are a real person.
- **Document capture** for each required document:
  - take or upload photos,
  - ID Pass OCR engine scans the document and reads the details,
  - customer reviews/edits if needed,
  - ID Pass verifies with DVS (or, for a New Zealand Driver Licence, with the New Zealand licence issuer).

    - for Centrelink card and Birth Certificate, no photo upload/OCR is used; the customer enters details manually before DVS validation.

- **Optional ID photo capture** (if you request it).
- **Completion / error screens** guide them to retry if possible, or stop if limits are exceeded.

See the ID Pass User Journey for more a complete walkthrough of the ID Pass process from start to finish.

Identified and Anonymous ID Passes

You can create an ID Pass **with or without Personally Identifiable Information** (PII).

The **preferred option** is to use an **identified ID Pass**, in this case you send the name (and optionally the date of birth) of the customer you intend to identify in the `idpass/register` call. The ID Pass service will then compare the supplied name and DOB to the verified documents, and fail the verification if they don't match.

> Note: The customer identity you supply is never displayed to the customer, it is only used for verification purposes.

The **alternative option** is to use an **anonymous ID Pass**, in this case you do not send any PII to the ID Pass service. After the customer has completed the verification, you will need to check the returned `verified_identity` field to confirm it matches the intended customer's identity.

These options exist to support use cases where you are not permitted to send any customer PII to the Global Data ID Pass service. Check with your legal team to confirm which option is permitted in your organisation.

Identified: With name and date of birth

You provide your customer's name and DOB in the `idpass/register` call inside the `requested_identity` object.

ID Pass will compare the supplied customer name and DOB to the verified documents, and fail the verification if they don't match.

Use this when you want the service to enforce identity matching automatically.

Example:

```
{
  "document_1_allowed_types": ["licence", "passport"],
  "document_2_allowed_types": ["licence", "passport"],
  "document_verification": "dvs",
  "requested_identity": {
    "first_name": "Sarah",
    "middle_name": "Jane",
    "last_name": "Smith",
    "date_of_birth": "1980-01-01"
  }
}
```

The middle name field is optional, see the API reference for more details.

The date of birth field is also optional. If you omit it (or send null), only the name is matched against the verified documents. In this case you should check the returned `verified_identity.date_of_birth` yourself to confirm it matches your customer.

See the Name Matching section for more details on how name matching works.

> Note: The requested identity data you supply is encrypted with the cipher key and stored in the ID Pass record. It is never displayed to the customer, and is only used for verification purposes.

Anonymous: No PII sent to ID Pass service

You don't provide a name or date of birth in the `idpass/register` call.

After the ID Pass has completed, you check the returned `verified_identity` field to confirm that it matches the intended customer's identity.

Use this when you prefer not to send PII to Global Data.

Example:

```
{
  "document_1_allowed_types": ["licence", "passport"],
  "document_2_allowed_types": ["licence", "passport"],
  "document_verification": "dvs"
}
```

What you can configure

When creating an ID Pass, you can choose:

- **Liveness check**: on or off (must be enabled for `idsp` verification strategy)
- **Number of required documents**: 1, 2, or 3
- **Allowed document** types per document: licence, passport, medicare, visa, centrelink, birth_certificate, nz_licence
- **Optional ID photo** step
- **Link validity period** (1-14 days, default 7)
- **Image retention period** (how long captured images are stored)
- **Webhook URL + events**
- **Return URL** to send the customer back to your app after completion
- **Optional name + DOB matching** (see above)

Quick start integration

Create an ID Pass

Call the `idpass/register` endpoint with your preferred configuration.

You will receive a response containing:

- **id** (ID Pass identifier)
- **link** (the secure link to send to your customer)
- **cipher_key** (needed to decrypt sensitive fields later)

Send the link to your customer

Provide the `link` to your customer. If you are inlining the ID Pass process in your application flow then you can just redirect the customer to the link, otherwise you can send the link via SMS or email. The customer opens it on mobile (recommended) or

desktop. If they start on desktop, they'll be prompted to scan a QR code to continue on mobile.

Track progress

When creating the ID Pass, you can subscribe to webhooks for real-time updates. When the status changes, you'll get a simple payload posted to your webhook URL.

```
{
  "id": "19517924-94f0-4000-8e02-afba96ad5101",
  "status": "complete"
}
```

You can use this information to update your system with the current status of the ID Pass or call the `idpass/details` endpoint to retrieve the full pending or completed results.

Retrieve results

Once the ID Pass has completed, you can call the `idpass/details` endpoint with the `id` to retrieve the completed ID Pass results.

If you included the `cipher_key` when creating the ID Pass, then the sensitive fields will be decrypted for you in the response. If you did not include the `cipher_key`, then the response will contain the encrypted values for you to decrypt locally.

The response will contain the following information:

- Overall ID Pass verification status and description
- Document validation results
- Biometric face match scores
- OCR extracted data with customer modifications
- Detailed verification logs
- Biometric images
- Document images

Using the results

See the API reference for full field-by-field definitions of the response fields.

The most important returned fields are:

- Verification status
- Verification description
- Verified identity
- Consent

Verification status

The `verification_status` field contains the final identification pass/fail decision:

- `passed` - identity verified successfully
- `failed` - identity could not be verified
- `null` - still in progress

Verification description

The `verification_description` field contains a human-readable reason for the verification status:

If verification fails (or includes important notes), you'll get a plain-English explanation.

Example:

```
{
  "verification_status": "failed",
  "verification_description": "
    The name on the Medicare Card does not match the name on the Australian Drivers Licence\n
    Your customers date of birth does not match the date of birth on the Australian Drivers Licence"
}
```

Verified identity

When verification passes, the `verified_identity` contains the name and DOB extracted from the verified documents. This is especially useful if you created the ID Pass anonymously and need to confirm the identity of the customer.

Example of the `verified_identity` field in the response:

```
{
  "validation_summary": {
    ...
    "verified_identity": {
      "first_name": "John",
      "middle_name": "Andrew",
      "last_name": "Johnson",
      "date_of_birth": "1980-01-01"
    }
    ...
  }
}
```

Consent

The consent fields contain the exact text of the consent that was displayed to the customer and the date and time when the customer gave consent to the verification process.

Example of the consent fields in the response:

```
{
  "consent_text": "I confirm that I am authorised to provide my personal details...",
  "consent_given_at": "2024-05-16T01:35:11+00:00"
}
```

> Note: It is important for you to store the consent details for audit and compliance purposes in line with the DVS requirements.

Billing

Calls to the ID Pass register endpoint are charged per call based on the number of documents requested.

The fee differs depending on the number of documents requested:

- 1 document
- 2 documents
- 3 documents

Calls to the ID Pass details endpoint are not charged.

Please refer to your commercial agreement or pricing documentation for details.

> Note: Calls to the ID Pass service in the sandbox environment are not charged.

Privacy and zero-knowledge encryption

ID Pass uses **zero-knowledge encryption** for sensitive data:

- Every ID Pass generates a unique `cipher_key`.
- Fields containing Personally Identifiable Information (PII) and all images are encrypted using that key.
- The key is returned to you in the create response.
- After the customer completes the check, the key is deleted from Global Data systems.

That means **only you can decrypt the sensitive parts of the result**, and you must store `cipher_key` securely. If it's lost, the encrypted data cannot be recovered.

Decrypting results

When calling the `idpass/details` endpoint to read results you can provide the `cipher_key` in the request. If you do, the ID Pass service will decrypt the sensitive fields and return the decrypted values over the HTTPS API response.

Note that this data is always encrypted at rest on the Global Data servers so you must provide the `cipher_key` with each request to decrypt the data.

Local decryption

If you prefer not to provide the `cipher_key` in the `idpass/details` request, then the response will contain the encrypted values for you to decrypt locally.

The data is encrypted using the AES-256-CBC cipher. Use the following pseudocode to decrypt the data locally:

```
function decryptData(encryptedData, cipherKey, iv):
    // Decode the cipher key and initialization vector (IV)
    decodedKey = base64Decode(cipherKey)
    decodedIV = base64Decode(iv)

    // Decrypt the data using AES-256-CBC
    decryptedData = AES256_CBC_Decrypt(encryptedData, decodedKey, decodedIV)

    return decryptedData
```

You can also validate the integrity of the data by validating the MAC.

```
function validateMAC(data, iv, cipherKey):
    // Concatenate the IV and encrypted data
    combinedData = iv + data
```

```
    // Generate a MAC using HMAC-SHA256
    mac = HMAC_SHA256(combinedData, base64Decode(cipherKey))

    return mac
```

Limits and retries (customer-visible behaviour)

These limits help protect against fraud and repeated guessing:

- **Face liveness attempts:** dynamically limited based on risk signals, with **at least 3 attempts** before the flow stops.
- **Document verification attempts:** 3 attempts per document.

If a customer hits one of these limits, the ID Pass will show the customer an error screen and return a failed status. You'll need to create a new ID Pass to allow the customer to try again.

Webhooks

Webhooks let you keep your system in sync as the customer progresses.

To subscribe to webhooks, pass the webhook URL and the list of events as part of the `idpass/register` API call.

Supported events

You can configure a webhook URL to receive notifications when specific events occur. The supported events are:

- `**opened` ** – The ID Pass process has started.
- `**in_progress**` – Consent has been provided.
- `**complete**` – The ID Pass process is successfully completed.
- `**expired**` – The ID Pass has expired without being completed.
- `**failed**` – The ID Pass process failed.
- `**cancelled**` – The ID Pass was manually cancelled.

Example configuration

```
{
  "webhook_url": "https://example.com/your/webhook/path",
  "webhook_events": ["opened", "complete"]
}
```

> We recommend subscribing to all events so you don't miss state changes.

Example webhook payload

When an event occurs, The ID Pass service will send a **POST** request to the configured webhook URL with a JSON payload.

```
{
    "id": "19517924-94f0-4000-8e02-afba96ad5101",
    "status": "complete"
}
```

Parameter	Type	Description
id	string	The unique identifier of the ID Pass. This corresponds to the ID received in the idpass/register API response.
status	string	The new status of the ID Pass, indicating the event that triggered the webhook.

Webhook response and retries

ID Pass expects a **200 OK** HTTP response from your webhook endpoint. If the request fails, the system will retry up to **3 times** before marking the webhook as failed.

Condition	Action
Success (200 OK)	Webhook is marked as delivered.
Failure (non-2xx response)	Webhook is retried up to 3 times.
All retries fail	The webhook is marked as failed, and no further attempts are made.

To ensure webhook requests originate from ID Pass, allow requests only from the IP addresses listed in the IP Whitelisting for Webhooks documentation.

ID Photo

The ID Pass service can optionally ask the customer to capture an ID photo of themselves.

Use this option if you are producing an ID card with the users photo or require a photo of the user for other purposes.

> **Note: Never use the liveness probe image as the ID photo!** The probe image is optimised for biometric face verification and is not suitable for use as an ID photo.

To capture an ID photo, you can set the `capture_id_photo` parameter to `true` in the `idpass/register` request.

You will also need to set the `id_photo_purpose` parameter to specify the purpose of the ID photo. This text will be displayed to the customer on the page where they are asked to capture the ID photo. The `id_photo_purpose` value must be a string between 3 and 255 characters.

```
{
    "document_1_allowed_types": ["licence", "passport", "visa"],
    "document_2_allowed_types": ["licence", "passport", "visa"],
    "require_id_photo": true,
    "id_photo_purpose": "the SampleCo Membership Card"
}
```

The captured ID photo will be verified using a biometric face verification algorithm against the face in the liveness check probe image to ensure they are the same person.

The ID Photo is also checked for a number of different criteria to ensure the photo is valid.

Check	Message shown to the customer
Invalid image file	We could not process your ID photo. Please try again.
No face found	Could not find a face in the image. Please ensure your face is visible.

Check	Message shown to the customer
Multiple faces found	Multiple faces found in the image. Please ensure only your face is visible.
Face occluded	Please ensure your face is fully visible.
Pitch too high	Please tilt your head down slightly - your chin is too high.
Pitch too low	Please raise your chin slightly - you are looking too far down.
Yaw too high	Please turn your head slightly to the right - you are facing too far left.
Yaw too low	Please turn your head slightly to the left - you are facing too far right.
Roll too high	Please straighten your head - it is tilted too far to the left.
Roll too low	Please straighten your head - it is tilted too far to the right.
Brightness too low	Please ensure the image is well lit.
Sharpness too low	The image is too blurry.
Eyes closed	Please ensure your eyes are open.
Sunglasses	Please remove your sunglasses.
Biometric Check Failed	This does not appear to be a photo of you

The photo will only be accepted when all of the checks are passed.

The captured ID photo image and cropped face image are available in the `idpass/details` response in the `images` field.

The cropped face image is a passport style cropped image of the face in the ID photo.

```
{
    "images": {
        "id_photo": {
            "image": "base64 encoded JPEG image",
            "cropped_image": "base64 encoded JPEG image"
        }
    }
}
```

Logs

The ID Pass service logs the validation process and these logs are available to you in the `idpass/details` response.

To return the logs in the `idpass/details` response, you can set the `return_logs` parameter to `true`.

```
{
    "id": "19517924-94f0-4000-8e02-afba96ad5101",
    "return_logs": true
}
```

The logs are returned in the `logs` field of the `idpass/details` response.

```
{
    "logs": [
        "2024-05-16T01:31:23+00:00 - 203.22.251.3 - Starting IdPass flow",
        "2024-05-16T01:35:11+00:00 - 203.22.251.3 - Consent given"
        ...
    ]
}
```

Verification Notes

Name Matching

If a name is supplied when creating the ID Pass, then the ID Pass service will check that the supplied name matches the name on the identification documents.

When two or more documents are verified, the ID Pass service will also check to see if the names on the documents are the same.

Name matching allows for some difference in middle names, however first and last names must be the same.

Name 1	Name2	Name Match	Description
Jane Johnson	Jane Johnson	Pass	The names are the same
Jane Johnson	Jane Mary Johnson	Pass	Additional middle name is permitted
Jane M Johnson	Jane Mary Johnson	Pass	Middle name initials match
Jane Mary Susan Johnson	Jane Susan Mary Johnson	Pass	Middle names in different order
Robert Johnson	Mary Johnson	Fail	First name does not match
Mary Smith	Mary Johnson	Fail	Last name does not match
Mary Susan Smith	Mary Beverly Smith	Fail	Middle name does not match

When a name match between the supplied name and the names on the documents fails, or when multiple documents are verified and the names on the documents are different, then the verification status is set to `failed`.

Passport and Visa

Passport and Visa verification is based on the following fields:

- Given Name(s)
- Last Name
- Date of Birth
- Passport Number

Note that passports and visas use multiple given names instead of a first / middle name format.

Therefore data returned from a passport or visa verification will have the following structure:

```
{
    "first_name": "John Andrew",
```

```
    "middle_name": null,
    "last_name": "Johnson",
    ...
}
```

Note that the middle name is included in the first name field and the middle name field is set to null.

Licence

Drivers licence verification is based on the following fields:

- First Name
- Last Name
- Date of Birth
- Licence Number
- Card Number
- Licence State

Note that the middle name is not used in the validation process.

This is because some jurisdictions store only the initial of the middle name. This causes matching issues when the customer enters their correct full middle name but the verification service only has the initial. The DVS service recommends that the middle name is not used in the validation process.

For this reason the ID Pass service will only match the first and last name and date of birth.

New Zealand Driver Licence

The New Zealand Driver Licence (`nz_licence`) is captured front and back like an Australian licence, and the customer reviews the following fields:

- First Name
- Middle Name
- Last Name
- Date of Birth
- Licence Number (two letters followed by six digits, e.g. `DB512036` ; the check digit is validated before submission)
- Version Number (exactly 3 digits, printed beside the licence number on the front of the card)

There is no licence state and no card number. The result is returned in the same shape as an Australian licence under the `dvs` strategy:

- Completed checks return `verification_result_code` `Y` or `N` . `D` is never returned for this document type. If the issuer's source is unavailable or the check cannot be completed, the code is null (no verdict) and the customer may retry with the same details.
- `originating_agency_code` is null.
- `verification_request_number` is the reference for the check and can be quoted in any query about the result.

Medicare

Medicare verification is based on the following fields:

- Full Name

- Date of Birth
- Card Number
- Individual Reference Number

Note that the Date of Birth is required for verification but is not printed on the Medicare card. Therefore manual entry is required for this field.

Data returned from a Medicare verification will have the following structure:

```
{
    "full_name": "John Andrew Johnson",
    "date_of_birth": "1980-01-01",
    ...
}
```

Centrelink

Centrelink verification is based on the following fields:

- Full Name
- Date of Birth
- CRN
- Card Type
- Card Expiry

Centrelink card details are entered manually in the ID Pass flow and are validated with DVS.

Note that some Health Care Cards print the name with the full middle name. However, the name should be entered with the middle name as an initial in order to get a successful match.

Data returned from a Centrelink verification will have the following structure:

```
{
    "full_name": "John Andrew Johnson",
    "date_of_birth": "1980-01-01",
    ...
}
```

Birth Certificate

Birth certificate verification is based on the following fields:

- Given Name/s
- Family Name
- Date of Birth
- State or Territory of Registration
- Registration Number, or Registration Date for Queensland and Tasmania
- Certificate Number

Birth certificate details are entered manually in the ID Pass flow and are validated with DVS. The certificate is verified against the registry for the state or territory the birth was **registered** in, which is not necessarily where the certificate was most recently printed.

Each registry identifies the record differently. Queensland and Tasmania are identified by the date the birth was registered; every other state and territory by a registration number. Where a registration number is printed in a form that is not what the DVS expects, ID Pass converts it before sending. Direct API callers are expected to send the value the DVS expects - see the Birth Certificate DVS endpoint for the per-state rules.

The certificate number and private sector callers

Private sector DVS users - which includes all Global Data API customers - must supply the certificate number for NT, SA and ACT certificates issued on or after these dates:

State	Certificate number required for certificates issued from
NT	12 July 1999
SA	1 November 1999
ACT	1 May 2002

This is a rule of the DVS, not of this API. Our validation deliberately does not enforce it: the base DVS rule accepts **either** a registration number or a certificate number, and tightening it here would reject requests that are legitimately valid for certificates issued before those dates.

The consequence is that a request can pass validation and still come back unverified. If you are checking an NT, SA or ACT certificate issued on or after the date above and you do not send `certificate_number`, expect a `D` result (no match on the data supplied) rather than a validation error. ID Pass handles this by asking the holder for the certificate number and letting them say their certificate does not have one.

Data returned from a birth certificate verification will have the following structure:

```
{
    "first_name": "John",
    "last_name": "Smith",
    "date_of_birth": "1980-01-01",
    "registration_state": "NSW",
    "registration_number": "123456",
    ...
}
```

OCR and Edits

The ID Pass service uses an Optical Character Recognition (OCR) engine to extract the data from most document images before verification. The OCR engine is very accurate, however there can sometimes be errors in how the information is read, especially if the document is worn, scratched or in poor condition.

Centrelink cards and birth certificates are manual-entry only and do not use OCR.

The customer is shown the result of the OCR reading and is given the option to edit the data if it is incorrect. This edited data is then used to verify the identity of the customer.

If the customer edits the data, then the changes are recorded in the `changes` field of the `validation_summary` for the document.

```
{
    "validation_summary": {
```

```
        "document_1_result": {
            ...
            "changes": [
                "first_name": [
                    "from": "John",
                    "to": "Jane"
                ]
            ]
        }
    }
}
```

The `verification_description` will also contain a message to indicate that the document had biographic identity edits before verification.

```
{
    "verification_description": "
        Document 1 (passport) had biographic identity edits before verification\n
        The identity was verified successfully"
    ...
}
```

Liveness check

The liveness check works best on mobile devices with a front-facing camera. If the customer is using a desktop computer, then they may receive a lower liveness score.

Device	Typical Liveness Score
Mobile	90 to 99%
Desktop	70 to 90%

The ID Pass service sets a liveness pass cut-off score of 70%. If the liveness score is below 70%, then the customer will be shown an error screen and asked to repeat the liveness check.

Presentation of a false face video during the liveness check will typically result in a liveness score below 20%, far below the 70% pass cut-off score.

Sandbox testing

Sandbox behaviour depends on the selected verification strategy (as set in the `document_verification` parameter when creating the ID Pass). In all strategies, no real DVS calls are made and no fees are charged.

The same set of DVS sample documents can be uploaded through ID Pass in sandbox; what the API surfaces back to you differs per strategy:

Strategy	Sample documents	Other data
`dvs`	Full simulated DVS result and codes, per the sample document	Follows normal DVS sandbox behaviour
`idsp`	Identity opinion only (valid / not valid)	Identity opinion only (valid / not valid)
`basic`	`passed`	Format pass/fail

DVS in sandbox

When using the `dvs` verification strategy, Australian documents flow through to the simulated DVS engine, which returns the same result codes and descriptions as a direct DVS sandbox call. Uploading a sample document produces its documented sandbox response. A New Zealand Driver Licence flows through a simulated issuer response instead; see New Zealand Driver Licence in sandbox.

See the DVS sandbox responses and DVS sample documents for the full result matrix.

IDSP in sandbox

When using the `idsp` verification strategy, the API returns only Global Data's identity opinion - a valid / not valid assessment produced by the combined checks of Global Data's Document Validation System and DVS.

For testing, the DVS sample documents produce predictable identity opinions:

- Sample documents that pass all checks ? identity opinion is **valid**
- Sample documents that fail any check (document not found, data mismatch, lockouts, system errors, or any other failure) ? identity opinion is **not valid**

New Zealand Driver Licence in sandbox

This section applies to `document_verification: dvs` only. Under the `basic` strategy a New Zealand Driver Licence is format-checked locally and returns `basic_validation_passed`, so none of the simulated results below apply.

In sandbox with the `dvs` strategy, a New Zealand Driver Licence is verified against a simulated issuer response rather than the live source. The following details produce a verified (`Y`) result:

Field	Value
First Name	Dana
Last Name	Mitchell
Date of Birth	1965-09-13
Licence Number	DB512036
Version Number	001

Any other otherwise format-valid submission produces an `N` result with the `NZDL-101` code; licence number `DB111112` with any other details is a convenient deterministic no-match example. Licence number `DB000000` simulates a source outage (no result code; the customer may retry), and version number `999` simulates a rejected request. Details that fail the review-step rules (an invalid licence-number check digit, or a version number that is not exactly three digits) are rejected on the review page and never reach the simulated check.

Basic in sandbox

When using the `basic` verification strategy, no DVS call is made, live or simulated - only format validation is performed. Uploading a sample document returns `passed`. Any other input is validated against the normal document format rules and passes or fails on its own merits.

Cancel an ID Pass

It is possible to cancel an ID Pass after it has been created. This might be useful if, for example, the ID Pass link has been sent to the wrong person.

Call `idpass/cancel` with the ID Pass `id` to cancel when status is `new`, `opened`, or `in_progress`. The response includes `fee_refunded` (`true` or `false`) to indicate whether a refund was applied. A refund is only issued when no verification steps have been taken (typically status `new` or `opened`); if any steps have been taken, no refund is issued.

See the ID Pass API Reference for the full request and response specification.

ID Pass validation strategies

ID Pass is Global Data's Australian KYC identity verification service. You create a one-time verification link, your customer completes a guided online check, and you retrieve a clear pass/fail result plus the verified identity details. For a full overview of how ID Pass works - including liveness, biometric face match, configuration options, and the customer experience - see the ID Pass Overview.

When you create an ID Pass, you choose a document verification strategy via the `document_verification` parameter. The strategy determines how each identity document is checked and what is returned in the response. ID Pass supports three strategies:

- **ID Pass - DVS mode** (default) - Full verification via the Australian Document Verification Service (DVS), with the DVS result codes and details returned to you.
- **ID Pass - IDSP mode** - DVS verification performed by Global Data on your behalf. An opaque identity opinion (valid / not valid) is returned; DVS result codes and details are not exposed.
- **ID Pass - Basic mode** - Format and checksum validation only. No DVS call is made.

Each section below is a self-contained quick start for that strategy.

ID Pass - DVS mode

DVS mode (`document_verification: dvs`) is the default ID Pass verification strategy. It verifies each identity document directly against the Australian Document Verification Service (DVS) and returns the DVS result code and details to you.

This is the right mode for most KYC use cases where you want full visibility into the DVS outcome.

Requirements

- Your account must have a DVS Identity (OAC) registered for live environments. Registration is a free process available for Australian businesses - contact your Global Data account manager to register for the DVS.
- No DVS Identity is required in sandbox.

Quick start

1. Create an ID Pass

Call `POST /idpass/register` with `document_verification` set to `dvs` and the documents you want to collect. Optionally enable the liveness and biometric face match checks via `check_liveness` .

Example request:

```
{
  "document_verification": "dvs",
  "check_liveness": true,
  "document_1_allowed_types": ["licence", "passport"],
```

```
  "document_2_allowed_types": ["licence", "passport", "medicare"],
  "webhook_url": "https://example.com/webhooks/idpass"
}
```

The response contains the `id`, the `link` to send to your customer, and a `cipher_key` for decrypting the results later:

```
{
  "id": "19517924-94f0-4000-8e02-afba96ad5101",
  "link": "https://idpass.example.com/fe4291ca-d831-4760-96df-c9cb03b3cd95",
  "cipher_key": "IHm81bkcMsTL7J1ilxhDNE59+p5OQkvUJ3mZQhlUNHA="
}
```

See the ID Pass API Reference for all available request options (number of documents, return URL, image retention, requested identity, and so on).

2. Send the link to your customer

Provide the `link` to your customer by SMS, email, or by redirecting them from your application. See What your customer experiences for what they will see.

3. Track progress (optional)

If you supplied a `webhook_url` when creating the ID Pass, Global Data will POST a small payload to your endpoint each time the status changes:

```
{
  "id": "19517924-94f0-4000-8e02-afba96ad5101",
  "status": "complete"
}
```

See the Webhooks section of the overview for supported events.

4. Retrieve results

Once the ID Pass status is `complete`, call `GET /idpass/details/{id}` with the `cipher_key` to retrieve the full result.

The top-level `verification_status` is `passed` when all documents match and cross-document identity data is consistent, `failed` otherwise. The default response also includes a decrypted `validation_summary` with strategy-agnostic per-document results:

```
{
  "message": "Ok",
  "id": "19517924-94f0-4000-8e02-afba96ad5101",
  "status": "complete",
  "verification_status": "passed",
  "verification_description": "The identity was verified successfully",
  "config": { "...": "..." },
  "created_at": "2026-04-16T04:28:53+00:00",
  "consent_given_at": "2026-04-16T04:29:00+00:00",
  "consent_text": "...",
  "completed_at": "2026-04-16T04:29:30+00:00",
  "ip_address": "203.22.251.3",
  "validation_summary": {
    "document_verification": "dvs",
    "liveness_result": {
```

```
      "validated": true,
      "confidence": "99.811",
      "threshold": 70,
      "liveness_attempts": 1,
      "bounding_box": { "...": "..." }
    },
    "document_1_result": {
      "document_type": "licence",
      "validated_fields": {
        "first_name": "John",
        "last_name": "Smith",
        "date_of_birth": "1990-03-21"
      },
      "biographic_validated": true,
      "changes": {},
      "biometric_validated": true,
      "biometric_similarity": "98.876",
      "biometric_threshold": 80
    },
    "verified_identity": {
      "first_name": "John",
      "middle_name": null,
      "last_name": "Smith",
      "date_of_birth": "1990-03-21"
    },
    "requested_identity_mismatch": false
  }
}
```

Key signals in DVS mode:

- `verification_status` - `passed` or `failed` for the overall ID Pass.
- `verification_description` - human-readable reason.
- `validation_summary.document_N_result.biographic_validated` - whether the document matched against DVS.
- `validation_summary.document_N_result.biometric_validated` - whether the biometric face match passed for that document (when a photo ID was supplied).

Opting in to the raw DVS detail

To see the DVS result code and details, pass `return_documents: true` on the details request. The response then also contains a `documents` object with a per-document `validation_result`:

```
{
  "...": "...top-level fields as above...",
  "documents": {
    "document_1": {
      "document_type": "licence",
      "document_number": 1,
      "ocr_status": "complete",
      "ocr_attempts": 1,
      "validation_status": "complete",
      "validation_result": {
        "strategy": "dvs",
        "verification_result_code": "Y",
        "verification_request_number": "7a3ed6e1-b707-4e2d-bacb-e2dfe8f57406",
        "additional_information": null,
        "originating_agency_code": "UOO1",
        "checked_at": "2026-04-16 04:28:53"
      },
```

```
      "validation_attempts": 1,
      "biometric_result": {
        "passed": true,
        "threshold": 80,
        "similarity": 98.8764,
        "face_occluded": false
      },
      "ocr_data": { "...": "..." },
      "validation_data": { "...": "..." }
    }
  }
}
```

Key `validation_result` fields for DVS mode:

- `strategy` - `dvs`
- `verification_result_code` - DVS result: `Y` (match), `N` (no match), or `D` (document invalid)
- `verification_request_number` - DVS request identifier, required for any official queries on the result
- `additional_information` - detail about any mismatch (e.g. "Card number did not match")
- `originating_agency_code` - the OAC used for the request
- `checked_at` - timestamp of the DVS call

Sandbox testing

In sandbox, no real DVS calls are made and no fees are charged. Uploading a DVS sample document produces that sample's documented sandbox response - when you opt in with `return_documents: true`, the `verification_result_code`, `additional_information`, and other DVS fields are returned exactly as they would be for a direct DVS sandbox call. See Sandbox testing in the ID Pass Overview for the full picture.

ID Pass - IDSP mode

In IDSP mode (`document_verification: idsp`), Global Data returns an **identity opinion** - a single valid / not valid assessment produced by the combined checks of Global Data's Document Validation System and DVS. Global Data runs the document verification on your behalf and provides you with an identity opinion on whether the documents are valid or not.

This mode is intended for customers who are not registered with DVS but need a compliant identity verification service. Global Data acts as an Identity Service Provider (IDSP) on your behalf, and the identity opinion returned is Global Data's own assessment, not a DVS response.

Requirements

- Liveness detection must be enabled on every ID Pass (`check_liveness` set to `true`). This is mandatory in IDSP mode - it ensures the identity opinion is based on multiple independent identity checks (liveness, biometric face match where a photo ID is supplied, and document verification).
- Your account must be enrolled as an IDSP Service Client. Contact your Global Data account manager to arrange enrolment.
- No DVS Identity (OAC) is required on your account.

Quick start

1. Create an ID Pass

Call `POST /idpass/register` with `document_verification` set to `idsp` and `check_liveness` set to `true`.

Example request:

```
{
  "document_verification": "idsp",
  "check_liveness": true,
  "document_1_allowed_types": ["licence", "passport"],
  "document_2_allowed_types": ["licence", "passport", "medicare"],
  "webhook_url": "https://example.com/webhooks/idpass"
}
```

The response contains the `id`, the `link` to send to your customer, and a `cipher_key` for decrypting the results later:

```
{
  "id": "19517924-94f0-4000-8e02-afba96ad5101",
  "link": "https://idpass.example.com/fe4291ca-d831-4760-96df-c9cb03b3cd95",
  "cipher_key": "IHm81bkcMsTL7J1ilxhDNE59+p5OQkvUJ3mZQhlUNHA="
}
```

See the ID Pass API Reference for all available request options.

2. Send the link to your customer

Provide the `link` to your customer by SMS, email, or by redirecting them from your application. See What your customer experiences for what they will see.

3. Track progress (optional)

If you supplied a `webhook_url` when creating the ID Pass, Global Data will POST a small payload to your endpoint each time the status changes:

```
{
  "id": "19517924-94f0-4000-8e02-afba96ad5101",
  "status": "complete"
}
```

See the Webhooks section of the overview for supported events.

4. Retrieve results

Once the ID Pass status is `complete`, call `GET /idpass/details/{id}` with the `cipher_key` to retrieve the full result.

The top-level `verification_status` carries Global Data's identity opinion: `passed` (valid) or `failed` (not valid). The default response also includes a decrypted `validation_summary` with strategy-agnostic per-document results:

```
{
  "message": "Ok",
  "id": "19517924-94f0-4000-8e02-afba96ad5101",
  "status": "complete",
```

```
  "verification_status": "passed",
  "verification_description": "The identity was verified successfully",
  "config": { "...": "..." },
  "created_at": "2026-04-16T04:28:53+00:00",
  "consent_given_at": "2026-04-16T04:29:00+00:00",
  "consent_text": "...",
  "completed_at": "2026-04-16T04:29:30+00:00",
  "ip_address": "203.22.251.3",
  "validation_summary": {
    "document_verification": "idsp",
    "liveness_result": {
      "validated": true,
      "confidence": "99.811",
      "threshold": 70,
      "liveness_attempts": 1,
      "bounding_box": { "...": "..." }
    },
    "document_1_result": {
      "document_type": "passport",
      "validated_fields": {
        "first_name": "John",
        "middle_name": null,
        "last_name": "Smith",
        "date_of_birth": "1990-03-21"
      },
      "biographic_validated": true,
      "changes": {},
      "biometric_validated": true,
      "biometric_similarity": "99.804",
      "biometric_threshold": 80
    },
    "verified_identity": {
      "first_name": "John",
      "middle_name": null,
      "last_name": "Smith",
      "date_of_birth": "1990-03-21"
    },
    "requested_identity_mismatch": false
  }
}
```


Key signals in IDSP mode:

- `verification_status` - this is Global Data's identity opinion: `passed` (valid) or `failed` (not valid).
- `verification_description` - human-readable reason.
- `validation_summary.document_N_result.biographic_validated` - whether the document passed Global Data's combined Document Validation System and DVS checks.
- `validation_summary.document_N_result.biometric_validated` - whether the biometric face match passed for that document.

DVS result codes, DVS error codes, DVS request identifiers, and the OAC used are not available in the API response.

Opting in to the per-document detail

To see the per-document verification boolean, pass `return_documents: true` on the details request. The response then also contains a `documents` object with a per-document `validation_result`:


```
{
  "...": "...top-level fields as above...",
```

```
  "documents": {
    "document_1": {
      "document_type": "passport",
      "document_number": 1,
      "ocr_status": "complete",
      "ocr_attempts": 1,
      "validation_status": "complete",
      "validation_result": {
        "strategy": "idsp",
        "verification_passed": true,
        "checked_at": "2026-04-16 04:28:53"
      },
      "validation_attempts": 1,
      "biometric_result": {
        "passed": true,
        "threshold": 80,
        "similarity": 99.8038,
        "face_occluded": false
      },
      "ocr_data": { "...": "..." },
      "validation_data": { "...": "..." }
    }
  }
}
```

Key `validation_result` fields for IDSP mode:

- `strategy` - `idsp`
- `verification_passed` - boolean (valid / not valid) for that document
- `checked_at` - timestamp of the validation

Sandbox testing

In sandbox, no real DVS calls are made and no fees are charged. Uploading a DVS sample document produces a predictable identity opinion:

- Sample documents that pass all checks ? `verification_status: "passed"` (identity opinion is valid)
- Sample documents that fail any check ? `verification_status: "failed"` (identity opinion is not valid)

See Sandbox testing in the ID Pass Overview for the full picture.

ID Pass - Basic mode

Basic mode (`document_verification: basic`) performs format and checksum validation on the supplied document fields but does not perform any external verification. No DVS call is made, in either live or sandbox.

This mode is useful when you need OCR extraction, liveness, and biometric face match but intend to perform document verification yourself, or when DVS verification is not required for your use case.

Requirements

- No DVS Identity or special enrolment is required.

Quick start

1. Create an ID Pass

Call `POST /idpass/register` with `document_verification` set to `basic` and the documents you want to collect. Optionally enable the liveness and biometric face match checks via `check_liveness`.

Example request:

```
{
  "document_verification": "basic",
  "check_liveness": true,
  "document_1_allowed_types": ["licence", "passport"],
  "document_2_allowed_types": ["licence", "passport", "medicare"],
  "webhook_url": "https://example.com/webhooks/idpass"
}
```

The response contains the `id`, the `link` to send to your customer, and a `cipher_key` for decrypting the results later:

```
{
  "id": "19517924-94f0-4000-8e02-afba96ad5101",
  "link": "https://idpass.example.com/fe4291ca-d831-4760-96df-c9cb03b3cd95",
  "cipher_key": "IHm81bkcMsTL7J1ilxhDNE59+p5OQkvUJ3mZQhlUNHA="
}
```

See the ID Pass API Reference for all available request options.

2. Send the link to your customer

Provide the `link` to your customer by SMS, email, or by redirecting them from your application. See What your customer experiences for what they will see.

3. Track progress (optional)

If you supplied a `webhook_url` when creating the ID Pass, Global Data will POST a small payload to your endpoint each time the status changes:

```
{
  "id": "19517924-94f0-4000-8e02-afba96ad5101",
  "status": "complete"
}
```

See the Webhooks section of the overview for supported events.

4. Retrieve results

Once the ID Pass status is `complete`, call `GET /idpass/details/{id}` with the `cipher_key` to retrieve the full result.

The top-level `verification_status` is `passed` when all documents pass format validation and cross-document identity data is consistent, `failed` otherwise. The default response also includes a decrypted `validation_summary` with strategy-agnostic per-document results:

```
{
  "message": "Ok",
  "id": "19517924-94f0-4000-8e02-afba96ad5101",
  "status": "complete",
  "verification_status": "passed",
  "verification_description": "The identity was verified successfully",
  "config": { "...": "..." },
  "created_at": "2026-04-16T04:28:53+00:00",
  "consent_given_at": "2026-04-16T04:29:00+00:00",
  "consent_text": "...",
  "completed_at": "2026-04-16T04:29:30+00:00",
  "ip_address": "203.22.251.3",
  "validation_summary": {
    "document_verification": "basic",
    "liveness_result": {
      "validated": true,
      "confidence": "99.811",
      "threshold": 70,
      "liveness_attempts": 1,
      "bounding_box": { "...": "..." }
    },
    "document_1_result": {
      "document_type": "licence",
      "validated_fields": {
        "first_name": "John",
        "last_name": "Smith",
        "date_of_birth": "1990-03-21"
      },
      "biographic_validated": true,
      "changes": {},
      "biometric_validated": true,
      "biometric_similarity": "98.876",
      "biometric_threshold": 80
    },
    "verified_identity": {
      "first_name": "John",
      "middle_name": null,
      "last_name": "Smith",
      "date_of_birth": "1990-03-21"
    },
    "requested_identity_mismatch": false
  }
}
```


Key signals in Basic mode:

- `verification_status` - `passed` or `failed` for the overall ID Pass.
- `verification_description` - human-readable reason.
- `validation_summary.document_N_result.biographic_validated` - whether the document's fields passed format and checksum validation.
- `validation_summary.document_N_result.biometric_validated` - whether the biometric face match passed for that document.

Opting in to the per-field validation errors

To see the per-field validation errors for a failed document, pass `return_documents: true` on the details request. The response then also contains a `documents` object with a per-document `validation_result`. When validation passes:

```
{
  "...": "...top-level fields as above...",
  "documents": {
    "document_1": {
      "document_type": "licence",
      "document_number": 1,
      "ocr_status": "complete",
      "ocr_attempts": 1,
      "validation_status": "complete",
      "validation_result": {
        "strategy": "basic",
        "basic_validation_passed": true,
        "errors": {},
        "checked_at": "2026-04-16 04:28:53"
      },
      "validation_attempts": 1,
      "biometric_result": {
        "passed": true,
        "threshold": 80,
        "similarity": 98.8764,
        "face_occluded": false
      },
      "ocr_data": { "...": "..." },
      "validation_data": { "...": "..." }
    }
  }
}
```

When validation fails, `errors` contains per-field messages:

```
{
  "verification_status": "failed",
  "verification_description": "Document 1 failed format validation",
  "documents": {
    "document_1": {
      "document_type": "licence",
      "document_number": 1,
      "validation_status": "complete",
      "validation_result": {
        "strategy": "basic",
        "basic_validation_passed": false,
        "errors": {
          "card_number": ["The VIC licence card number must be 8 or 10 digits."]
        },
        "checked_at": "2026-04-16 04:28:53"
      }
    }
  }
}
```

Key `validation_result` fields for Basic mode:

- `strategy` - `basic`
- `basic_validation_passed` - boolean
- `errors` - per-field validation errors; empty object when validation passed
- `checked_at` - timestamp of the validation

Sandbox testing

No DVS call is made in sandbox (or live) in Basic mode. Uploading a DVS sample document produces a passing format validation (`basic_validation_passed: true` when you opt in via `return_documents: true`). Any other input is validated against the normal document format rules and passes or fails on its own merits. See Sandbox testing in the ID Pass Overview for the full picture.

Learn more

- ID Pass Overview - full walkthrough including liveness, biometric face match, configuration, billing, and webhook details.
- ID Pass User Journey - what your customer will experience end-to-end.
- DVS Overview - about the DVS service and result codes.
- ID Pass API Reference - full field-by-field definitions for every request and response.

User Journey

This is a walkthrough of the ID Pass workflow.

Scenario: An ID Pass is configured to require liveness check and three different ID documents. For each ID document, we will accept one of Licence, Passport or Medicare Card:

- Liveness check: Required
- Document 1: Allowed types: Licence, Passport, Medicare
- Document 2: Allowed types: Licence, Passport, Medicare
- Document 3: Allowed types: Licence, Passport, Medicare
- ID Photo: Required

An ID Pass link looks like this:

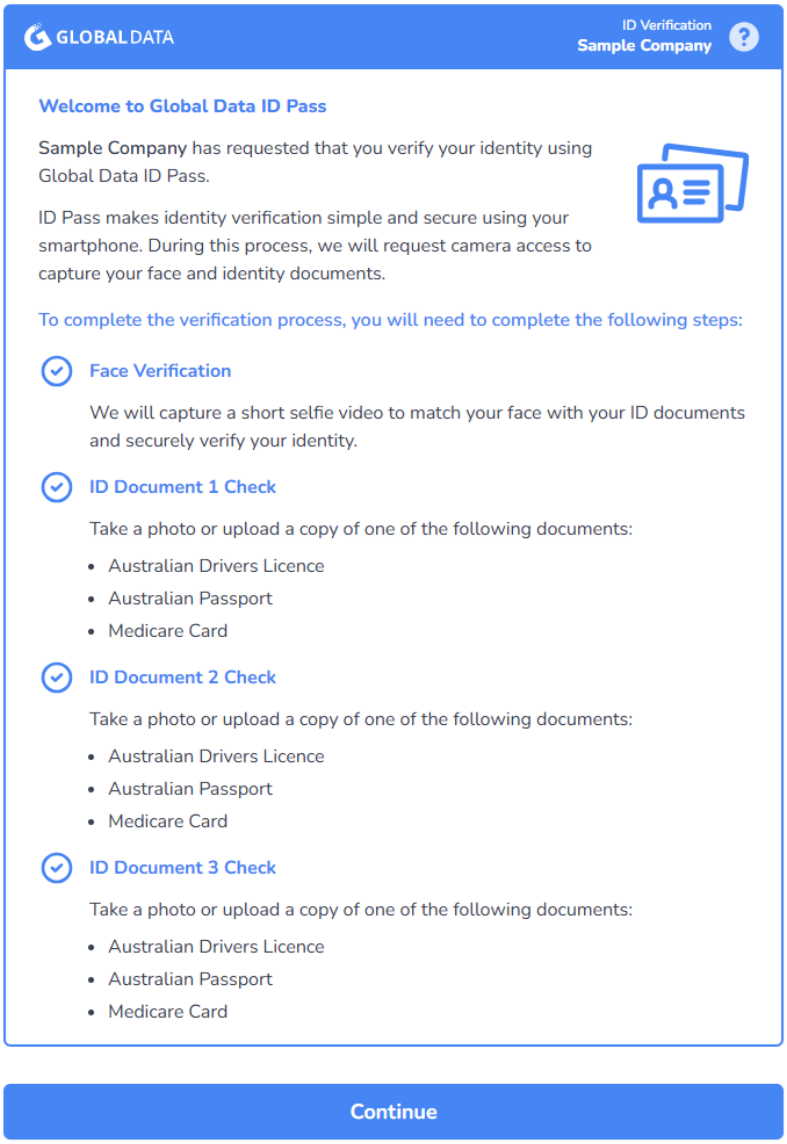
https://idpass.globaldata.net.au/idpass?token=CLubZpt0QcBWQ

This link can then be sent to the user using an email to SMS with instructions to visit the ID Pass link and follow the on-screen instructions to complete the identity verification.

> Each section shows the user interface for that step of the ID Pass process, The user will click the **Continue** button to proceed to the next step.

ID Pass Start

When the user first opens the ID Pass link, they start at the introduction page which outlines the process. This page is dynamically generated to reflect the configured options on the ID Pass. In our example case this tells the user that we will complete a Face Verification and three separate ID document checks:

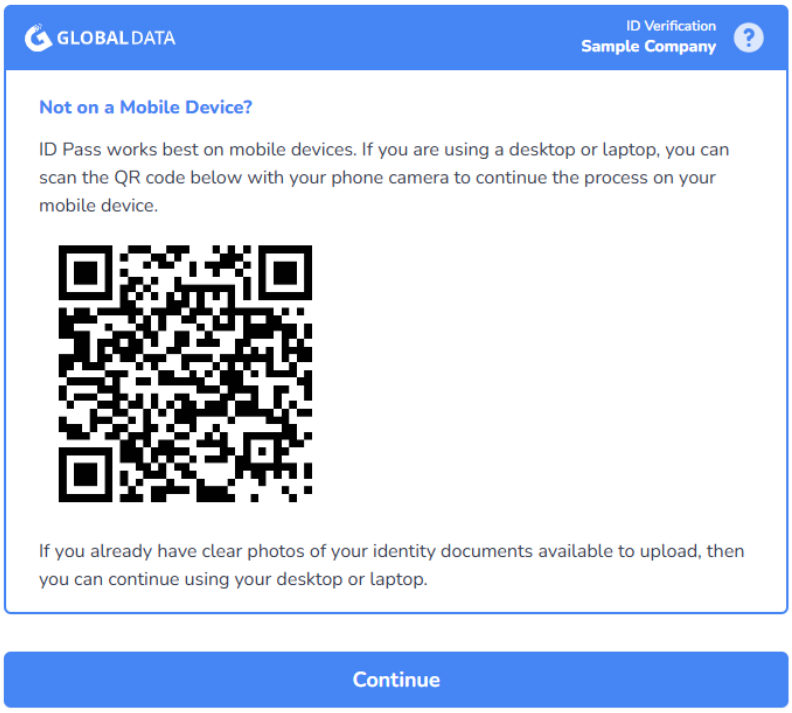


The header bar on each page displays the requesting company name (in this case “Sample Company”) and a context-specific help link. Users are guided through each verification step dynamically based on their selected ID verification options.

Mobile Device Optimisation

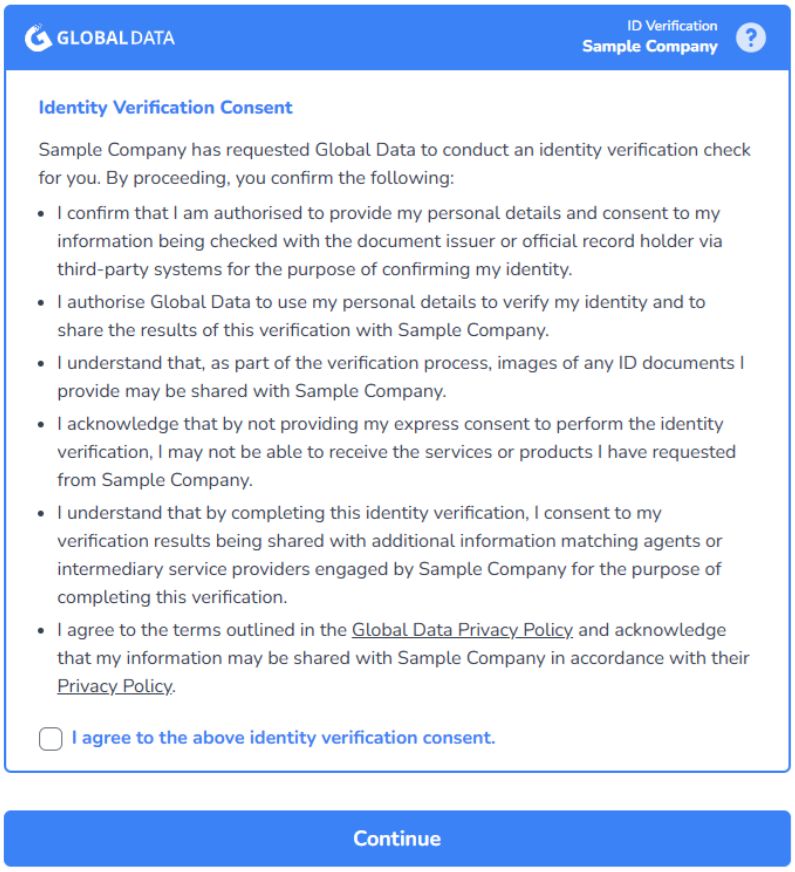
ID Pass has been optimised to work best on mobile devices. While it will function on a desktop with a web camera, best results (especially for the liveness check) are obtained when the user is using a mobile device.

If the user is not using a mobile device, then they will be presented with a screen which suggests that they scan the QR code to continue on a mobile device.



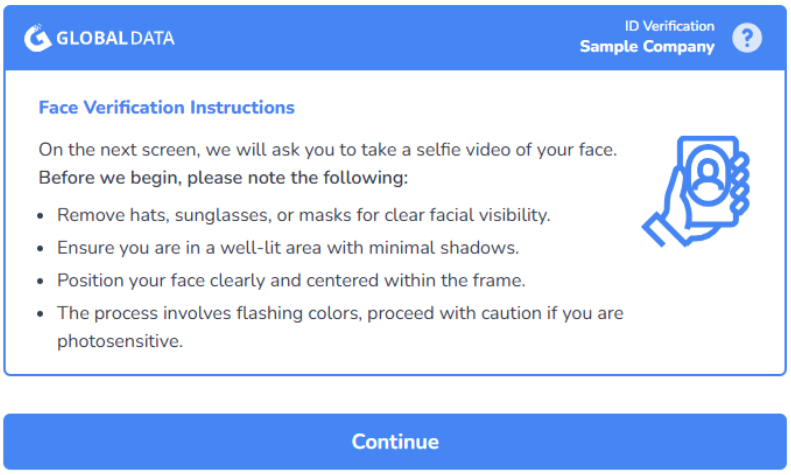
Consent Request

The user is presented with a consent screen. The content of this screen is dynamically generated depending on which options are selected for the ID Pass (for example, the face verification section is only included if a liveness check is requested during configuration). The Privacy policy for your company is included as a clickable link. Please contact support to update your privacy policy URL.

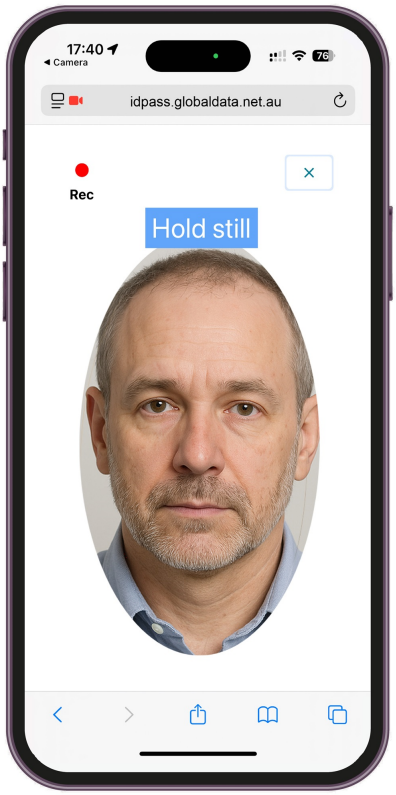


Face Verification

The face verification step starts with the Instructions page, followed by the actual verification and then either a success or failure page. The user is requested to take a selfie video for liveness detection and biometric face matching.



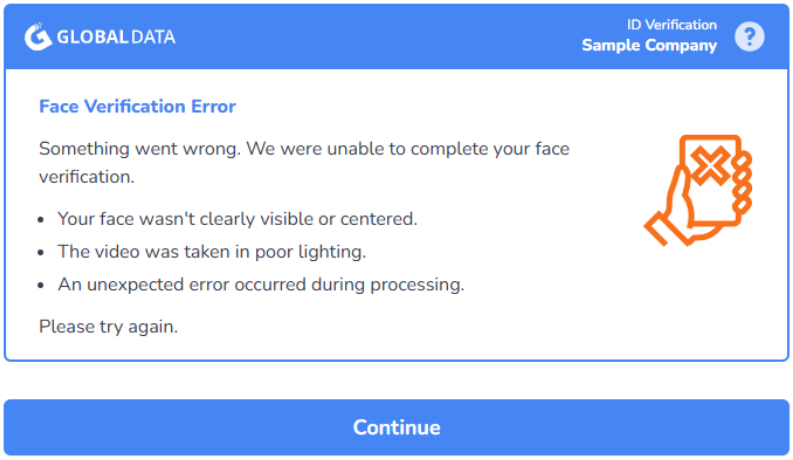
The user is then requested to take a selfie video for liveness detection and biometric face matching.



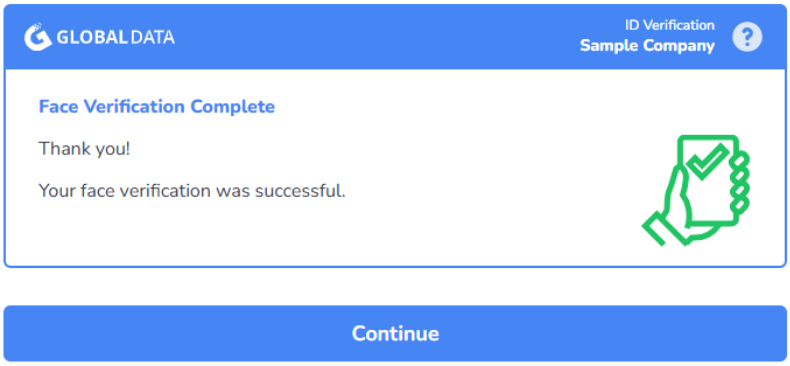
It takes a few seconds to analyse the selfie video. Once completed the user is presented with either a completed screen or an error screen if the liveness check was not successful.

Face Verification Error

If the verification fails, then the user is prompted to try again.



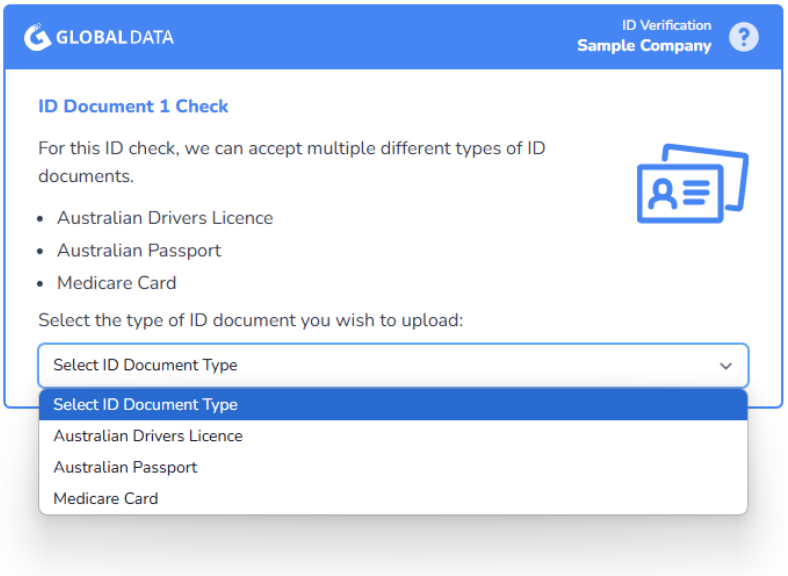
Face Verification Completed



First ID document (Licence)

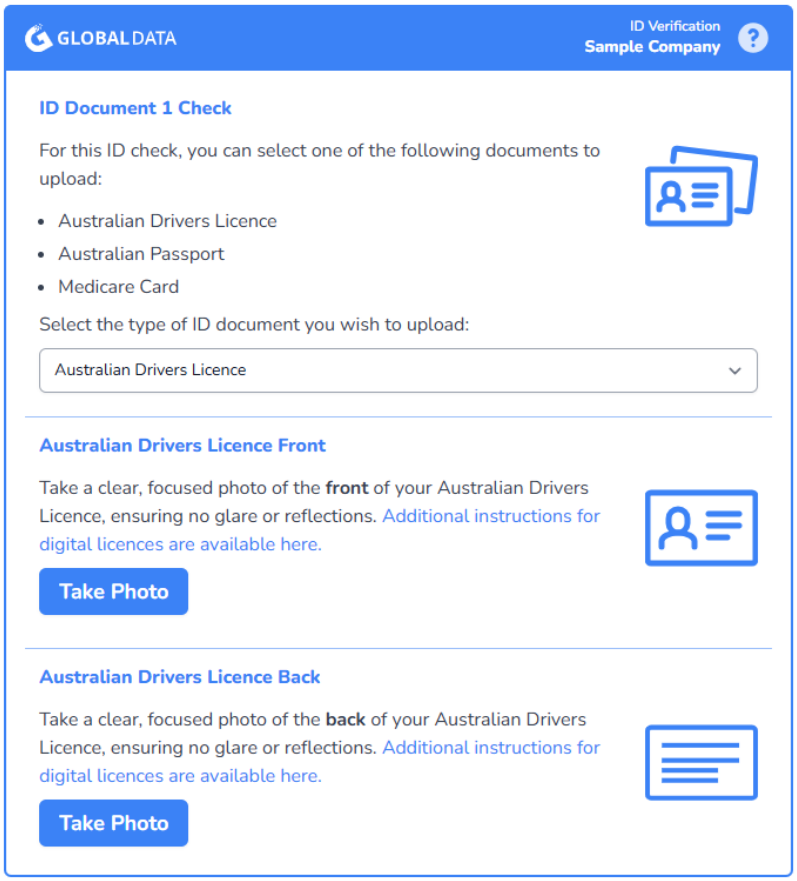
When we configure the ID Pass, we can specify the document types that are allowed for each ID check. In this example, we have configured the ID Pass to allow one of Licence, Passport or Medicare for the first ID check.

The user is requested to select which type of document they wish to use for the first ID check



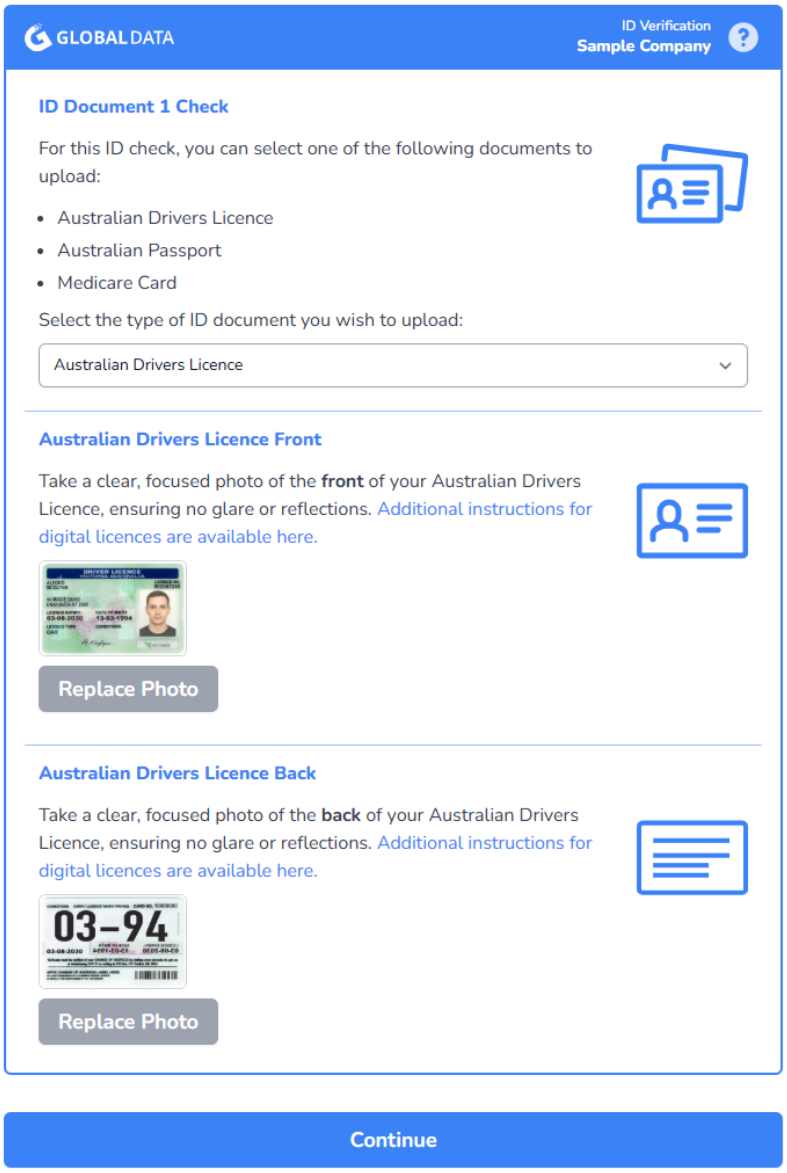
In this example the user has selected to use their licence and is requested to upload the front and back images of their licence.

They can choose to take a photo or upload an image from their drive or photo library.

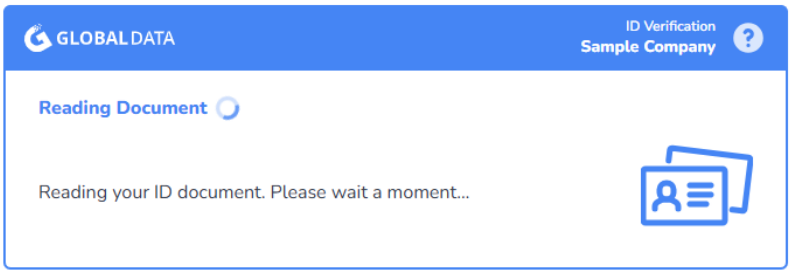


Licence Photo Upload

The user can upload photos or chose to replace an uploaded photo. The continue button appears once the required photos are uploaded.



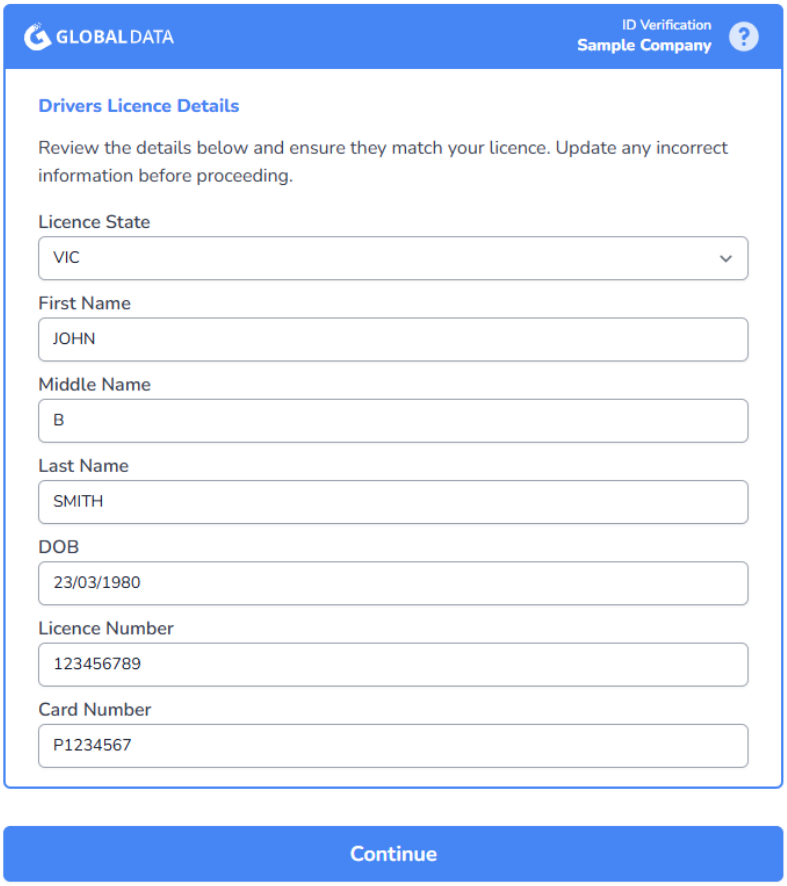
After continuing, the licence data is read from the photos using the licence OCR engine. This process takes a few seconds.



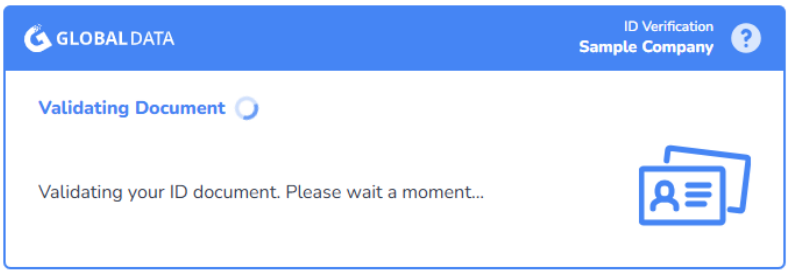
Licence Detail Review

The detected licence details are then displayed for review. While the OCR engine is very accurate, there can sometimes be errors in how the information is read, especially if the licence is worn, scratched or in poor condition.

The user is free to update the information before continuing. Note that any changes made by the user will be recorded in the change log and can be reviewed once the ID Pass process has completed.

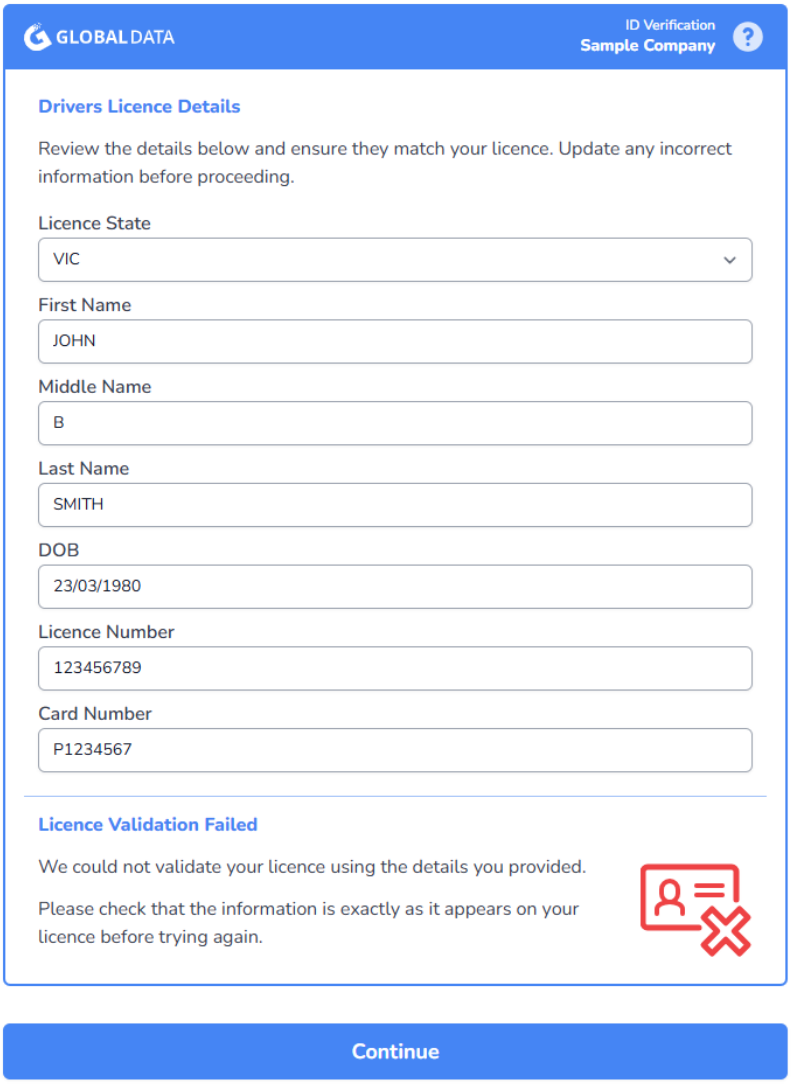


After continuing, the licence data is verified using the DVS service. This process takes a few seconds. Once completed, the use is presented with either the completed or error page.

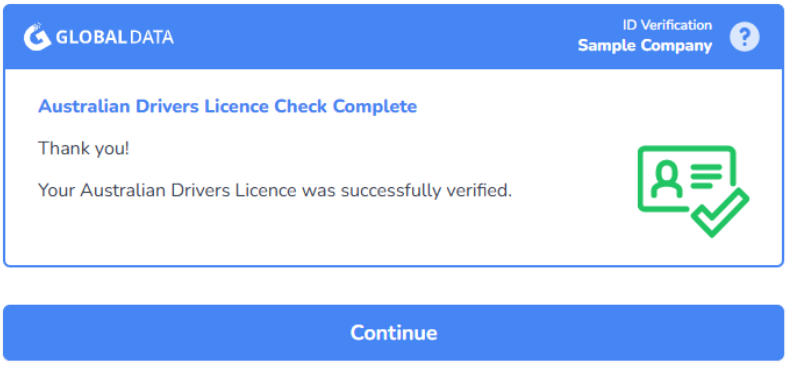


Licence Verification Error

If the verification fails, the user is presented with an error screen and asked to review the information before retrying the validation.



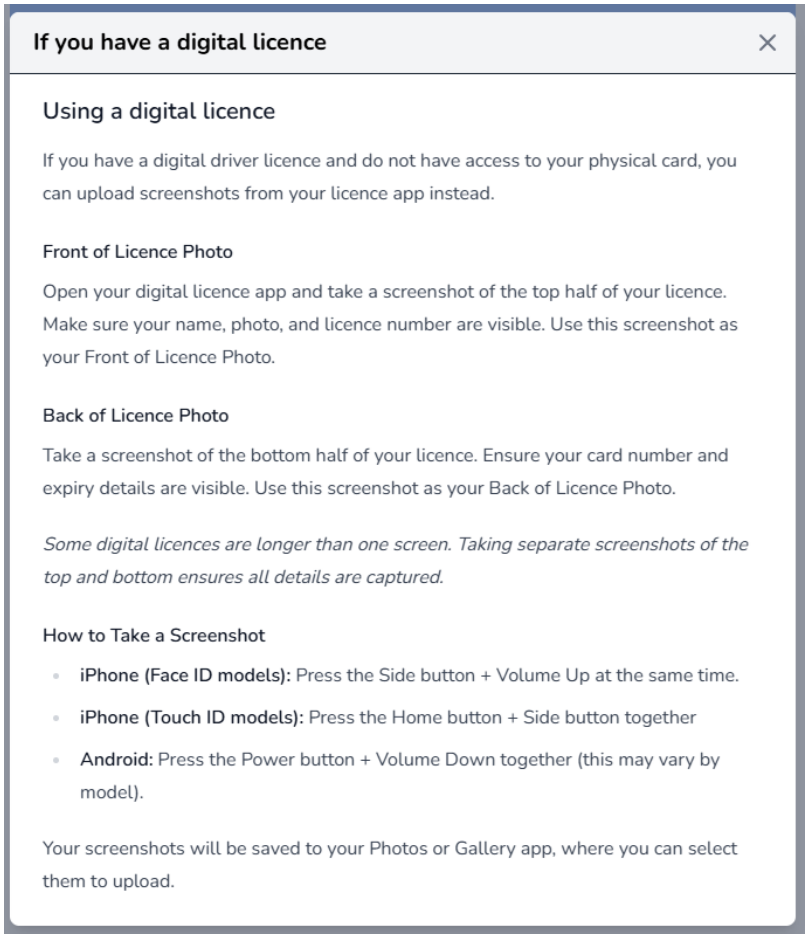
Licence Verification Completed



Digital Licences

The IDPass system supports the use of digital licences (ie licence stored in a mobile app). To use a digital licence the user will be required to take two screenshots, one of the top and one of the bottom of the digital licence screen. These screenshots are then uploaded as the front and back images.

Additional help is linked from the licence upload page:

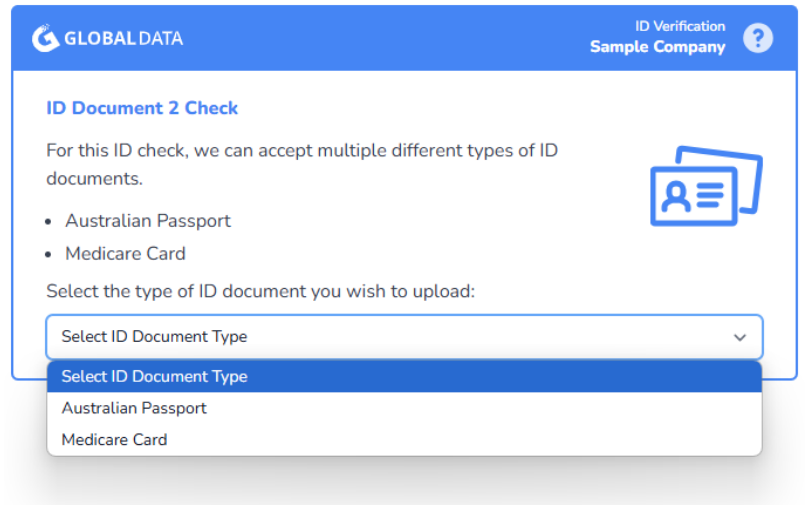


Second ID document (Passport)

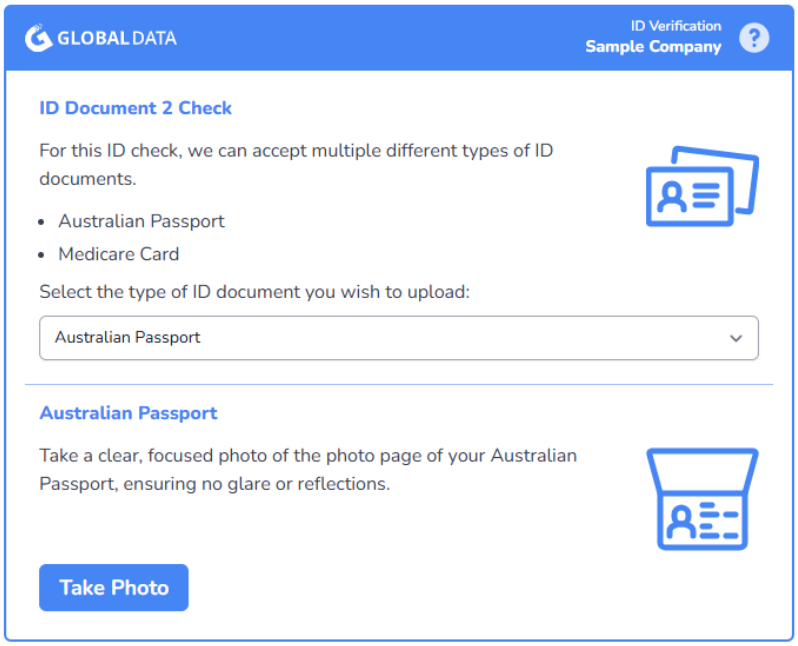
When we configure the ID Pass, we can specify the document types that are allowed for each ID check. In this example, we have configured the ID Pass to allow one of Passport or Medicare for the second ID check.

Note that although we configured the ID Pass to allow Licence, Passport and Medicare for the second ID document, the select box only shows Passport and Medicare. This is because the user has already used their licence as the first ID document, so it is not allowed to be used again here.

The user is requested to select which type of document they wish to use for the second ID check.

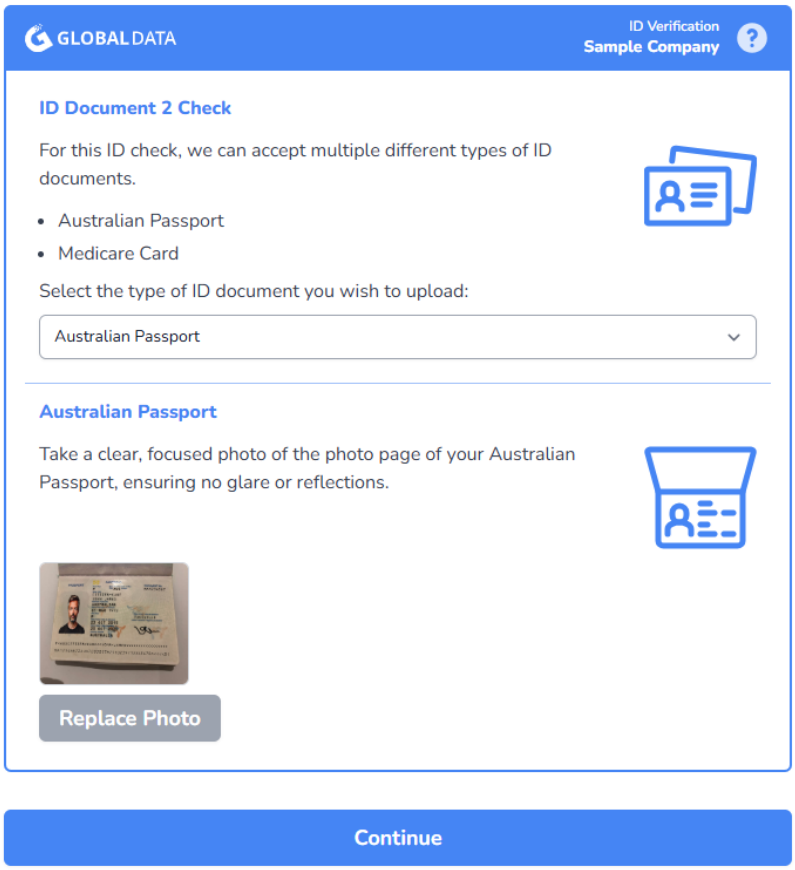


In this example the user has selected to use their passport and is requested to upload a photo of the information page of their passport. They can choose to take a photo or upload an image from their drive or photo library.

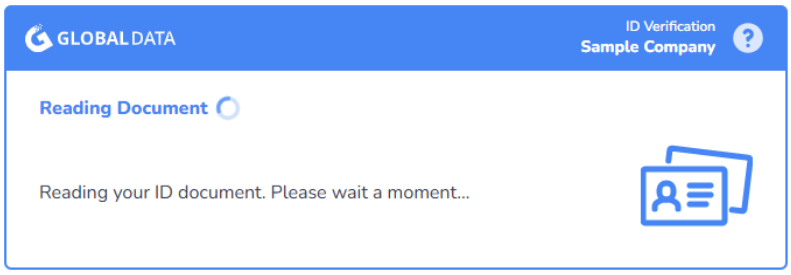


Passport Photo Upload

The user can then upload the photo or chose to replace an uploaded photo. The continue button appears once the photo is uploaded.

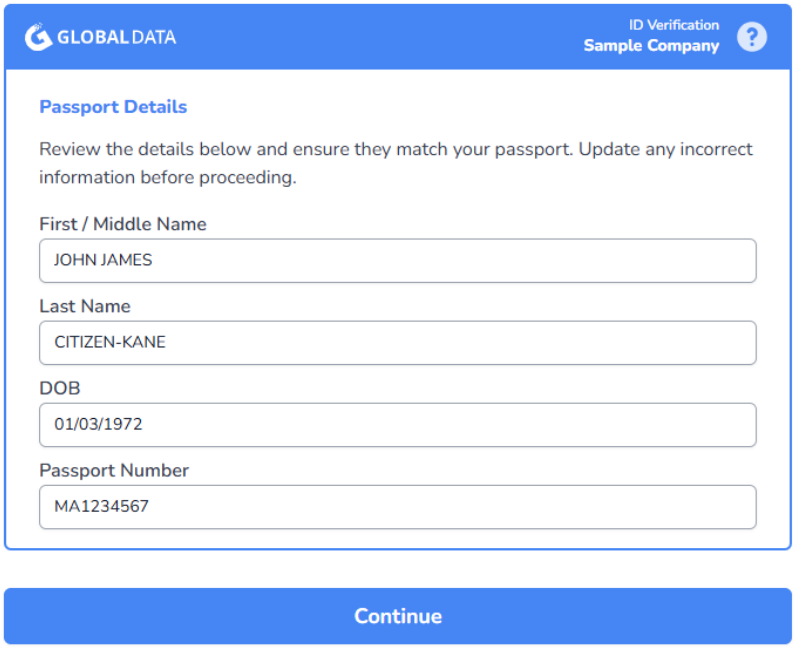


After continuing, the passport data is read from the photo using the passport OCR engine. This process takes a few seconds



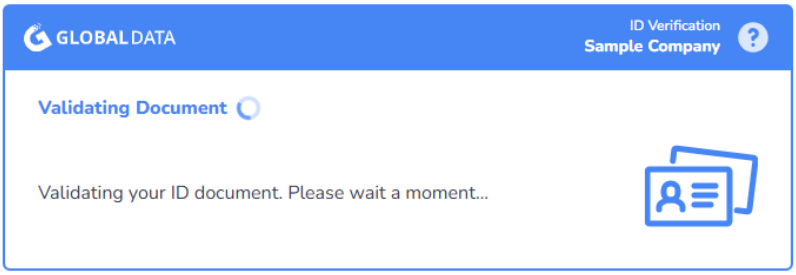
Passport Detail Review

The detected passport details are then displayed for review. While the OCR engine is very accurate, there can sometimes be errors in how the information is read, especially if the passport is worn, scratched or in poor condition. The user is free to update the information before continuing. Note that any changes made by the user will be recorded in the change log and can be reviewed once the ID Pass process has completed.



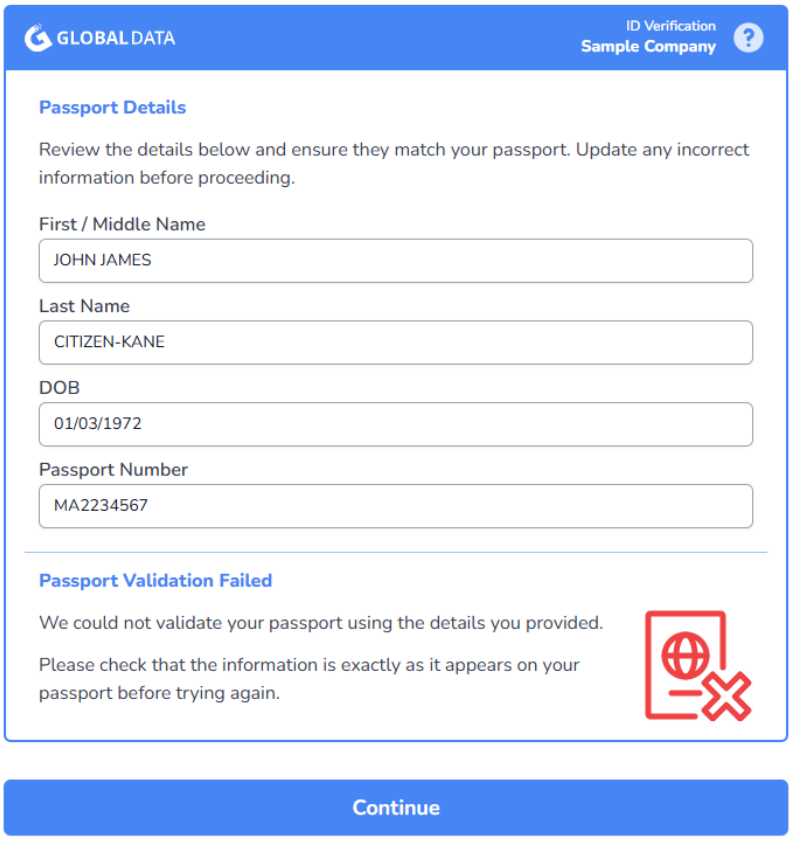
After continuing, the passport data is verified using the DVS service. This process takes a few seconds.

Once completed, the use is presented with either the completed or error page.

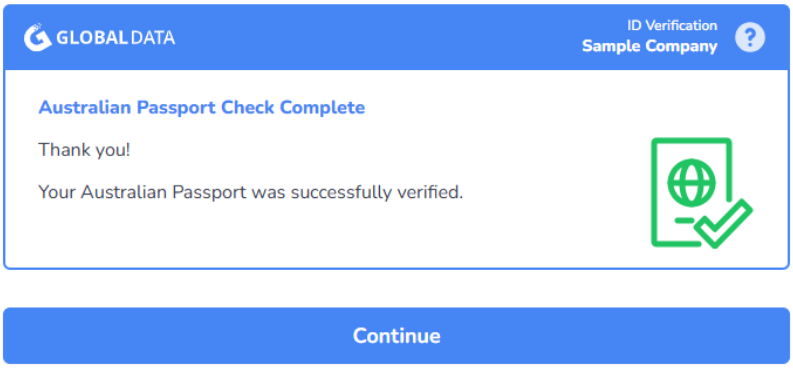


Passport Verification Error

If the verification fails, the user is presented with an error and asked to review the information before retrying the validation.



Passport Verification Completed



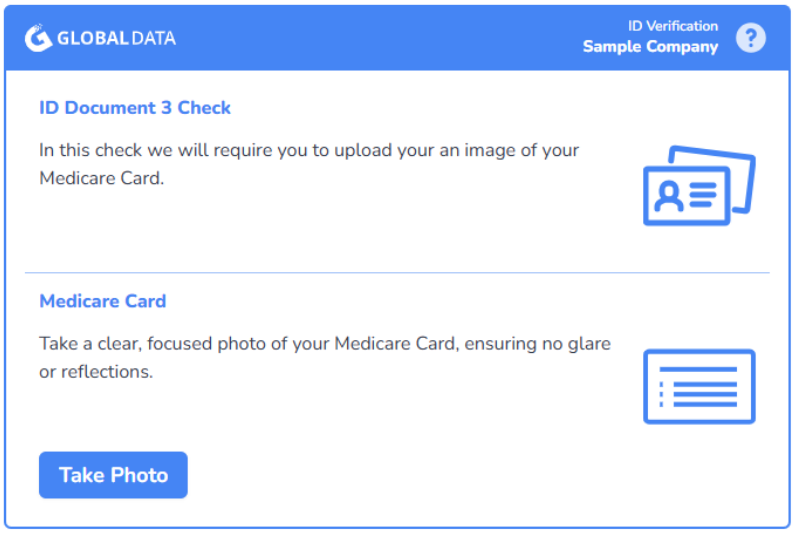
Third ID document (Medicare)

When we configure the ID Pass, we can specify the document types that are allowed for each ID check. In this example, we have configured the ID Pass to allow one of Medicare for the third ID check.

Note that although we configured the ID Pass to allow Licence, Passport and Medicare for the third ID document, there is no select box shown and the user is just asked to validate their Medicare card.

This is because the user has already used their licence and passport as the first and second ID document, leaving only the Medicare card option for the third ID document.

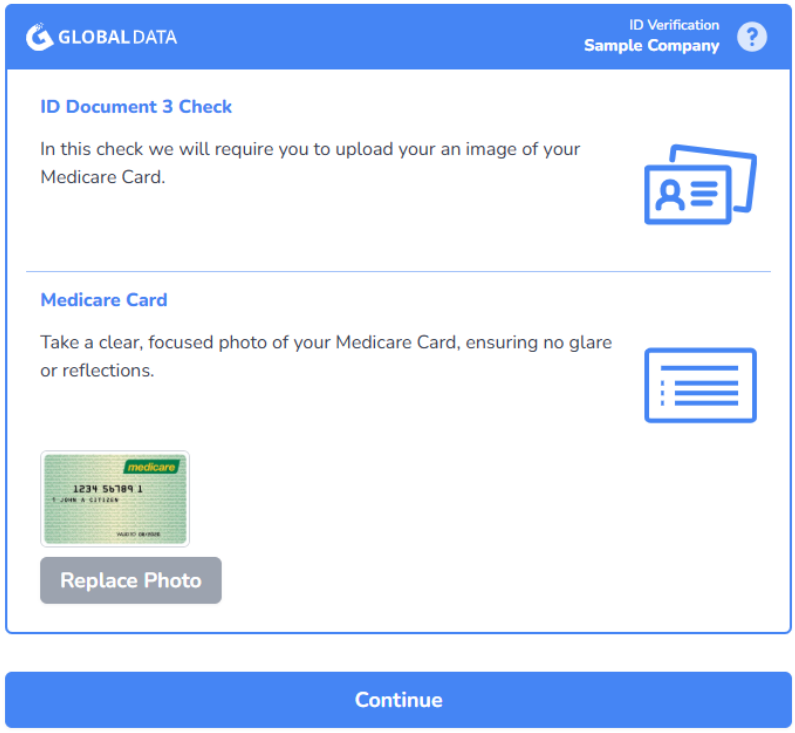
The user is requested for the third ID document. This is the Medicare card.



Medicare Photo Upload

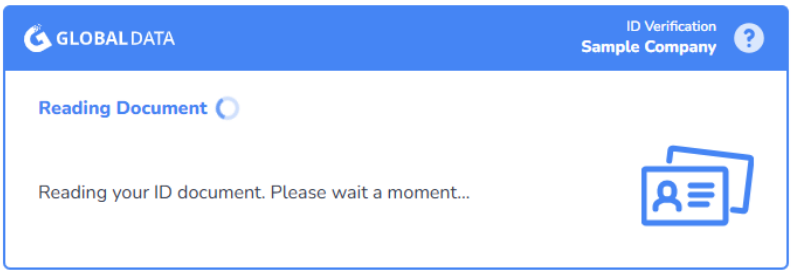
The user can then upload the photo or chose to replace an uploaded photo.

The continue button appears once the photo is uploaded.



After continuing, the Medicare data is read from the photo using the Medicare OCR engine.

This process takes a few seconds

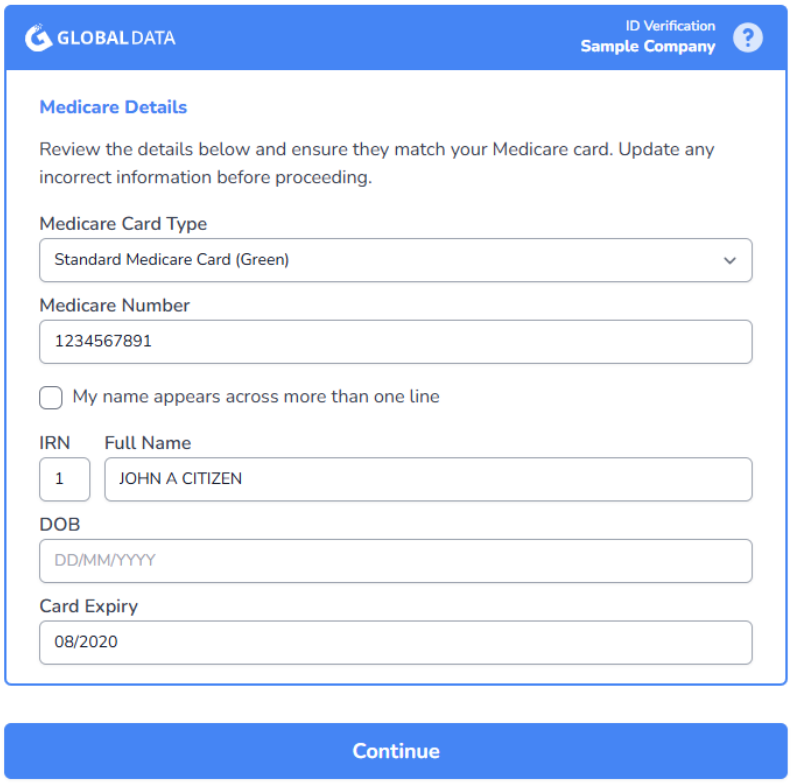


Medicare Detail Review

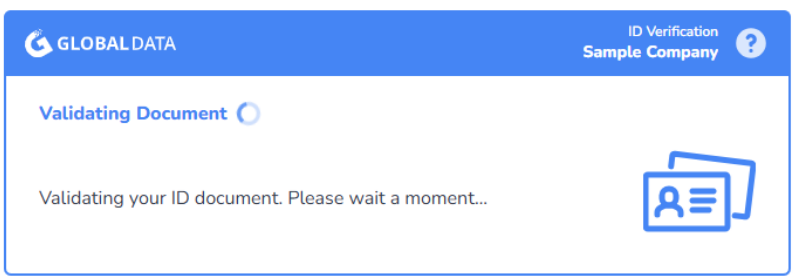
The detected Medicare details are then displayed for review. While the OCR engine is very accurate, there can sometimes be errors in how the information is read, especially if the Medicare card is worn, scratched or in poor condition.

The user is free to update the information before continuing. Note that any changes made by the user will be recorded in the change log and can be reviewed once the ID Pass process has completed.

Note that the user is required to enter their DOB for this verification. Manual entry is required as the users DOB does not appear on the Medicare card and therefore is not pre-filled by the OCR engine.

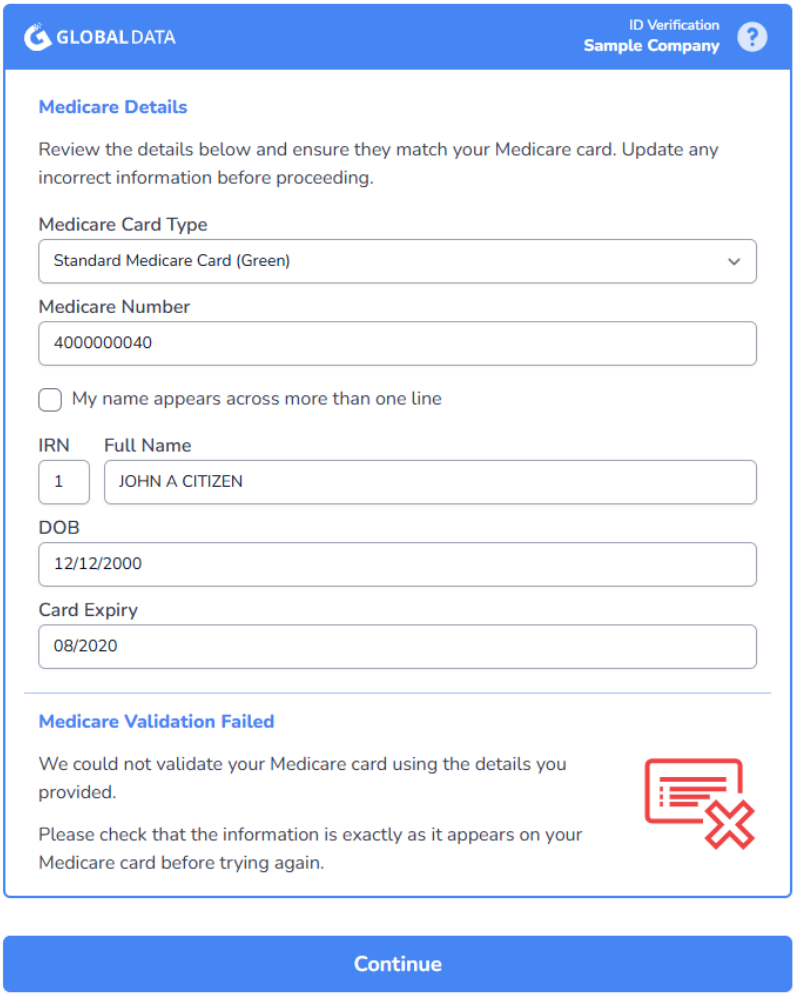


After continuing, the Medicare data is verified using the DVS service. This process takes a few seconds. Once completed, the use is presented with either the completed or error page.

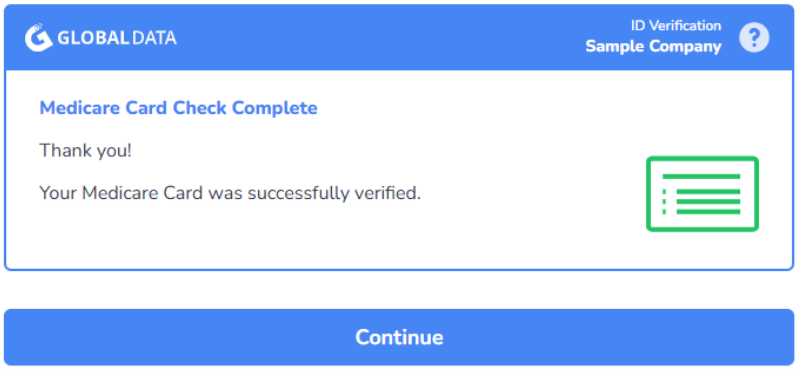


Medicare Card Verification Error

If the verification fails, the user is presented with an error and asked to review the information before retrying the validation.



Medicare Card Verification Completed

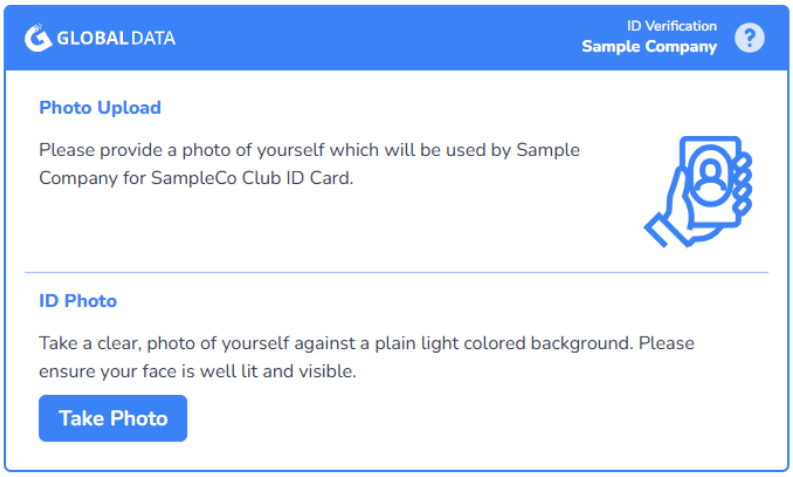


ID Photo Upload

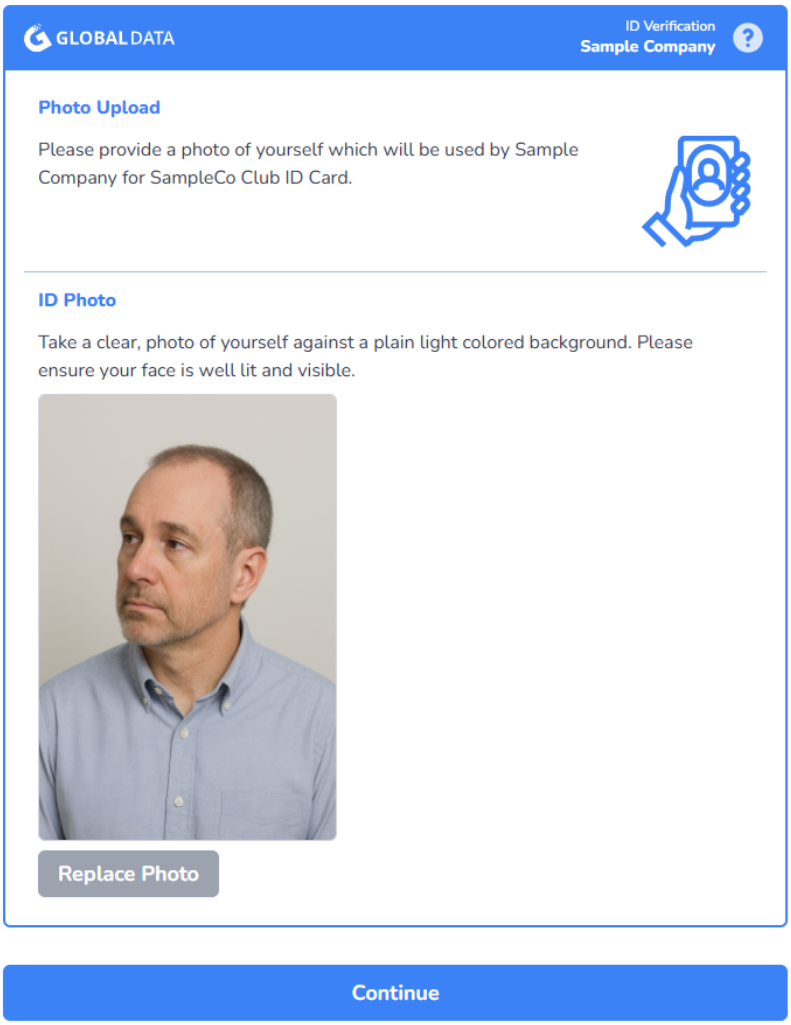
The user is requested to take or upload a photo of themselves to use on their ID card. This optional step is only included if the ID Photo option is selected during configuration. This function can be used if you are producing an ID card with the users photo or require a photo of the user for other purposes.

If you require a photo of the user, then it is essential to include this step as the photo (probe image) provided by the liveness check is not suitable for use as an ID photo.

When requesting an ID Photo, the ID Photo Purpose is also required and this is displayed on the page to the user. Note that in this example the ID Photo Purpose is set to "SampleCo Club ID Card", and this is what is displayed on the page.

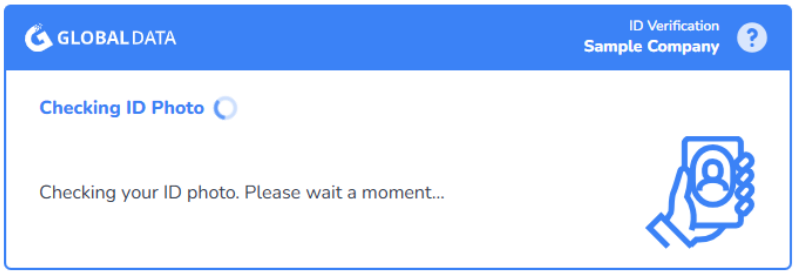


The user can then take a photo or upload an image from their drive or photo library.



ID Photo Checking

The photo is then checked to ensure it is a clear and focused photo of the user. The checking process takes a few seconds.



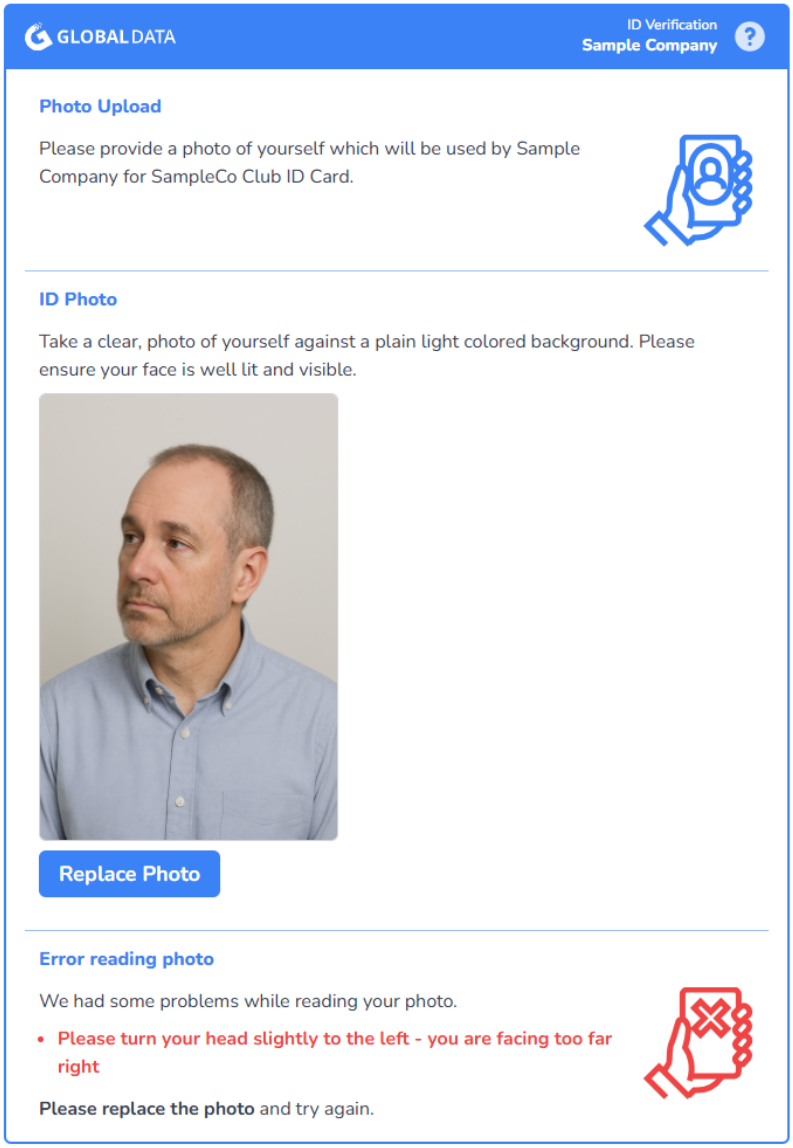
The system checks for a number of different criteria to ensure the photo is valid.

Check	Message
Invalid image file	We could not process your ID photo. Please try again.
No face found	Could not find a face in the image. Please ensure your face is visible.
Multiple faces found	Multiple faces found in the image. Please ensure only your face is visible.
Face occluded	Please ensure your face is fully visible.
Pitch too high	Please tilt your head down slightly - your chin is too high.
Pitch too low	Please raise your chin slightly - you are looking too far down.
Yaw too high	Please turn your head slightly to the right - you are facing too far left.
Yaw too low	Please turn your head slightly to the left - you are facing too far right.
Roll too high	Please straighten your head - it is tilted too far to the left.
Roll too low	Please straighten your head - it is tilted too far to the right.
Brightness too low	Please ensure the image is well lit.
Sharpness too low	The image is too blurry.
Eyes closed	Please ensure your eyes are open.
Sunglasses	Please remove your sunglasses.
Biometric Check Failed	This does not appear to be a photo of you

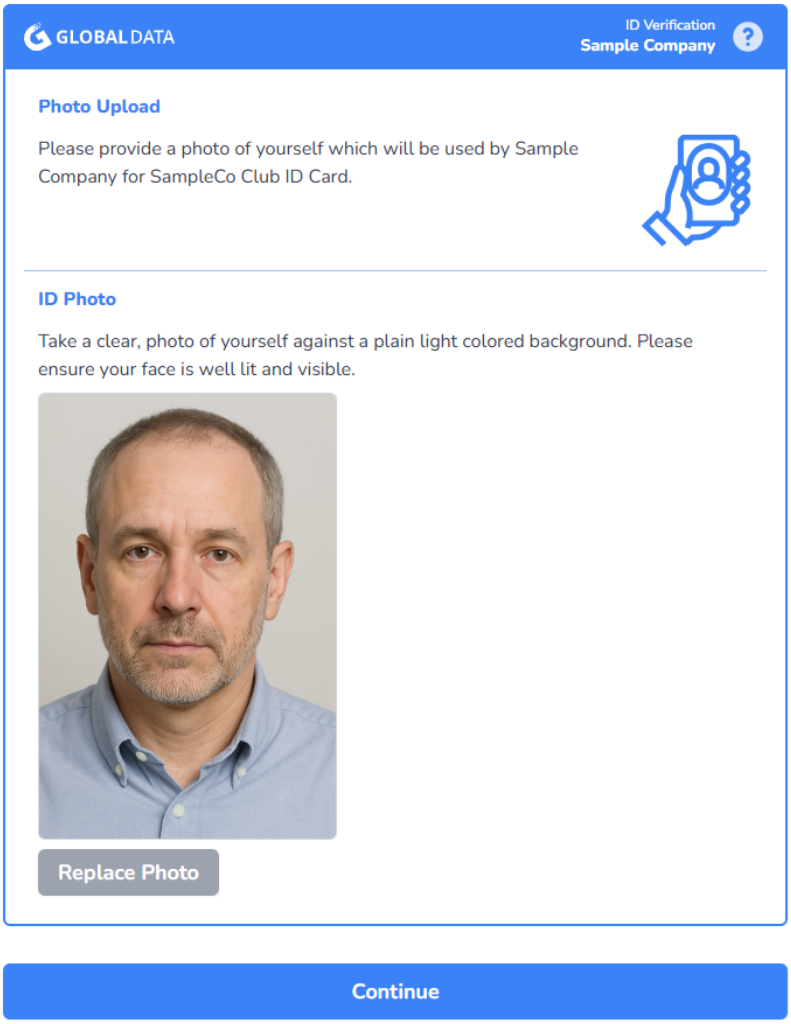
The final check is a biometric check to ensure the photo is of the user. This check is only performed if the liveness check was performed during configuration and will compare the photo with the probe image from the liveness check to ensure they are the same person.

ID Photo Error

If an error is detected then the user is presented with an error screen and asked to try again.

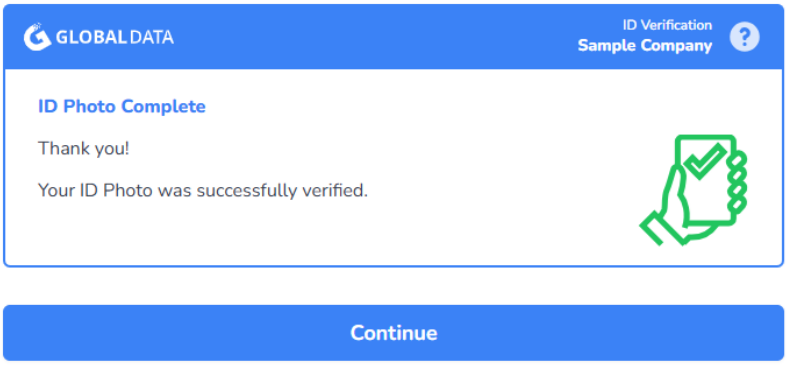


The user can then replace the photo and try again.



ID Photo Complete

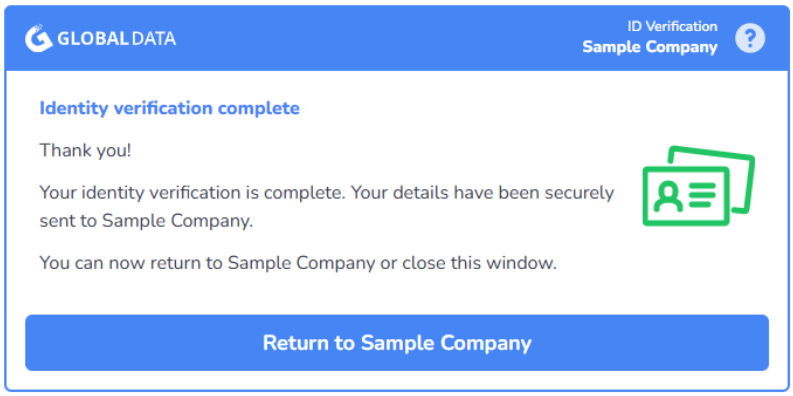
If the photo is clear and focused, then the user is presented with a success screen.



ID Pass Complete

The user has completed all requirements of the ID Pass.

If a return URL was provided during the ID Pass configuration, then the final page will contain a button which will return the user to that URL.



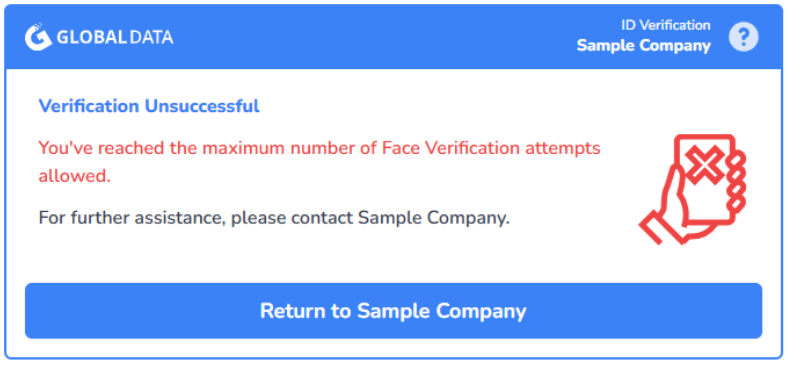
The Completed ID Pass details can now be retrieved and reviewed.

Possible error screens

There are several possible error conditions for an ID Pass.

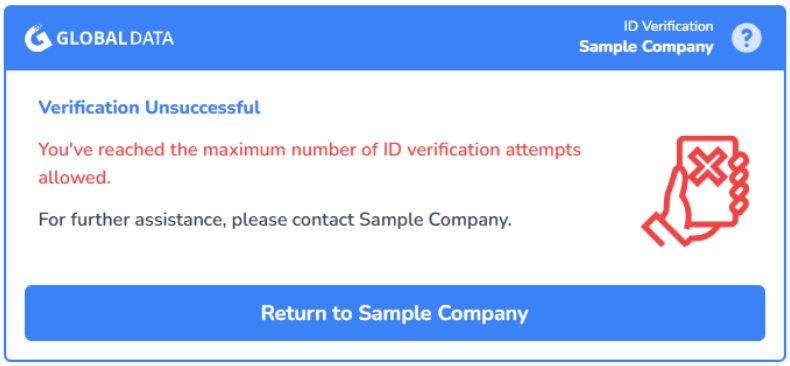
Max Face Verifications Exceeded

For security reasons, the number of face verification attempts is limited. This limit changes dynamically depending on the relative trust of the IP address, mobile device and browser. A minimum of 3 attempts are permitted before the user receives an error screen. Once this error occurs then ID Pass is no longer usable and a new ID Pass will need to be generated to allow the user to try again.



Max DVS Verification Attempts Exceeded

The number of DVS ID verification attempts is limited to 3 per document. If a user is unable to verify their document after 3 attempts, then they receive an error screen. Once this error occurs then the ID Pass is no longer usable, and a new ID Pass will need to be generated to allow the user to try again.



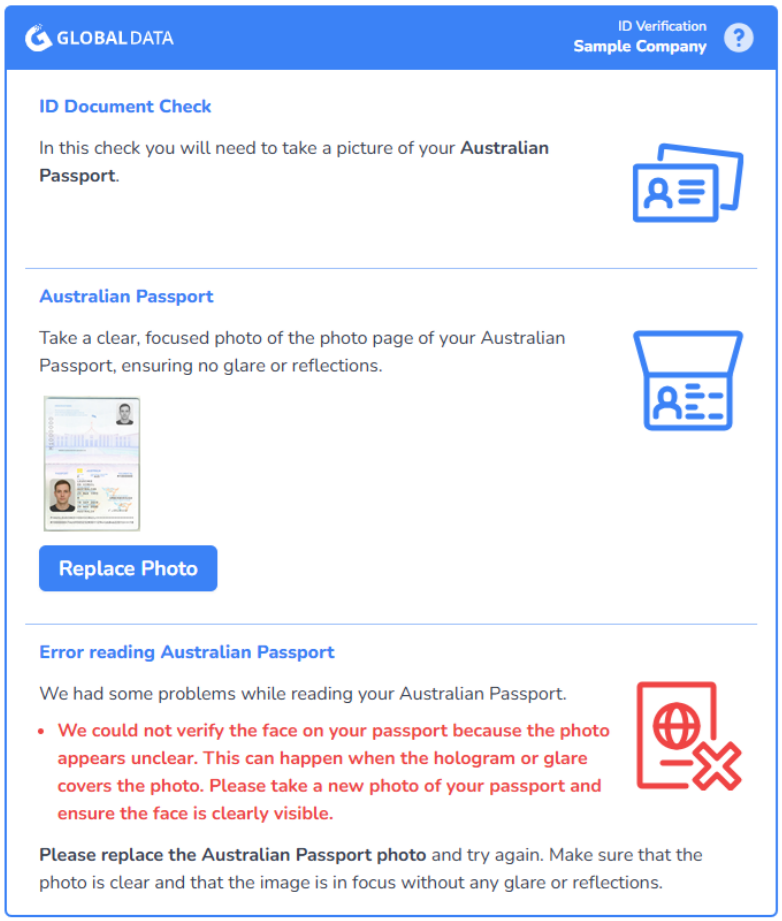
Document Face Occluded

During the document reading step, the system extracts the face image from the uploaded ID document and compares it biometrically against the liveness probe image. If the face on the document cannot be verified because the photo appears unclear, typically caused by a hologram overlay or glare covering the face, the user is shown an error on the document upload screen.

The error message is specific to the document type. For example, for an Australian Passport:

> *We could not verify the face on your Australian Passport because the photo appears unclear. This can happen when the hologram or glare covers the photo. Please take a new photo of your Australian Passport and ensure the face is clearly visible.*

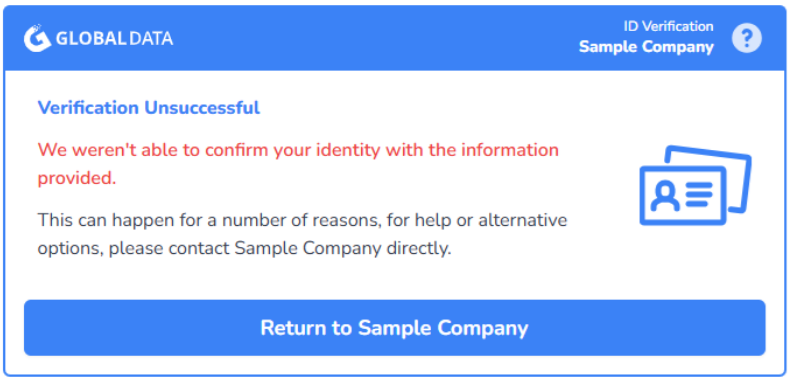
The user is prompted to replace the document photo and try again. This is treated as a document upload error, so the user can retry without consuming a DVS verification attempt.



Verification Failed

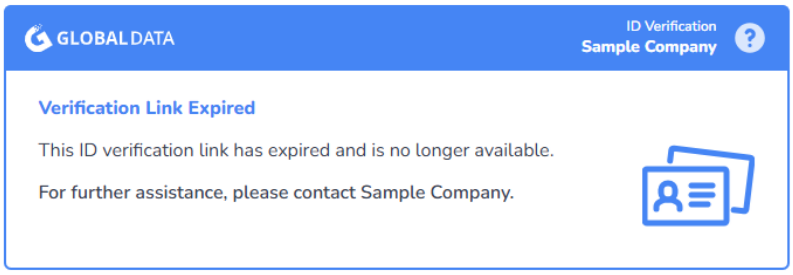
The ID Pass verification has failed. This can be due to a number of reasons, including:

- The user has failed liveness verification
- The document face image did not match the users face (biometric failure)
- The document data did not pass biographic validation



ID Pass Expired

The ID Pass validity period is set to a preset number of days when the ID Pass is configured. If the ID Pass has not been completed in this time, then it is considered to be expired, and the user will receive the expired screen.

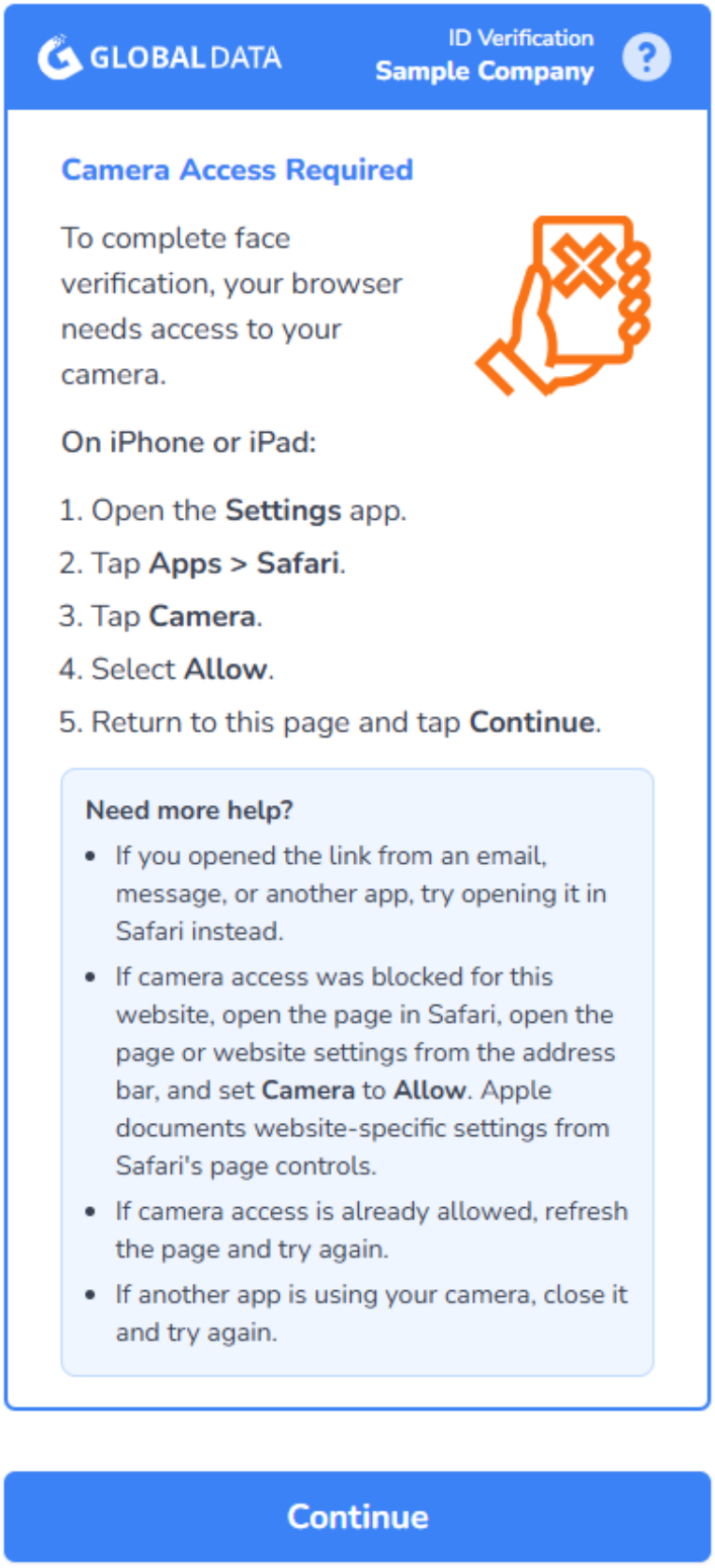


Camera Access Error

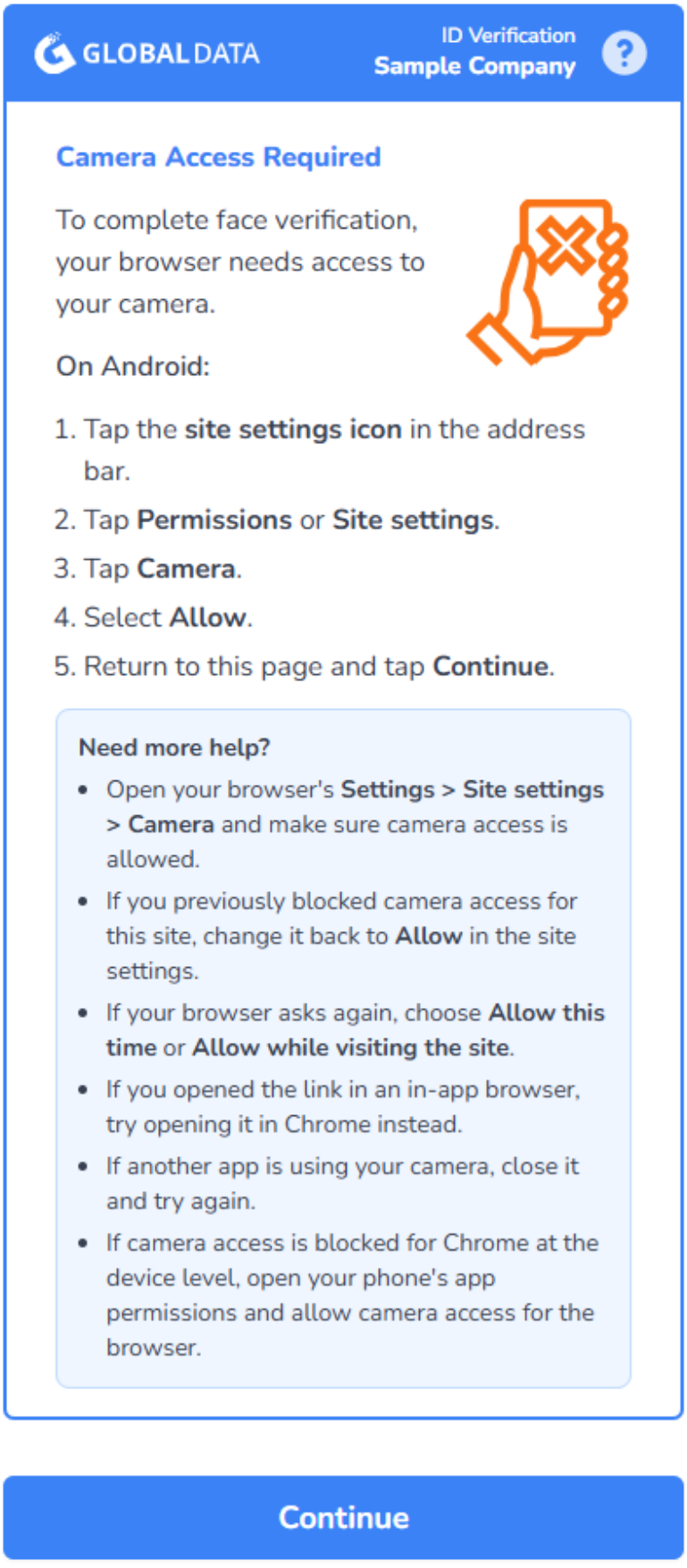
If the user denies camera access or the camera is unavailable during the face verification step, the system detects the error and displays a device-specific help page with instructions on how to enable camera access. The help page content is tailored to the user's device (iOS, Android, or desktop) and provides step-by-step instructions for their platform.

The user can follow the instructions to enable camera access, then tap **Continue** to return to the face verification step and try again.

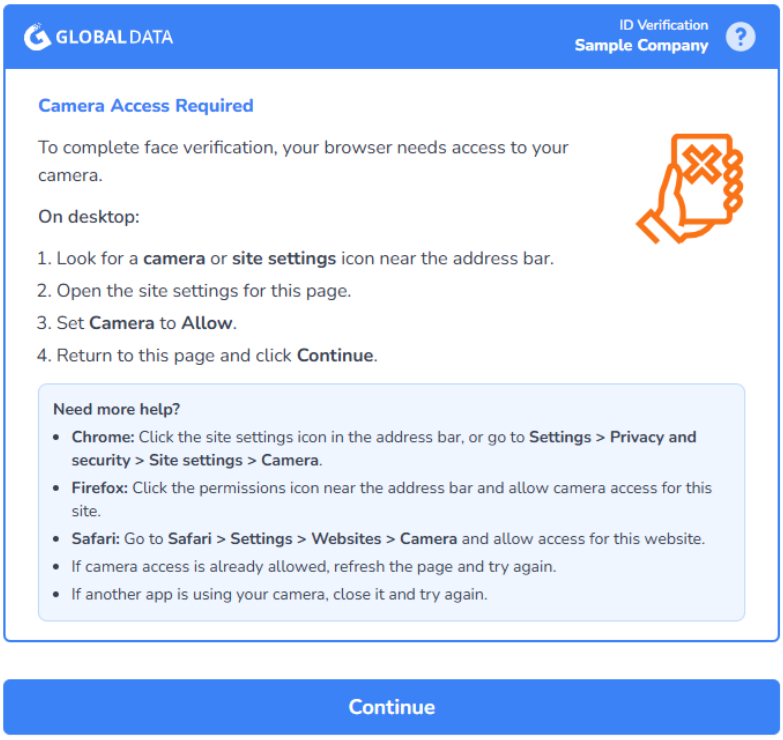
iPhone / iPad:


GLOBAL DATA
ID Verification
Sample Company
Camera Access Required
To complete face verification, your browser needs access to your camera.
On iPhone or iPad:
1. Open the Settings app.
2. Tap Apps > Safari.
3. Tap Camera.
4. Select Allow.
5. Return to this page and tap Continue.
Need more help?
If you opened the link from an email, message, or another app, try opening it in Safari instead.
If camera access was blocked for this website, open the page in Safari, open the page or website settings from the address bar, and set Camera to Allow. Apple documents website-specific settings from Safari's page controls.
If camera access is already allowed, refresh the page and try again.
If another app is using your camera, close it and try again.
Continue

Android:


GLOBAL DATA
ID Verification
Sample Company
Camera Access Required
To complete face verification, your browser needs access to your camera.
On Android:
1. Tap the site settings icon in the address bar.
2. Tap Permissions or Site settings.
3. Tap Camera.
4. Select Allow.
5. Return to this page and tap Continue.
Need more help?
• Open your browser's Settings > Site settings > Camera and make sure camera access is allowed.
• If you previously blocked camera access for this site, change it back to Allow in the site settings.
• If your browser asks again, choose Allow this time or Allow while visiting the site.
• If you opened the link in an in-app browser, try opening it in Chrome instead.
• If another app is using your camera, close it and try again.
• If camera access is blocked for Chrome at the device level, open your phone's app permissions and allow camera access for the browser.
Continue

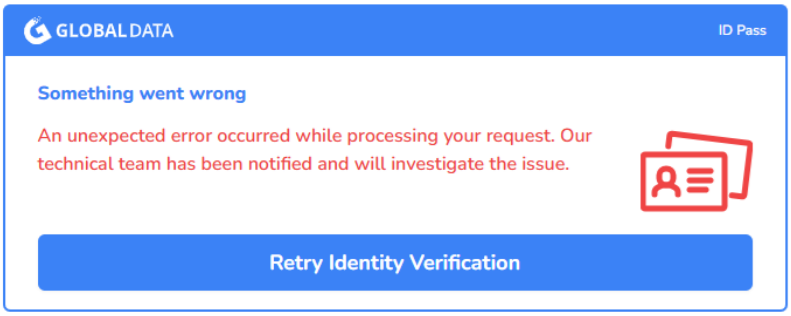
Desktop:



General Error

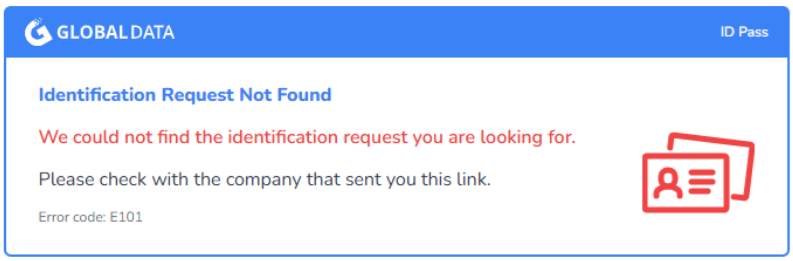
An unexpected error occurred while processing the ID Pass.

Any such error will log an automatic notification to the technical team and the user will be presented with a general error screen.



ID Pass Not Found

The ID Pass was not found. This can happen if the ID Pass has been deleted or the link has been copied incorrectly.



Custom Identity Verification

This guide shows how to structure a custom identity verification journey using four API endpoints:

- `POST /api/v2/liveness_check`
- `POST /api/v2/id_extract`
- `POST /api/v2/dvs/passport` (or other DVS endpoints depending on the document type)
- `POST /api/v2/likeness_check`

Quick Start

The following is a typical orchestration for a custom identity verification flow that includes liveness, ID document OCR, ID document verification with DVS and face comparison:

1. Create a liveness check link.
2. Send the customer to that link and wait for completion.
3. Fetch the liveness result (including a probe image).
4. Collect identity document images in your app.
5. Run `id_extract` on the document images to get biographic data and the ID photo on the document.
6. Run a DVS check using the biographic data obtained from the ID document.
7. Compare `probe_image` vs extracted licence/passport `photo` using `likeness_check`.
8. Apply your business decision rules and store the outcome.

1) Liveness

Create a liveness check:

```
POST /api/v2/liveness_check
```

Recommended request:

```
{
  "return_url": "https://yourapp.example.com/verify/return",
  "webhook_url": "https://yourapp.example.com/webhooks/liveness",
  "link_valid_mins": 10,
  "threshold": 70,
  "audit_images": 0
}
```

You will receive:

- `token`
- `url` (send user here)
- `secret` (for webhook signature verification)

The token is a reference to the liveness session and is used to retrieve results. Send the user to `url`.

The user will complete the liveness flow on our hosted UI. You will receive a webhook on completion or expiry, but you can also poll for results as described in the next step.

2) Retrieve liveness result safely

When the user returns to your app, you may receive `lc_token` in query params to help identify the session the user was using. As it comes from the user, treat it as untrusted input.

Verify the results from the server-side by calling:

```
GET /api/v2/liveness_check?token={token}
```

Use your server-stored liveness token from session creation when calling GET result.

Possible outcomes:

- `202` : still `pending` or `in_progress` (keep polling)
- `200` : complete with `result`
- `410` : expired

Use the response `result` fields:

- `passed`
- `confidence`
- `threshold`
- `image` (probe image; short-lived)

For this workflow, continue only when:

- `passed = true`
- `image` exists (probe image required for face match)

3) Collect ID document images

In your app, ask the user for images of the identity document. We are able to perform OCR on Australian passport, driver licences and Medicare cards.

For a licence, you typically need the front and back images to extract all required fields. Medicare cards only need the front image. For a passport, the an image of the photo page is required.:

- `document_photo` (front)
- `document_back` (back - only for licences)

Then call:

```
POST /api/v2/id_extract
```

Example request for a driver licence:

```
{
  "document_type": "licence",
  "document_photo": "...",
  "document_back": "..."
```

```
}
```

Success (`200`) returns extracted fields in `result` , including:

- Biographic fields (`first_name` , `last_name` , `birth_date` , etc.)
- `photo` (cropped face from the document, extracted during OCR)

If `photo` is missing or extraction fails, ask for better images and retry.

4) Run DVS check

Confirm the biographical data with the user and, with their consent, use the data to run a DVS check:

```
POST /api/v2/dvs/passport
```

Example request:

```
{
    "first_name": "ED VIRGIL",
    "last_name": "LEUSCHKE",
    "birth_date": "1993-03-23",
    "travel_document_number": "M1000000",
    "consent": true
}
```

5) Run face similarity check

If you are working with a licence or passport, you will have a photo extracted from the document. You can compare this with the liveness probe image to confirm the person in front of the camera is the same as the person on the ID document.

Compare the liveness probe image with the extracted ID document photo using `likeness_check` :

```
POST /api/v2/likeness_check
```

Example request:

```
{
  "photo": "...",
  "probe_image": "...",
  "similarity_threshold": 80
}
```

`200` response includes:

- `result.similarity`
- `result.threshold`
- `result.biometric_passed`

6) Make a final decision

A common decision policy:

- Liveness must pass

- Likeness must pass
- DVS Identity Validation must pass

Any significant editing from the user of the biographic data extracted from the ID document should be treated as a risk signal and may require additional review or rejection.

7) Security, privacy and consent (recommended)

This workflow processes sensitive identity and biometric data (for example, liveness probe images and document photos). You should:

- Obtain clear user consent before running DVS and biometric checks.
- Use HTTPS/TLS for all API calls and webhook endpoints.
- Verify webhook signatures before trusting webhook events.
- Restrict access to identity data to only the systems and staff that need it.
- Keep only the minimum data needed for compliance and audit requirements, then securely delete data you no longer need.
- Avoid storing raw base64 images long term unless required by your regulatory obligations.

API Reference

The Global Data API allows external programmatic access to the full functionality of the Global Data suite of validation,verification and search functions. The API accesses the same dataset and will produce the same results as the api.globaldata.net.au web system.

Sandbox vs Live

The GlobalData API provides two environments: Sandbox and Live.

- Use the Sandbox API server when working with a Sandbox API key.
- Use the Live API server when working with a Live API key.

Requests made with a key to the wrong environment will return a 401 Unauthorized error.

Servers

URL	Description
`https://gdapi.globaldata.net.au/api/v2`	Live API V2 endpoint
`https://sandbox-gdapi.globaldata.net.au/api/v2`	Sandbox API V2 endpoint

Authentication

BearerAuth: http (bearer)

Address Autocomplete

GET `/address_autocomplete`

Address Auto-Complete is a powerful API query that can save users time and reduce errors by predicting and suggesting Australian addresses as they are being typed. This feature can be integrated into any web or mobile application to streamline data entry and improve overall data accuracy.

The address autocomplete method is used for performing address autocomplete on user entry forms using AJAX or similar. This method will return matching addresses when sent the starting characters of an address.

Match Types

The following match types are supported:

- `address` - Match only full addresses. This search returns an address with a GNAF id
- `street` - Match only street names. This search returns a list of matching streets with suburb, postcode and state
- `suburb` - Match only suburb names. This search returns a list of matching suburbs with postcode and state
- `postcode` - Match only postcodes. This search returns a list of suburbs within the existing postcode, and the state and postcode

Sandbox environment

When simulating queries in the sandbox environment, a restricted dataset is used and a number of responses are possible. The response will be determined by the partial_address:

Type	Partial Address	Response
address	Any even street number	A valid search will be performed
address	Any odd street number	No matches will be found
postcode	Any even postcode	A valid search will be performed and returned
postcode	Any odd postcode	No matches will be found
suburb	Any suburb beginning with A - H	A valid search will be performed and returned
suburb	Any suburb beginning with I - Z	No matches will be found
street	Any street beginning with A - H	A valid search will be performed and returned
street	Any street beginning with I - Z	No matches will be found

> In the production environment, the full Australian address database will be used to generate matches.

Request body

The details of the record to enhance

Field	Description
`partial_address`	**string [4..100 characters]** The first 4 or more characters of the address/street/suburb/postcode to find **Example:** `123 Smith`
`type`	**string or null** The type of address match to find - `address` - Match only full addresses. This search returns an address with a GNAF id - `street` - Match only street names. This search returns a list of matching streets with suburb, postcode and state - `suburb` - Match only suburb names. This search returns a list of matching suburbs with postcode and state - `postcode` - Match only postcodes. This search returns a list of suburbs within the existing postcode, and the state and postcode **Enum:** `address`, `street`, `suburb`, `postcode` **Default:** `address` **Example:** `address`
`max_results`	**integer [1..100] or null** The maximum number of results to return **Default:** `20` **Example:** `20`

Sample request

```
{
    "partial_address": "123 Smith",
    "type": "address",
    "max_results": 20
}
```

Responses

200 **Successful 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. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`matches`	**array of objects** Array of matched addresses
`matches[].type`	**string** The type of match returned **Example:** `address`
`matches[].address_id`	**string** The GNAF address id of the address **Example:** `GAACT714845933`
`matches[].address_option`	**string** Complete match as a single line ready to pass to a SELECT element **Example:** `6 PACKHAM PL CHARNWOOD ACT 2615`
`matches[].address_combined`	**string** The combined street address as one string. eg *2/10 Paper St*. This attribute is only returned for types address and street requests **Example:** `6 PACKHAM PL`
`matches[].suburb`	**string** The matched suburb **Example:** `CHARNWOOD`
`matches[].postcode`	**string** The matched postcode **Example:** `2615`

matches[]. state	**string** The matched state **Example:** `ACT`
more_results	**boolean** Indicates if there are more than *max_results* available. **Example:** `false`

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "matches": [
        {
            "type": "address",
            "address_id": "GAACT714845933",
            "address_option": "6 PACKHAM PL CHARNWOOD ACT 2615",
            "address_combined": "6 PACKHAM PL",
            "suburb": "CHARNWOOD",
            "postcode": "2615",
            "state": "ACT"
        }
    ],
    "more_results": false
}
```


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

Address Validate

POST `/address_validate`

The Address Validate method tests the supplied address and returns the matched address and a score from 0 to 10 indicating the quality of the match.

Corrections to the address

The address validator will attempt to match the address to the GNAF address database and make corrections to the supplied address if necessary in order to find the closest match.

Permitted changes include:

- Spelling corrections to the street name. For example, "10 Smyth Street" to "10 Smith Street" (permitted if the street number and street name are valid in the given suburb / postcode)
- Changes to the street type. For example, "10 Paper Rd" to "10 Paper St" (permitted if the street number and street name are valid in the given suburb / postcode)
- Changes to the unit type. For example, "Flat" to "Unit"

- Changes to the postcode. For example, "3125" to "3124" (permitted if the street number and street name are valid in an adjacent postcode)
- Spelling corrections to the suburb name. For example, "Smithtown" to "Smithton" (permitted if the street number and street name are valid in the given suburb)
- Changes to the suburb if the street number and street name are valid in an adjacent suburb.
- Changes to the street number if the street number is a range and one of the numbers in the range is matched. For example, "123 Smith St" to "123-125 Smith St"

When changes are made to the supplied address, the `address_changes` array will be returned with the details of the changes made and the `matched_address` will be returned with the corrections applied.

Minimum Validation Requirements

The following combinations of parameters are required on order to validate an address:

- `street_address` plus at least one of `suburb` or `postcode`
- `street_address` plus `suburb_state_postcode`
- `full_address`

Request body

The address to validate

Field	Description
`street_address`	**string [4..100 characters] or null** First line of the street or postal address Requires at least one of suburb or postcode to be provided **Example:** `Unit 12B, 123-125 Smith St`
`suburb`	**string [2..60 characters] or null** Suburb name Requires street_address to be provided **Example:** `Smithtown`
`state`	**string or null** State Requires street_address to be provided **Enum:** `ACT`, `NSW`, `NT`, `QLD`, `SA`, `TAS`, `VIC`, `WA` **Example:** `ACT`
`postcode`	**string [3..4 characters] or null** Postcode Requires street_address to be provided **Example:** `2000`

Field	Description
`suburb_state_postcode`	**string [4..100 characters] or null** Combined suburb, state and postcode. Requires street_address to be provided Supports various address formats: • Smithtown, ACT 2000 • Smithtown 2000 ACT **Example:** `Smithtown ACT 2000`
`full_address`	**string [4..255 characters] or null** Full address Supports various address formats: • 123 Smith St, Smithtown, ACT 2000 • Unit 12, 123 Smith Street, Smithtown 2000 ACT • 1/12-34 Smith St, Upper Smithtown, 2000 **Example:** `Unit 12B, 123-125, Smithtown, ACT 2000`

Sample request

```
{
    "street_address": "Unit 12B, 123-125 Smith St",
    "suburb": "Smithtown",
    "state": "ACT",
    "postcode": "2000",
    "suburb_state_postcode": "Smithtown ACT 2000",
    "full_address": "Unit 12B, 123-125, Smithtown, ACT 2000"
}
```

Responses

200 **Successful 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. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Field	Description
`valid`	**boolean** Indicates if the address is valid **Example:** `true`
`score`	**integer or null** A score from 0 to 10 indicating the quality of the match with 0 being the worst match and 10 being the best match or null if the address is not valid. The score is calculated as: 10 - number of address changes (with a minimum of 0) **Example:** `9`
`matched_address`	**object** The matched address or an empty object if the address is not valid
`matched_address.` `address_id`	**string** The GNAF address id of the address **Example:** `GAACT714845933`
`matched_address.` `full_address`	**string** The full address **Example:** `Unit 12B, 123-125 Smith St, Smithtown, ACT 2000`
`matched_address.` `address_line_1`	**string** The first line of the address **Example:** `Unit 12B, 123-125 Smith St`
`matched_address.` `address_line_2`	**string** The second line of the address **Example:** `Smithtown ACT 2000`
`matched_address.` `suburb`	**string** The suburb of the address **Example:** `Smithtown`
`matched_address.` `state`	**string** The state of the address **Example:** `ACT`

Field	Description
matched_address. postcode	**string** The postcode of the address **Example:** `2000`
matched_address. unit_type	**string or null** The type of unit of the address eg: 'Flat' in the address: "Flat AA12B, 100 Some St" **Example:** `U`
matched_address. unit_number_prefix	**string or null** The prefix of the unit number eg: 'AA' in the address: "Flat AA12B, 100 Some St"
matched_address. unit_number	**string or null** The unit number of the address eg: '12' in the address: "Flat AA12B, 100 Some St" **Example:** `12`
matched_address. unit_number_suffix	**string or null** The suffix of the unit number eg: 'B' in the address: "Flat AA12B, 100 Some St" **Example:** `B`
matched_address. level_type	**string or null** The type of level of the address eg: 'L' in the address: "L A12F, 100 Some St"
matched_address. level_number_prefix	**string or null** The prefix of the level number eg: 'A' in the address: "L A12F, 100 Some St"
matched_address. level_number	**string or null** The level number of the address eg: '12' in the address: "L A12F, 100 Some St"
matched_address. level_number_suffix	**string or null** The suffix of the level number eg: 'F' in the address: "L A12F, 100 Some St"
matched_address. street_number_1_prefix	**string or null** The prefix of the street number eg: 'B' in the address: "B100C - D110E Some St"
matched_address. street_number_1	**string or null** The numeric part of the first street number eg: '100' in the address: "B100C - D110E Some St" **Example:** `123`

Field	Description
matched_address. street_number_1_suffix	**string or null** The suffix of the first street number eg: 'C' in the address: "B100C - D110E Some St"
matched_address. street_number_2_prefix	**string or null** The prefix of the second street number eg: 'D' in the address: "B100C - D110E Some St"
matched_address. street_number_2	**string or null** The numeric part of the second street number eg: '110' in the address: "B100C - D110E Some St" **Example:** `125`
matched_address. street_number_2_suffix	**string or null** The suffix of the second street number eg: 'E' in the address: "B100C - D110E Some St"
matched_address. lot_number_prefix	**string or null** The prefix of the lot number eg: 'F' in the address: "Lot F200G Some St"
matched_address. lot_number	**string or null** The numeric part of the lot number eg: '200' in the address: "Lot F200G Some St"
matched_address. lot_number_suffix	**string or null** The suffix of the lot number eg: 'G' in the address: "Lot F200G Some St"
matched_address. postal_delivery_type	**string or null** The type of postal delivery eg: 'PO Box' in the address: "PO Box H1234J"
matched_address. postal_delivery_number_prefix	**string or null** The prefix of the postal delivery number eg: 'H' in the address: "PO Box H1234J"
matched_address. postal_delivery_number	**string or null** The numeric part of the postal delivery number eg: '1234' in the address: "PO Box H1234J"
matched_address. postal_delivery_number_suffix	**string or null** The suffix of the postal delivery number eg: 'J' in the address: "PO Box H1234J"

Field	Description
matched_address. street_name	**string or null** The name of the street eg: 'Smith St' in the address: "123 Smith St North" **Example:** `Smith`
matched_address. street_type	**string or null** The type of street eg: 'St' in the address: "123 Smith St North" **Example:** `ST`
matched_address. street_suffix	**string or null** The suffix of the street eg: 'North' in the address: "123 Smith St North"
address_changes	**array of objects** Array of changes made to the provided address in order to match with the GNAF address
address_changes[]. field	**string** The address field that was changed **Example:** `street_type`
address_changes[]. change	**string** The change made to the address field **Example:** `Changed from ST to PL`

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "valid": true,
    "score": 9,
    "matched_address": {
        "address_id": "GAACT714845933",
        "full_address": "Unit 12B, 123-125 Smith St, Smithtown, ACT 2000",
        "address_line_1": "Unit 12B, 123-125 Smith St",
        "address_line_2": "Smithtown ACT 2000",
        "suburb": "Smithtown",
        "state": "ACT",
        "postcode": "2000",
        "unit_type": "U",
        "unit_number_prefix": null,
        "unit_number": "12",
        "unit_number_suffix": "B",
        "level_type": null,
        "level_number_prefix": null,
        "level_number": null,
```

```
        "level_number_suffix": null,
        "street_number_1_prefix": null,
        "street_number_1": "123",
        "street_number_1_suffix": null,
        "street_number_2_prefix": null,
        "street_number_2": "125",
        "street_number_2_suffix": null,
        "lot_number_prefix": null,
        "lot_number": null,
        "lot_number_suffix": null,
        "postal_delivery_type": null,
        "postal_delivery_number_prefix": null,
        "postal_delivery_number": null,
        "postal_delivery_number_suffix": null,
        "street_name": "Smith",
        "street_type": "ST",
        "street_suffix": null
    },
    "address_changes": [
        {
            "field": "street_type",
            "change": "Changed from ST to PL"
        }
    ]
}
```


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

Address Validate Bulk

POST `/address_validate_bulk`

Bulk variant of Address Validate. This endpoint accepts up to **200** address validation requests in a single call.

Each item in the `requests` array uses the same validation rules and fields as `address_validate`:

- `street_address` plus at least one of `suburb` or `postcode`
- `street_address` plus `suburb_state_postcode`
- `full_address`

If `check_id` is provided per request, it will be echoed back in the corresponding `results` item.

Request body

The list of addresses to validate

Field	Description
`requests`	**array of objects [1..200 items]** The list of address validation requests to make.

Field	Description
requests[]. check_id	**string** The unique identifier for this request. This can be used to match the request in the results. **Example:** `check_123`
requests[]. street_address	**string [4..100 characters] or null** First line of the street or postal address. Requires at least one of suburb or postcode to be provided. **Example:** `Unit 12B, 123-125 Smith St`
requests[]. suburb	**string [2..60 characters] or null** Suburb name. Requires street_address to be provided. **Example:** `Smithtown`
requests[]. state	**string or null** State. Requires street_address to be provided. **Enum:** `ACT`, `NSW`, `NT`, `QLD`, `SA`, `TAS`, `VIC`, `WA` **Example:** `ACT`
requests[]. postcode	**string [3..4 characters] or null** Postcode. Requires street_address to be provided. **Example:** `2000`
requests[]. suburb_state_postcode	**string [4..100 characters] or null** Combined suburb, state and postcode. Requires street_address to be provided. Supports various address formats: - Smithtown, ACT 2000 - Smithtown 2000 ACT **Example:** `Smithtown ACT 2000`
requests[]. full_address	**string [4..255 characters] or null** Full address. Supports various address formats: - 123 Smith St, Smithtown, ACT 2000 - Unit 12, 123 Smith Street, Smithtown 2000 ACT - 1/12-34 Smith St, Upper Smithtown, 2000 **Example:** `Unit 12B, 123-125 Smith St, Smithtown, ACT 2000`

Sample request

```
{
    "requests": [
        {
            "check_id": "check_123",
            "street_address": "Unit 12B, 123-125 Smith St",
            "suburb": "Smithtown",
            "state": "ACT",
            "postcode": "2000",
            "suburb_state_postcode": "Smithtown ACT 2000",
            "full_address": "Unit 12B, 123-125 Smith St, Smithtown, ACT 2000"
        }
    ]
}
```

Responses

200 **Successful 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`
`function`	**string** The function that was called. **Example:** `address_validate_bulk`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`results`	**array of objects** A list of results for each of the requests made.
`results[].` `check_id`	**string or null** The unique identifier for this request. This can be used to match the request in the results. **Example:** `check_123`

Field	Description
results[]. valid	**boolean or null** Indicates if the address is valid. **Example:** `true`
results[]. score	**integer or null** A score from 0 to 10 indicating the quality of the match with 0 being the worst match and 10 being the best match, or null if the address is not valid. **Example:** `10`
results[]. matched_address	**object** The matched address or an empty object if the address is not valid.
results[]. address_changes	**array of objects** Array of changes made to the provided address in order to match with the GNAF address.
results[]. address_changes[]. field	**string** **Example:** `street_type`
results[]. address_changes[]. change	**string** **Example:** `Changed 'St' to 'Rd'`
results[]. error	**string** Present if an internal error occurred processing this item. **Example:** `Internal error`

Sample response


```
{
    "message": "Ok",
    "function": "address_validate_bulk",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "results": [
        {
            "check_id": "check_123",
            "valid": true,
            "score": 10,
            "matched_address": {
                "address_id": "GAACT714845933",
                "full_address": "Unit 12B, 123-125 Smith St, Smithtown, ACT 2000"
            },
            "address_changes": [
                {
                    "field": "street_type",
                    "change": "Changed 'St' to 'Rd'"
```

```
                }
            ],
            "error": "Internal error"
        }
    ]
}
```

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

Address Lookup by Contact

GET `/address_lookup_by_contact`

The address lookup by contact query will return the GNAF ID which corresponds to a given email address or phone number from the Global Data universe.

Note: If both a phone number and email address are supplied, the phone number will searched first, and if no match is found, the email address will be searched.

Sandbox environment

When simulating queries in the sandbox environment, the following records will return the matching GNAF ID:

Email	Phone	GNAF ID
jsmith1998@example.com	0399999999, 0491222111	GAVIC421647320
	0399998888	GAVIC419608268
mary_jones89@example.com	0399995000	GANSW716615055
rp_brown71@example.com		GANSW712900961

Query parameters

Field	Description
`email`	**string** The email address to search for. Email addresses are not case sensitive.
`phone`	**string** The phone number to search for. Phone numbers must be in 0NSN format (10 digits, no spaces or other intermediate characters and starting with a 0) eg 0299995000, or 0491570006. **Example:** `0491222111`

Responses

200 **Result of address lookup** (application/json)

Field	Description
`message`	**string** A message indicating the result of the request. This will be `Ok` if the request was successful. or `No record` if the an address could not be found. **Example:** `Ok`
`address_id`	**string** The GNAF address id of the address (supplied if the request was successful) **Example:** `GANSW716615055`
`record_date`	**string (date)** The date the record was create or last updated **Example:** `2021-01-01`
`api_reference`	**string (uuid)** A unique identifier for this request **Example:** `738d9a89-b6fe-4fc1-96be-1389b2a506a2`

Sample response

```
{
    "message": "Ok",
    "address_id": "GANSW716615055",
    "record_date": "2021-01-01",
    "api_reference": "738d9a89-b6fe-4fc1-96be-1389b2a506a2"
}
```

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

Property Detail

GET `/property_detail`

The property detail query is designed to provide the resident and contact details (phone and email) linked to an address referenced by a GNAF ID in the Global Data universe.

The query will automatically perform a connectivity check on any phone numbers and email addresses linked to the resident. A DNC (Do Not Call) register check will also be performed and the results returned with the response.

The connectivity and DNC checks can take some time to complete and this would cause the query to block until the results are available. By providing the optional timeout parameter, the query will return at the end of the timeout even if the phone and email checks have not yet completed. The query can then be retried at no cost by providing the original api_reference.

Service	Typical response time	Maximum response time
Address resolution and resident lookup	< 100mS	< 1 second
Phone connectivity check	< 1 second	25 seconds (for some older land line exchanges)
Phone DNC check	< 2 seconds	10 seconds
Email deliverability check	< 3 seconds	25 seconds

After the initial address and resident lookup, all phone and email checks are performed in parallel, so the total blocking time is dependent on the slowest response.

Polling for updates

Using a polling methodology allows for an improved user experience whereby the initial query can return quickly with the resident data, and the application can then poll for completion of the email and phone checks.

If the api_reference is provided then the API will use the data from the previous request and wait for any uncompleted checks to finish. The design purpose of this is to allow a shorter timeout to be set in the initial query which will return the basic resident data and then poll for completion of the phone and email checks.

An example use case:

```
$data = Http::get("https://gdapi.globaldata.net.au/api/v2/property_detail?" .
    "api_key={$api_key}&" .
    "gnaf_id={$gnaf_id}"
)->object();

// Render completed output to UI
...

while (! $data->complete) {
    $data = Http::get("https://gdapi.globaldata.net.au/api/v2/property_detail?" .
        "api_key={$api_key}&" .
        "gnaf_id={$gnaf_id}&" .
        "api_reference={$data->api_reference}"
    )->object();

    // Push update to UI
    ...
}
```

Sandbox environment

When simulating queries in the sandbox environment the system will use a reduced sample dataset. The following GNAF IDs will return resident and contact details (other sample addresses may return the address only with no records):

GNAF ID	Address	Suburb	State	Postcode
GANSW716615055	35 YARALLA ST	CONCORD WEST	NSW	2138
GANSW712900961	22 BRABYN ST	WINDSOR	NSW	2756
GAVIC421647320	20 HARDY ST	LILYDALE	VIC	3140

GNAF ID	Address	Suburb	State	Postcode
GAVIC419608268	8 WALTHAM ST	RICHMOND	VIC	3121

Query parameters

Field	Description
`gnaf_id` REQUIRED	**string** The address GNAF ID. This is equivalent to the address_detail_pid on the GNAF table
`api_reference`	**string (uuid)** The api_reference returned from a previous API request. Used to poll for updates to the result.
`timeout`	**integer [1..120]** The maximum number of seconds to wait for phone and email checks to complete. If results are not completed before this timeout then the result will be returned with 'complete set to false, which indicates that one or more of the connectivity_status, dnc_status or deliverability_status are still 'processing'. **Default:** `20`

Responses

200 **Result of property detail** (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`
`complete`	**boolean** Indicates if the phone and email checks have completed. If this is false then the connectivity_status, dnc_status and deliverability_status will be set to 'processing'. **Example:** `true`
`api_reference`	**string (uuid)** A unique identifier for this request, used to get an update for the result at a later time **Example:** `738d9a89-b6fe-4fc1-96be-1389b2a506a2`

Field	Description
`total_records_available`	**integer** Number of records in the result set **Example:** `1`
`address`	**object** The address details for GNAF ID
`address.` `address_id`	**string** The GNAF address id of the address **Example:** `GANSW716615055`
`address.` `address`	**string** The combined street address as one string. **Example:** `35 YARALLA ST`
`address.` `suburb`	**string** The matched suburb **Example:** `CONCORD WEST`
`address.` `postcode`	**string** The matched postcode **Example:** `2138`
`address.` `state`	**string** The matched state **Example:** `NSW`
`records`	**array of objects** Resident records for this address
`records[].` `name_title`	**string** Title of matched person or blank if unknown **Example:** `Ms`
`records[].` `name_first`	**string** First name or Initial of matched person or blank if unknown **Example:** `MARY`

Field	Description
records[]. name_middle	**string** Middle name or Initial of matched person or blank if unknown **Example:** `SALLY`
records[]. name_last	**string** Last name of matched person **Example:** `JONES`
records[]. deceased	**string** Indicates if the person is deceased **Enum:** `Y`, `N` **Example:** `N`
records[]. gender	**string** Gender of the person or blank if unknown **Enum:** `M`, `F`, `X`
records[]. date_last_seen	**string (date)** Date the person was last seen at this address or blank if unknown **Example:** `2019-01-01`
records[]. phones	**array of objects** Phone numbers associated with matched person
records[]. phones[]. phone	**string** Phone number in 0NSN format eg 0491570006 **Example:** `0299995000`
records[]. phones[]. connectivity_status	**string** The status of the connectivity check for the phone - `processing` - The check is still in progress - `complete` - The check has completed and the connectivity_result is available - `error` - An error occurred during the check **Enum:** `processing`, `complete`, `error` **Example:** `complete`

Field	Description
records[]. phones[]. connectivity_result	**string** The result of the connectivity check for the phone - `C` - The phone is connected - `D` - The phone is disconnected - `U` - The state of the phone number cannot be detected - `I` - The phone number is invalid **Enum:** `C`, `D`, `U`, `I` **Example:** `C`
records[]. phones[]. dnc_status	**string** The status of the DNC check for the phone: - `processing` - The check is still in progress - `complete` - The check has completed and the dnc_result is available - `error` - An error occurred during the check **Enum:** `processing`, `complete`, `error` **Example:** `complete`
records[]. phones[]. dnc_result	**string** The result of the DNC check for the phone: - `Y` - The phone number is on the DNC register - `N` - The phone number is not on the DNC register - `I` - The phone number is invalid **Enum:** `Y`, `N`, `I` **Example:** `N`
records[]. phones[]. dnc_reference	**string** The DNC wash reference number returned by the DNC register Note: DNC results are valid for 30 days and must be rechecked after that time. **Example:** `123456789`

Field	Description
records[]. phones[]. marketing_opt_in	**boolean** Indicates if the record is opted in to receive marketing - `true` - Record is opted in for marketing - `false` - Record is not opted in for marketing must not be contacted for any promotional or marketing purposes and is supplied for verification only **Example:** `true`
records[]. phones[]. suppression	**string or null** Some records in the Global Data universe have been suppressed at the request of the record owner. If this field is present then the record has been suppressed. All other fields will be blank.
records[]. emails	**array of objects** Email addresses associated with matched person
records[]. emails[]. email	**string** Email address **Example:** `mary_jones89@example.com`
records[]. emails[]. deliverability_status	**string** The status of the deliverability check for the email - `processing` - The check is still in progress - `complete` - The check has completed and the deliverability_result is available - `error` - An error occurred during the check **Enum:** `processing`, `complete`, `error` **Example:** `complete`
records[]. emails[]. deliverability_result	**string** The result of the deliverability check for the email - `Y` - The email address is deliverable - `N` - The email address is not deliverable - `U` - The email address cannot be checked - `I` - The email address is invalid - `E` - An error occurred during the check **Enum:** `Y`, `N`, `U`, `I`, `E` **Example:** `Y`

Field	Description
records[]. emails[]. deliverability_ok_to_send	**string** Should email be sent to this address - `Y` - OK to send - `N` - Do not send - `U` - Unknown Some email addresses will report as deliverable (deliverability_result = 'Y') but not ok to send (deliverability_ok_to_send = 'N'). This can be because the address is a known spam trap or is enrolled in a global block list (serial complainer, etc). Where possible information about this will be provided in the deliverability_detail field. **Enum:** `Y`, `N`, `U` **Example:** `N`
records[]. emails[]. deliverability_detail	**string** Additional detail for the result if available **Example:** `Known spam trap`
records[]. emails[]. marketing_opt_in	**boolean** Indicates if the record is opted in to receive marketing - `true` - Record is opted in for marketing - `false` - Record is not opted in for marketing must not be contacted for any promotional or marketing purposes and is supplied for verification only **Example:** `true`
records[]. emails[]. suppression	**string or null** Some records in the Global Data universe have been suppressed at the request of the record owner. If this field is present then the record has been suppressed. All other fields will be blank.

Sample response


```
{
    "message": "Ok",
    "complete": true,
    "api_reference": "738d9a89-b6fe-4fc1-96be-1389b2a506a2",
    "total_records_available": 1,
    "address": {
        "address_id": "GANSW716615055",
        "address": "35 YARALLA ST",
        "suburb": "CONCORD WEST",
        "postcode": "2138",
        "state": "NSW"
    },
    "records": [
```

```
        {
            "name_title": "Ms",
            "name_first": "MARY",
            "name_middle": "SALLY",
            "name_last": "JONES",
            "deceased": "N",
            "gender": "M",
            "date_last_seen": "2019-01-01",
            "phones": [
                {
                    "phone": "0299995000",
                    "connectivity_status": "complete",
                    "connectivity_result": "C",
                    "dnc_status": "complete",
                    "dnc_result": "N",
                    "dnc_reference": "123456789",
                    "marketing_opt_in": true,
                    "suppression": "string"
                }
            ],
            "emails": [
                {
                    "email": "mary_jones89@example.com",
                    "deliverability_status": "complete",
                    "deliverability_result": "Y",
                    "deliverability_ok_to_send": "N",
                    "deliverability_detail": "Known spam trap",
                    "marketing_opt_in": true,
                    "suppression": "string"
                }
            ]
        }
    ]
}
```


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

Real Estate Search

POST `/realestate_search`

The real estate search query returns **Australian real estate listing history** (sale, sold, and rental events) for a given property address, sourced from the Global Data Australian property universe.

The query accepts an address in one of three formats:

- A **GNAF ID** (the unique GNAF address identifier)
- A **full address** as a single string
- A **structured address** with street_address, suburb, state, and postcode as separate fields

Results are returned in **reverse chronological order** (most recent first).

An optional `date_from` parameter can be provided to filter results to only include records on or after the specified date.

Address resolution

The `address` object in the response represents the resolved address that was used to search the real estate universe:

- When a **GNAF ID** is provided, the returned address is the exact address referenced by that GNAF ID.
- When a **text-based address** is provided (full address or structured address), the system will resolve the input to the best matching address in the GNAF database. The returned address may include minor corrections or standardisations compared to the original input (e.g. corrected spelling, expanded abbreviations, standardised formatting) if these were needed to match a valid address.

Response behaviour

The endpoint always returns HTTP 200. The `message` and `address` fields indicate the outcome:

Scenario	`message`	`address`	`records`
Address found, records exist	`Ok`	Populated with the resolved address	Array of real estate records
Address found, no records	`Ok`	Populated with the resolved address	Empty array
Address not found	`Address not found`	Empty object	Empty array

An address may not be found if the GNAF ID does not exist, or if the text-based address could not be resolved to a valid address in the GNAF database.

Address input modes

Only one address input mode can be used per request. The modes are mutually exclusive:

Mode	Parameters	Description
GNAF ID	`gnaf_id`	Direct lookup by GNAF address ID. This is the most efficient lookup.
Full address	`full_address`	A complete address as a single string (e.g. "20 Hardy St Lilydale VIC 3140"). The system will parse and resolve the address.
Structured address	`street_address`, `suburb`, `state`, `postcode`	Address components provided separately. At minimum, `street_address` is required.

Sandbox environment

When simulating queries in the sandbox environment the system will use a reduced sample dataset. The following GNAF IDs will return real estate records:

GNAF ID	Address	Suburb	State	Postcode	Records
GAVIC421647320	20 HARDY ST	LILYDALE	VIC	3140	3
GANSW716615055	35 YARALLA ST	CONCORD WEST	NSW	2138	2
GANSW712900961	22 BRABYN ST	WINDSOR	NSW	2756	1

The following GNAF IDs exist in the sample dataset but have no real estate records:

GNAF ID	Address	Suburb	State	Postcode
GAVIC419608268	8 WALTHAM ST	RICHMOND	VIC	3121
GANSW705730292	6 WATERVIEW CL	PORT MACQUARIE	NSW	2444

Request body

Field	Description
`gnaf_id`	**string [max 15 characters]** The GNAF address ID. Mutually exclusive with full_address and street_address/suburb/state/postcode. **Example:** `GAVIC421647320`
`full_address`	**string [4..255 characters]** A complete Australian address as a single string. The system will parse and resolve this to a GNAF address. Mutually exclusive with gnaf_id and street_address/suburb/state/postcode. **Example:** `20 Hardy St Lilydale VIC 3140`
`street_address`	**string [4..100 characters]** The street address component (e.g. "20 Hardy St"). Required when using structured address mode. Mutually exclusive with gnaf_id and full_address. **Example:** `20 Hardy St`
`suburb`	**string [3..60 characters]** The suburb name. Used with street_address for structured address mode. **Example:** `Lilydale`
`state`	**string** The Australian state or territory abbreviation. Used with street_address for structured address mode. **Enum:** `ACT`, `NSW`, `NT`, `QLD`, `SA`, `TAS`, `VIC`, `WA` **Example:** `VIC`
`postcode`	**string [3..4 characters]** The 3 or 4 digit Australian postcode. Used with street_address for structured address mode. **Pattern:** `^\d{3,4}$` **Example:** `3140`

Field	Description
`date_from`	**string (date)** Optional date filter in YYYY-MM-DD format. When provided, only real estate records with a date on or after this value will be returned. **Example:** `2015-01-01`

Sample request

```
{
    "gnaf_id": "GAVIC421647320",
    "full_address": "20 Hardy St Lilydale VIC 3140",
    "street_address": "20 Hardy St",
    "suburb": "Lilydale",
    "state": "VIC",
    "postcode": "3140",
    "date_from": "2015-01-01"
}
```

Responses

200 **Result of real estate search** (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 **Example:** `738d9a89-b6fe-4fc1-96be-1389b2a506a2`
`total_records`	**integer** Number of real estate records in the result set **Example:** `3`
`address`	**object** The resolved address details. Empty object if the address could not be found.
`address.` `address_id`	**string** The GNAF address ID **Example:** `GAVIC421647320`

Field	Description
address. address	**string** The combined street address **Example:** `20 HARDY ST`
address. suburb	**string** The suburb **Example:** `LILYDALE`
address. state	**string** The state **Example:** `VIC`
address. postcode	**string** The postcode **Example:** `3140`
records	**array of objects** Real estate listing records for this address, sorted by date descending (most recent first)
records[]. listing_type	**string** The type of real estate listing: - `sale` - Property listed for sale - `sold` - Property has been sold - `rent` - Property listed for rent **Enum:** `sale`, `sold`, `rent` **Example:** `sale`
records[]. property_type	**string** The type of property (e.g. house, townhouse, apartment, unit, flat, studio) **Example:** `townhouse`
records[]. num_bedrooms	**integer** Number of bedrooms **Example:** `3`

Field	Description
records[]. num_bathrooms	**integer** Number of bathrooms **Example:** `3`
records[]. num_car_spaces	**integer** Number of car spaces **Example:** `2`
records[]. date	**string (date)** The date of the listing or sale (YYYY-MM-DD) **Example:** `2018-03-01`
records[]. estate_agent	**string** The name of the estate agent **Example:** `ABC Property`
records[]. price	**string** The listed or sold price. For rentals this will include the rental period (e.g. "$550 per week"). **Example:** `$720,000`

Sample response


```
{
    "message": "Ok",
    "api_reference": "738d9a89-b6fe-4fc1-96be-1389b2a506a2",
    "total_records": 3,
    "address": {
        "address_id": "GAVIC421647320",
        "address": "20 HARDY ST",
        "suburb": "LILYDALE",
        "state": "VIC",
        "postcode": "3140"
    },
    "records": [
        {
            "listing_type": "sale",
            "property_type": "townhouse",
            "num_bedrooms": 3,
            "num_bathrooms": 3,
            "num_car_spaces": 2,
            "date": "2018-03-01",
            "estate_agent": "ABC Property",
            "price": "$720,000"
        }
```

```
    ]
}
```

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

Real Estate Search (Bulk)

POST `/realestate_searches`

This is a bulk version of the real estate search endpoint that allows multiple address lookups to be performed in a single request.

This endpoint accepts up to 50 address lookups in the `requests` array and applies a shared optional `date_from` filter to each item.

Each item in the `requests` array accepts an address in one of three formats:

- A **GNAF ID** (the unique GNAF address identifier)
- A **full address** as a single string
- A **structured address** with street_address, suburb, state, and postcode as separate fields

Only one address input mode can be used per request item. The modes are mutually exclusive within each item.

Results for each item are returned in **reverse chronological order** (most recent first).

Address resolution

The `address` object in each result item represents the resolved address that was used to search the real estate universe:

- When a **GNAF ID** is provided, the returned address is the exact address referenced by that GNAF ID.
- When a **text-based address** is provided (full address or structured address), the system will resolve the input to the best matching address in the GNAF database. The returned address may include minor corrections or standardisations compared to the original input (e.g. corrected spelling, expanded abbreviations, standardised formatting) if these were needed to match a valid address.

Response behaviour

The endpoint always returns HTTP 200. Each result item contains its own `message` and `address` fields indicating the outcome for that particular address lookup:

Scenario	`message`	`address`	`records`
Address found, records exist	`Ok`	Populated with the resolved address	Array of real estate records
Address found, no records	`Ok`	Populated with the resolved address	Empty array
Address not found	`Address not found`	Empty object	Empty array

An address may not be found if the GNAF ID does not exist, or if the text-based address could not be resolved to a valid address in the GNAF database.

Sandbox environment

When simulating queries in the sandbox environment the system will use a reduced sample dataset. The following GNAF IDs will return real estate records:

GNAF ID	Address	Suburb	State	Postcode	Records
GAVIC421647320	20 HARDY ST	LILYDALE	VIC	3140	3
GANSW716615055	35 YARALLA ST	CONCORD WEST	NSW	2138	2
GANSW712900961	22 BRABYN ST	WINDSOR	NSW	2756	1

The following GNAF IDs exist in the sample dataset but have no real estate records:

GNAF ID	Address	Suburb	State	Postcode
GAVIC419608268	8 WALTHAM ST	RICHMOND	VIC	3121
GANSW705730292	6 WATERVIEW CL	PORT MACQUARIE	NSW	2444

Request body

Field	Description
`date_from`	**string (date)** Optional date filter in YYYY-MM-DD format. When provided, only real estate records with a date on or after this value will be returned. This filter applies to all items in the requests array. **Example:** `2015-01-01`
`requests`	**array of objects [1..50 items]** The list of address lookups to perform.
`requests[].` `check_id`	**string** Optional client reference returned in the matching result item. **Example:** `property_001`
`requests[].` `gnaf_id`	**string [max 15 characters]** The GNAF address ID. Mutually exclusive with full_address and street_address/suburb/state/postcode. **Example:** `GAVIC421647320`

Field	Description
requests[]. full_address	**string [4..255 characters]** A complete Australian address as a single string. The system will parse and resolve this to a GNAF address. Mutually exclusive with gnaf_id and street_address/suburb/state/postcode. **Example:** `20 Hardy St Lilydale VIC 3140`
requests[]. street_address	**string [4..100 characters]** The street address component (e.g. "20 Hardy St"). Required when using structured address mode. Mutually exclusive with gnaf_id and full_address. **Example:** `20 Hardy St`
requests[]. suburb	**string [3..60 characters]** The suburb name. Used with street_address for structured address mode. **Example:** `Lilydale`
requests[]. state	**string** The Australian state or territory abbreviation. Used with street_address for structured address mode. **Enum:** `ACT`, `NSW`, `NT`, `QLD`, `SA`, `TAS`, `VIC`, `WA` **Example:** `VIC`
requests[]. postcode	**string [3..4 characters]** The 3 or 4 digit Australian postcode. Used with street_address for structured address mode. **Pattern:** `^\d{3,4}$` **Example:** `3140`

Sample request

```
{
    "date_from": "2015-01-01",
    "requests": [
        {
            "check_id": "property_001",
            "gnaf_id": "GAVIC421647320",
            "full_address": "20 Hardy St Lilydale VIC 3140",
            "street_address": "20 Hardy St",
            "suburb": "Lilydale",
            "state": "VIC",
            "postcode": "3140"
        }
    ]
}
```

Responses

200 **Result of bulk real estate search** (application/json)

Field	Description
`message`	**string** A message indicating the overall result of the request. **Example:** `Ok`
`api_reference`	**string (uuid)** A unique identifier for this request **Example:** `738d9a89-b6fe-4fc1-96be-1389b2a506a2`
`results`	**array of objects** Result item for each request in the requests array.
`results[].` `check_id`	**string or null** Echoed check_id from the request item. **Example:** `property_001`
`results[].` `message`	**string** A message indicating the result for this item. `Ok` if the address was found, `Address not found` otherwise. **Example:** `Ok`
`results[].` `total_records`	**integer** Number of real estate records in this item's result set. **Example:** `3`
`results[].` `address`	**object** The resolved address details. Empty object if the address could not be found.

`results[].`
`address.`
`address_id`
	string
	The GNAF address ID
	Example: `GAVIC421647320`

`results[].`
`address.`
`address`
	string
	The combined street address
	Example: `20 HARDY ST`

Field	Description
results[]. address. suburb	**string** The suburb **Example:** `LILYDALE`
results[]. address. state	**string** The state **Example:** `VIC`
results[]. address. postcode	**string** The postcode **Example:** `3140`
results[]. records	**array of objects** Real estate listing records for this address, sorted by date descending (most recent first)
results[]. records[]. listing_type	**string** The type of real estate listing: - `sale` - Property listed for sale - `sold` - Property has been sold - `rent` - Property listed for rent **Enum:** `sale`, `sold`, `rent` **Example:** `sale`
results[]. records[]. property_type	**string** The type of property (e.g. house, townhouse, apartment, unit, flat, studio) **Example:** `townhouse`
results[]. records[]. num_bedrooms	**integer** Number of bedrooms **Example:** `3`
results[]. records[]. num_bathrooms	**integer** Number of bathrooms **Example:** `3`

Field	Description
results[]. records[]. num_car_spaces	**integer** Number of car spaces **Example:** `2`
results[]. records[]. date	**string (date)** The date of the listing or sale (YYYY-MM-DD) **Example:** `2018-03-01`
results[]. records[]. estate_agent	**string** The name of the estate agent **Example:** `ABC Property`
results[]. records[]. price	**string** The listed or sold price. For rentals this will include the rental period (e.g. "$550 per week"). **Example:** `$720,000`
results[]. error	**string** Present when an item-level error occurs.

Sample response


```
{
    "message": "Ok",
    "api_reference": "738d9a89-b6fe-4fc1-96be-1389b2a506a2",
    "results": [
        {
            "check_id": "property_001",
            "message": "Ok",
            "total_records": 3,
            "address": {
                "address_id": "GAVIC421647320",
                "address": "20 HARDY ST",
                "suburb": "LILYDALE",
                "state": "VIC",
                "postcode": "3140"
            },
            "records": [
                {
                    "listing_type": "sale",
                    "property_type": "townhouse",
                    "num_bedrooms": 3,
                    "num_bathrooms": 3,
                    "num_car_spaces": 2,
                    "date": "2018-03-01",
                    "estate_agent": "ABC Property",
                    "price": "$720,000"
```

```
                }
            ],
            "error": "string"
        }
    ]
}
```

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

Adverse Media Check

POST `/adverse_media_check`

Experimental

The Adverse Media API is still in the experimental stage and is subject to change as we continue to refine the API.

The adverse_media_check method will perform a search for adverse media for a person or business entity.

The check will return a summary of the information found as well as a set of links and categories that the subject is involved in.

The categories are:

Category	Description
financial_crime	Fraud, bribery, money laundering, or embezzlement.
violent_crime	Assault, murder, armed robbery, or other violent acts.
organised_crime	Gang activity, drug trafficking, human trafficking, or cartels.
drug_offences	Drug trafficking, possession, or other drug-related offences.
sexual_offences	Sexual assault, exploitation, or possession of illegal material.
regulatory_action	Disqualifications, fines, bans, or other legal enforcement by regulators.
terrorism	Extremist activity, terrorism-related charges, or national security threats.
scams_and_fraud	Ponzi schemes, pyramid schemes, or public scam warnings.
environmental_violations	Pollution, environmental damage, or EPA-related prosecutions.
employment_violations	Workplace safety breaches, wage theft, or industrial violations.
corruption_and_abuse_of_office	Political or corporate corruption, including bribery, nepotism, or misuse of power or funds.

Sandbox environment

When simulating queries in the sandbox environment, the adverse media check response will return a match for the following details:

People

First Name	Middle Name	Last Name	Birth Date	Country
John	James	Doe	1990-01-01	AU

First Name	Middle Name	Last Name	Birth Date	Country
Jane	Karen	Smith	1995-02-18	AU
Liam	Raphael	O'Connor	1980-02-01	AU
Olivia	Rose	Smith	1973-06-06	UK

Businesses

Business Name	Country
Acme Systems	AU
Sample Software Services	NZ

Request body

The details of the person or business entity to check

Field	Description
`type`	**string** The type of entity to check **Enum:** `person`, `business` **Example:** `person`
`first_name`	**string** The first name of the individual (Required for person checks) **Example:** `John`
`middle_name`	**string or null** The middle name of the individual (Nullable for person checks) **Example:** `James`
`last_name`	**string** The last name of the individual (Required for person checks) **Example:** `Doe`
`birth_date`	**string (date) or null** The date of birth of the individual Format: YYYY-MM-DD (Nullable for business checks) The date of birth is used to filter results if provided and if the adverse media has an age or age range. **Example:** `1990-01-01`

Field	Description
`business_name`	**string** The name of the business entity (Required for business checks) **Example:** `Acme Pty Ltd`
`country`	**string or null** The ISO 3166 country code of the person or business entity (optional). When provided, the country is used as a hint to prefer results from that country. It can be omitted. **Example:** `AU`

Sample request

```
{
    "type": "person",
    "first_name": "John",
    "middle_name": "James",
    "last_name": "Doe",
    "birth_date": "1990-01-01",
    "business_name": "Acme Pty Ltd",
    "country": "AU"
}
```

Responses

200 **Adverse media check result.** (application/json)

Field	Description
`message`	**string** Always `Ok` on success. **Example:** `Ok`
`function`	**string** The API function that handled the request. **Example:** `adverse_media`
`api_reference`	**string (uuid)** Audit reference for this adverse media check request. **Example:** `16af1201-c34c-4525-9ae5-9d43e2ca469f`
`match`	**object** The result of the adverse media check

Field	Description
match. name	**string** The name of the person or business entity **Example:** `John Doe`
match. age	**number or null** The age of the person (null for business entities or where unavailable)
match. location	**string or null** The location of the entity if available **Example:** `Sydney, NSW, Australia`
match. summary	**string** A summary of the adverse media found **Example:** `John Doe was charged with fraud.`
match. links	**array of strings** A list of links to the news articles used to create the summary
match. categories	**array of strings** A list of categories that the entity is involved in

Sample response


```
{
    "message": "Ok",
    "function": "adverse_media",
    "api_reference": "16af1201-c34c-4525-9ae5-9d43e2ca469f",
    "match": {
        "name": "John Doe",
        "age": null,
        "location": "Sydney, NSW, Australia",
        "summary": "John Doe was charged with fraud.",
        "links": [
            "https://example.com/news/adverse-media"
        ],
        "categories": [
            "financial_crime"
        ]
    }
}
```


Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

ASIC Banned Persons Search

POST `/asic_banned_persons`

ASIC Banned Persons Search searches the ASIC register of banned persons.

This endpoint allows multiple requests to be simultaneously made in a single query.

Narrow Search

A narrow search will match exact first name, exact last name. The middle name is matched based on the similarity_threshold. This is the most restrictive search and should be used for low risk searches.

Medium Search

A medium search will match exact last name, and will allow for a partial match on other names based on the similarity_threshold. This is a good balance between accuracy and flexibility.

Broad Search

A broad search will match partial first name, partial middle name, and partial last name. This is the most flexible search and should be used for high risk searches.

Sandbox environment data

When simulating queries in the sandbox environment, the following records will return a match. Any other name returns an empty `matches` array.

First name	Middle name	Last name	Banned type	Suburb	State
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	-	-
Adam	Darren	Halliday	disqualified_director	Illawong	NSW

Request body

The details of the individuals to search for.

Field	Description
`search_type`	**string** The type of search to perform. Narrow searches for exact matches, medium searches for similar matches and broad searches for partial matches. **Enum:** `narrow_search`, `medium_search`, `broad_search` **Example:** `medium_search`
`banned_types`	**array of strings** The type of banned person to search for. Leave empty to search for all types. **Enum:** `afs_banned_disqualified`, `banned_futures`, `banned_securities`, `credit_banned_disqualified`, `disqualified_director`
`similarity_threshold`	**number [0..100]** The minimum similarity percentage to return results for. **Default:** `80` **Example:** `80`
`requests`	**array of objects** The list of requests to make.
`requests[].` `check_id`	**string** The unique identifier for this request. This can be used match the request in the results. **Example:** `abcd123456789`
`requests[].` `first_name`	**string [1..50 characters]** The first name of the individual **Example:** `Phil`
`requests[].` `middle_name`	**string [1..50 characters] or null** The middle name of the individual
`requests[].` `last_name`	**string [1..50 characters]** The last name of the individual **Example:** `Smith`

Sample request

```
{
    "search_type": "medium_search",
    "banned_types": [
        "afs_banned_disqualified"
    ],
    "similarity_threshold": 80,
    "requests": [
        {
            "check_id": "abcd123456789",
            "first_name": "Phil",
            "middle_name": null,
            "last_name": "Smith"
        }
    ]
}
```

Responses

200 **Successful 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`
`function`	**string** The function that was called. **Example:** `asic_banned_persons`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`results`	**array of objects** A list of results for each of the requests made.
`results[].` `check_id`	**string** The unique identifier for this request. This can be used match the request in the results. **Example:** `abcd123456789`
`results[].` `matches`	**array of objects** Array of matched individuals.

Field	Description
results[]. matches[]. first_name	**string** The first name of the individual. **Example:** `Phil`
results[]. matches[]. middle_name	**string or null** The middle name of the individual. **Example:** `Michael`
results[]. matches[]. last_name	**string** The last name of the individual. **Example:** `Smith`
results[]. matches[]. banned_type	**string** The type of banned person the individual is. **Example:** `afs_banned_disqualified`
results[]. matches[]. document_number	**string** ASIC's internal document number used to identify the document containing the ban or disqualification notice/order. **Example:** `1234567890`
results[]. matches[]. start_date	**string** The start date of the ban or disqualification. **Example:** `2021-01-01`
results[]. matches[]. end_date	**string or null** The end date of the ban or disqualification. **Example:** `2021-01-01`
results[]. matches[]. address_suburb	**string or null** The suburb of the individual. **Example:** `Sydney`
results[]. matches[]. address_state	**string or null** The state of the individual. **Example:** `NSW`

Field	Description
results[]. matches[]. address_postcode	**string or null** The postcode of the individual. **Example:** `2000`
results[]. matches[]. address_country	**string or null** The country of the individual. **Example:** `Australia`
results[]. matches[]. comments	**string or null** Additional information associated with the banned or disqualified person. **Example:** `This is a comment`
results[]. matches[]. name_similarity	**number** The similarity of the name of the individual to the search name. **Example:** `100`

Sample response


```
{
    "message": "Ok",
    "function": "asic_banned_persons",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "results": [
        {
            "check_id": "abcd123456789",
            "matches": [
                {
                    "first_name": "Phil",
                    "middle_name": "Michael",
                    "last_name": "Smith",
                    "banned_type": "afs_banned_disqualified",
                    "document_number": "1234567890",
                    "start_date": "2021-01-01",
                    "end_date": "2021-01-01",
                    "address_suburb": "Sydney",
                    "address_state": "NSW",
                    "address_postcode": "2000",
                    "address_country": "Australia",
                    "comments": "This is a comment",
                    "name_similarity": 100
                }
            ]
        }
    ]
}
```


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

ASIC Search

POST `/asic_search`

ASIC Search searches the ASIC register for the given company or business name and returns the results.

Company Search

A company can be searched by either a number (ABN / ACN / State Number)

Type	Format	Example	Description
ABN	11 digits	12345678901	Australian Business Number
ACN	9 digits	123456789	Australian Company Number
State Number	Jurisdiction:number	NSW:1234567	State business registration number

Business Name Search

A business name can be searched by providing the exact name of the business,

Request body

The details of the business to search for.

Field	Description
`number`	**string** The number of the company to search for (ABN / ACN / State Number) **Example:** `12345678901`
`business_name`	**string** The exact name of the business to search for. **Example:** `Acme Services`

Sample request

```
{
    "number": "12345678901",
    "business_name": "Acme Services"
}
```

Responses

200 **Successful 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. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`match`	**object** Details of the match found.
`match.` `identifier`	**object** The number of the business.

`match.`
`identifier.`
`numberHeading`
	string
	
	The heading of the number.
	
	Example: `ABN`

`match.`
`identifier.`
`number`
	string
	
	The number of the business.
	
	Example: `12345678901`

`match.` `jurisdiction`	**string** The jurisdiction of the business. **Example:** `Australian Securities & Investments Commission`
`match.` `name`	**object** The name of the business.

`match.`
`name.`
`name`
	string
	
	The name of the business.
	
	Example: `Acme Services`

`match.` `type`	**object** The type of business.

match. type. code	**string** The code of the business type. **Example:** `BUSN`
match. type. description	**string** The description of the business type. **Example:** `Business Names`
match. class	**object** The class of business (available for companies only).
match. class. code	**string** The code of the business class. **Example:** `LMSH`
match. class. description	**string** The description of the business class. **Example:** `Limited By Shares`
match. subClass	**object** The subclass of business (available for companies only).
match. subClass. code	**string** The code of the business subclass. **Example:** `PROP`
match. subClass. description	**string** The description of the business subclass. **Example:** `Proprietary Company`
match. status	**object** The status of business.
match. status. code	**string** The code of the business status. **Example:** `REGD`

match. status. description	**string** The description of the business status. **Example:** `Registered`
match. status. isRegistered	**boolean** Whether the business is registered. **Example:** `true`
match. abrEntity	**object** Whether the business is an ABR entity.
match. abrEntity. abn	**string** The ABN of the ABR entity. **Example:** `12345678901`
match. abrEntity. entityName	**string** The name of the ABR entity. **Example:** `Acme Services`
match. abrEntity. entityType	**string** The type of ABR entity. **Example:** `Company`
match. abrEntity. effectiveDate	**string** The effective date of the ABR entity. **Example:** `2021-01-01`
match. dateReview	**string** The date of the next review of the ABR entity. **Example:** `2024-01-01`
match. dateRegistered	**string** The date the business was registered. **Example:** `2021-01-01`
match. address	**array of objects** The address of the business.

match. address[]. type	**string** The type of address. (RG = registered office, PA = principal place of business) **Example:** `RG`
match. address[]. state	**string** The state of the address. **Example:** `NSW`
match. address[]. postcode	**string** The postcode of the address. **Example:** `2000`
match. address[]. locality	**string** The locality / suburb of the address. **Example:** `Sydney`
match. address[]. addressLine	**string** The first line of the address. (This is RESTRICTED in most cases) **Example:** `RESTRICTED`
match. address[]. iso3166CountryCode	**string** The ISO 3166 country code of the address. **Example:** `AU`
match. incorporationState	**string** The state of the incorporation. **Example:** `NSW`
match. recentDocument	**array of objects** The most recent documents for the company
match. recentDocument[]. formCode	**string** The form code of the document. **Example:** `484`
match. recentDocument[]. description	**string** The description of the document. **Example:** `CHANGE TO COMPANY DETAILS`

match. recentDocument[]. dateReceived	**string** The date of the document. **Example:** `2021-01-01`
match. recentDocument[]. numberOfPages	**number** The number of pages in the document. **Example:** `1`
match. recentDocument[]. documentNumber	**string** The document number. **Example:** `ABC12345678`
match. recentDocument[]. additionalDescription	**array of objects** Additional description of the document.
match. recentDocument[]. additionalDescription[]. subformCode	**string** The subform code of the additional description. **Example:** `484N`
match. recentDocument[]. additionalDescription[]. subformDescription	**string** CHANGES TO (MEMBERS) SHARE HOLDINGS. **Example:** `Change of name`

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "match": {
        "identifier": {
            "numberHeading": "ABN",
            "number": "12345678901"
        },
        "jurisdiction": "Australian Securities & Investments Commission",
        "name": {
            "name": "Acme Services"
        },
        "type": {
            "code": "BUSN",
            "description": "Business Names"
        },
        "class": {
            "code": "LMSH",
            "description": "Limited By Shares"
        },
        "subClass": {
```

```
            "code": "PROP",
            "description": "Proprietary Company"
        },
        "status": {
            "code": "REGD",
            "description": "Registered",
            "isRegistered": true
        },
        "abrEntity": {
            "abn": "12345678901",
            "entityName": "Acme Services",
            "entityType": "Company",
            "effectiveDate": "2021-01-01"
        },
        "dateReview": "2024-01-01",
        "dateRegistered": "2021-01-01",
        "address": [
            {
                "type": "RG",
                "state": "NSW",
                "postcode": "2000",
                "locality": "Sydney",
                "addressLine": "RESTRICTED",
                "iso3166CountryCode": "AU"
            }
        ],
        "incorporationState": "NSW",
        "recentDocument": [
            {
                "formCode": "484",
                "description": "CHANGE TO COMPANY DETAILS",
                "dateReceived": "2021-01-01",
                "numberOfPages": 1,
                "documentNumber": "ABC12345678",
                "additionalDescription": [
                    {
                        "subformCode": "484N",
                        "subformDescription": "Change of name"
                    }
                ]
            }
        ]
    }
}
```


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

ASIC Search Bulk

POST `/asic_searches`

ASIC Search Bulk searches the ASIC register for the given companies or business name and returns the results.

Company Search

A company can be searched by either a number (ABN / ACN / State Number)

Type	Format	Example	Description
ABN	11 digits	12345678901	Australian Business Number
ACN	9 digits	123456789	Australian Company Number
State Number	Jurisdiction:number	NSW:1234567	State business registration number

Business Name Search

A business name can be searched by providing the exact name of the business,

Request body

The details of the business to search for.

Field	Description
`requests`	**array of objects** The requests to search for.
`requests[].` `check_id`	**string** The unique identifier for this request. This can be used match the request in the results. **Example:** `abcd123456789`
`requests[].` `number`	**string** The number of the company to search for (ABN / ACN / State Number) **Example:** `12345678901`
`requests[].` `business_name`	**string** The exact name of the business to search for. **Example:** `Acme Services`

Sample request

```
{
    "requests": [
        {
            "check_id": "abcd123456789",
            "number": "12345678901",
            "business_name": "Acme Services"
        }
    ]
}
```

Responses

200 **Successful 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. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`results`	**array of objects** A list of results for each of the requests made.
`results[].` `check_id`	**string** The unique identifier for this request. This can be used match the request in the results. **Example:** `abcd123456789`
`results[].` `match`	**object** Details of the match found or null if no match was found.

`results[].`
`match.`
`identifier`
	object
	
	The number of the business.

`results[].`
`match.`
`identifier.`
`numberHeading`
	string
	
	The heading of the number.
	
	Example: `ABN`

`results[].`
`match.`
`identifier.`
`number`
	string
	
	The number of the business.
	
	Example: `12345678901`

`results[].`
`match.`
`jurisdiction`
	string
	
	The jurisdiction of the business.
	
	Example: `Australian Securities & Investments Commission`

`results[].`
`match.`
`name`
	object
	
	The name of the business.

Field	Description
results[]. match. name. name	**string** The name of the business. **Example:** `Acme Services`
results[]. match. type	**object** The type of business.
results[]. match. type. code	**string** The code of the business type. **Example:** `BUSN`
results[]. match. type. description	**string** The description of the business type. **Example:** `Business Names`
results[]. match. class	**object** The class of business (available for companies only).
results[]. match. class. code	**string** The code of the business class. **Example:** `LMSH`
results[]. match. class. description	**string** The description of the business class. **Example:** `Limited By Shares`
results[]. match. subClass	**object** The subclass of business (available for companies only).
results[]. match. subClass. code	**string** The code of the business subclass. **Example:** `PROP`
results[]. match. subClass. description	**string** The description of the business subclass. **Example:** `Proprietary Company`

Field	Description
results[]. match. status	**object** The status of business.
results[]. match. status. code	**string** The code of the business status. **Example:** `REGD`
results[]. match. status. description	**string** The description of the business status. **Example:** `Registered`
results[]. match. status. isRegistered	**boolean** Whether the business is registered. **Example:** `true`
results[]. match. abrEntity	**object** Whether the business is an ABR entity.
results[]. match. abrEntity. abn	**string** The ABN of the ABR entity. **Example:** `12345678901`
results[]. match. abrEntity. entityName	**string** The name of the ABR entity. **Example:** `Acme Services`
results[]. match. abrEntity. entityType	**string** The type of ABR entity. **Example:** `Company`
results[]. match. abrEntity. effectiveDate	**string** The effective date of the ABR entity. **Example:** `2021-01-01`
results[]. match. dateReview	**string** The date of the next review of the ABR entity. **Example:** `2024-01-01`

Field	Description
results[]. match. dateRegistered	**string** The date the business was registered. **Example:** `2021-01-01`
results[]. match. address	**array of objects** The address of the business.
results[]. match. address[]. type	**string** The type of address. (RG = registered office, PA = principal place of business) **Example:** `RG`
results[]. match. address[]. state	**string** The state of the address. **Example:** `NSW`
results[]. match. address[]. postcode	**string** The postcode of the address. **Example:** `2000`
results[]. match. address[]. locality	**string** The locality / suburb of the address. **Example:** `Sydney`
results[]. match. address[]. addressLine	**string** The first line of the address. (This is RESTRICTED in most cases) **Example:** `RESTRICTED`
results[]. match. address[]. iso3166CountryCode	**string** The ISO 3166 country code of the address. **Example:** `AU`
results[]. match. incorporationState	**string** The state of the incorporation. **Example:** `NSW`
results[]. match. recentDocument	**array of objects** The most recent documents for the company

Field	Description
results[]. match. recentDocument[]. formCode	**string** The form code of the document. **Example:** `484`
results[]. match. recentDocument[]. description	**string** The description of the document. **Example:** `CHANGE TO COMPANY DETAILS`
results[]. match. recentDocument[]. dateReceived	**string** The date of the document. **Example:** `2021-01-01`
results[]. match. recentDocument[]. numberOfPages	**number** The number of pages in the document. **Example:** `1`
results[]. match. recentDocument[]. documentNumber	**string** The document number. **Example:** `ABC12345678`
results[]. match. recentDocument[]. additionalDescription	**array of objects** Additional description of the document.
results[]. match. recentDocument[]. additionalDescription[]. subformCode	**string** The subform code of the additional description. **Example:** `484N`
results[]. match. recentDocument[]. additionalDescription[]. subformDescription	**string** CHANGES TO (MEMBERS) SHARE HOLDINGS. **Example:** `Change of name`

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "results": [
        {
            "check_id": "abcd123456789",
            "match": {
```

```
                "identifier": {
                    "numberHeading": "ABN",
                    "number": "12345678901"
                },
                "jurisdiction": "Australian Securities & Investments Commission",
                "name": {
                    "name": "Acme Services"
                },
                "type": {
                    "code": "BUSN",
                    "description": "Business Names"
                },
                "class": {
                    "code": "LMSH",
                    "description": "Limited By Shares"
                },
                "subClass": {
                    "code": "PROP",
                    "description": "Proprietary Company"
                },
                "status": {
                    "code": "REGD",
                    "description": "Registered",
                    "isRegistered": true
                },
                "abrEntity": {
                    "abn": "12345678901",
                    "entityName": "Acme Services",
                    "entityType": "Company",
                    "effectiveDate": "2021-01-01"
                },
                "dateReview": "2024-01-01",
                "dateRegistered": "2021-01-01",
                "address": [
                    {
                        "type": "RG",
                        "state": "NSW",
                        "postcode": "2000",
                        "locality": "Sydney",
                        "addressLine": "RESTRICTED",
                        "iso3166CountryCode": "AU"
                    }
                ],
                "incorporationState": "NSW",
                "recentDocument": [
                    {
                        "formCode": "484",
                        "description": "CHANGE TO COMPANY DETAILS",
                        "dateReceived": "2021-01-01",
                        "numberOfPages": 1,
                        "documentNumber": "ABC12345678",
                        "additionalDescription": []
                    }
                ]
            }
        }
    ]
}
```


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

ASIC Extract Person Search

POST `/asic_extract_person_search`

Runs an ASIC person search to locate individual ASIC person IDs that can be used when requesting a full ASIC extract. Provide the subject's legal name, date of birth, and whether you want **current** or **historical** data. The API returns a list of matching ASIC identities (`matches`) including the `person_id` and `search_id` you must supply to `/asic_extract` when ordering the actual extract.

Each result can also include an `additional` array. These entries represent linked ASIC person IDs that have either the same name or the same name and birth date (for example, with shareholdings). When requesting a person extract you should copy the `person_id` values from `additional` into the `additional_person_ids` field so the resulting extract contains every related person.

Input expectations

- Names must be provided exactly as they appear on ASIC records. A combined maximum of three given names is allowed across `first_name` and `middle_name` .
- `birth_date` must be an ISO `YYYY-MM-DD` date after 1900-01-01.
- `extract_type` selects the extract type that the search results will be used for. This must be provided before ordering an extract as it will affect the search id result returned. Choose one of the following:
  - `current` – returns people with current roles/shareholdings.
  - `historical` – includes ceased roles/shareholdings.

Sandbox environment

The sandbox dataset is deterministic. Use one of the following combinations to receive sample matches:

First Name	Middle Name	Last Name	Birth Date	Extract Type	Notes
John	Michael	Doe	1980-05-15	historical	Standard three-part name with linked extract.
Jane	Alice	Smith	1987-03-22	current	Standard three-part name with linked extract.
Lucas	Benjamin	Carter	1992-08-09	current	Standard three-part name with linked extract.
Maria		Scott	1971-11-02	historical	No middle name on the ASIC record.
Patrick		O'Connor	1965-07-19	historical	Apostrophe in the surname (also matches `OConnor`).
Mary-Jane		Parker	1983-02-28	historical	Hyphenated first name.

Request body

Details of the individual to search for.

Field	Description

`first_name` REQUIRED	**string [max 100 characters]** The person's first given name (one to three given names when combined with `middle_name`). **Example:** `John`
`middle_name`	**string [max 100 characters] or null** Optional middle names. Combined with `first_name` this cannot exceed three words. **Example:** `Michael`
`last_name` REQUIRED	**string [max 100 characters]** The person's family name as recorded with ASIC. **Example:** `Doe`
`birth_date` REQUIRED	**string (date)** The person's date of birth in `YYYY-MM-DD` format (must be after 1900-01-01). **Example:** `1980-05-15`
`extract_type` REQUIRED	**string** Indicates whether to search current or historical ASIC data. Historical searches include ceased roles and can take longer to return. **Enum:** `current`, `historical` **Example:** `historical`

Sample request

```
{
    "first_name": "John",
    "middle_name": "Michael",
    "last_name": "Doe",
    "birth_date": "1980-05-15",
    "extract_type": "historical"
}
```

Responses

200 **Search completed successfully.** (application/json)

Field	Description

`message`	**string** Always `Ok` when the request succeeds. **Example:** `Ok`
`function`	**string** The API function that handled the request. **Example:** `asic_extract_person_search`
`api_reference`	**string (uuid)** Audit reference you can use with `/asic_extract/{api_reference}` to retrieve logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`matches`	**array of objects** A list of ASIC identities that match the supplied name and birth date.
`matches[].` `person_id`	**string** ASIC person identifier to pass as `person_id` when ordering an extract. **Example:** `123456780`
`matches[].` `search_id`	**string** ASIC search reference identifier to pass as `search_id` when ordering an extract. **Example:** `11223344`
`matches[].` `name`	**string** The full formatted name returned by ASIC (usually uppercase). **Example:** `JOHN MICHAEL DOE`
`matches[].` `birth_date`	**string (date)** Date of birth for the matched person. Some upstream providers may label this field `dob`. **Example:** `1980-05-15`
`matches[].` `state`	**string** State or territory associated with the person's ASIC address. **Example:** `NSW`

matches[]. suburb	**string** Suburb associated with the person's ASIC address. **Example:** `Sydney`
matches[]. additional	**array of objects** Additional ASIC person identifiers linked to this match (for example joint shareholdings). Supply each `person_id` in `additional_person_ids` when calling `/asic_extract`.
matches[]. additional[]. person_id	**string** Linked ASIC person identifier. **Example:** `123456781`
matches[]. additional[]. name	**string** Full name of the linked individual (if provided). **Example:** `JOHN MICHAEL DOE`
matches[]. additional[]. state	**string or null** State or territory associated with the linked record.
matches[]. additional[]. suburb	**string or null** Suburb associated with the linked record.
matches[]. additional[]. birth_date	**string (date) or null** Date of birth for the linked individual.
matches[]. additional[]. additional	**array of objects** Nested linked identities (rare). Typically empty.

Sample response


```
{
    "message": "Ok",
    "function": "asic_extract_person_search",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "matches": [
        {
            "person_id": "123456780",
            "search_id": "11223344",
            "name": "JOHN MICHAEL DOE",
            "birth_date": "1980-05-15",
            "state": "NSW",
            "suburb": "Sydney",
            "additional": [
                {
```

```
                    "person_id": "123456781",
                    "name": "JOHN MICHAEL DOE",
                    "state": "string",
                    "suburb": "string",
                    "birth_date": "2024-01-01",
                    "additional": []
                }
            ]
        }
    ]
}
```

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

ASIC Extract Company Search

POST `/asic_extract_company_search`

`asic_extract_company_search` locates the ASIC identifiers you must supply when ordering a company extract via `/asic_extract`. The endpoint accepts any of the following search modes and automatically determines which one to use based on the `search` value:

Search mode	Input format	Typical use
ABN search	11 digits (spacing ignored)	When you already know the company's ABN
ACN search	9 digits (spacing ignored)	When you have the ACN / NNI
Name search	Free?form string matched against ASIC extract names	When you only know the legal company name

Behaviour

- Number searches return a single match (or an empty list) containing the ASIC record retrieved.
- Name searches can return multiple organisations. Use the optional `status` filter and `max_results` to keep the result set manageable. If the ASIC upstream indicates there are more results than returned, `more_results` is set to `true`.
- Copy either the `identifier.number` (usually an ACN) or the `abn` from a match when submitting a company extract request.

Status filters

When running a name search you can restrict the response to companies in a particular ASIC registration status:

status value	Meaning
`registered`	Only active/registered companies
`deregistered`	Only deregistered companies
`all` (default)	Return both registered and deregistered companies

Status filters are ignored for ABN / ACN lookups because those queries already return a single company.

Location filters

When running a name search you can restrict the response to a postcode or state. Results will only be included if they have a registered office address in the specified location. Postcode searches will include neighbouring postcodes.

If both a postcode and state are provided, only the postcode is applied.

Location filters are ignored for ABN / ACN lookups because those queries already return a single company.

Sandbox environment

The sandbox dataset is deterministic. Use one of the following sample organisations to receive predictable matches:

Company Name	ABN	ACN	Registration Status
Tech Innovators Ltd	32111111114	111111114	Registered
Green Energy Solutions	11123456780	123456780	Deregistered
Urban Development Corp	54222222228	222222228	Registered
Creative Media Agency	33234567894	234567894	Registered

Request body

Parameters describing the company search.

Field	Description
`search` REQUIRED	**string [max 100 characters]** Company name, ABN, or ACN to look up. Letter casing is ignored for name searches; whitespace is ignored for number searches. **Example:** `ACME CORPORATION PTY LTD`
`max_results`	**integer [1..90] or null** Maximum number of matches to return for name searches (1-90). Defaults to 20. Ignored for ABN/ACN lookups because those return at most one record. **Example:** `25`
`status`	**string or null** Optional name-search filter that limits responses to registered, deregistered, or all companies. If not supplied the API searches all companies. **Enum:** `registered`, `deregistered`, `all` **Example:** `registered`
`postcode`	**string [max 4 characters] or null** Optional name-search filter that limits responses to companies with a registered office in the specified postcode or neighbouring postcodes. Ignored for ABN/ACN lookups.

Field	Description
`state`	**string or null** Optional name-search filter that limits responses to companies with a registered office in the specified state. Use the two/three-letter state code (e.g. `VIC`, `NSW`). Ignored for ABN/ACN lookups.

Sample request

```
{
    "search": "ACME CORPORATION PTY LTD",
    "max_results": 25,
    "status": "registered",
    "postcode": "string",
    "state": "string"
}
```

Responses

200 **Company search completed successfully.** (application/json)

Field	Description
`message`	**string** Always `Ok` on success. **Example:** `Ok`
`function`	**string** The API function that handled the request. **Example:** `asic_extract_company_search`
`api_reference`	**string (uuid)** Audit reference for the request. Use `/asic_extract/{api_reference}` to retrieve audit logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`matches`	**array of objects** A list of companies that match the search criteria. Each object contains the metadata you will need when calling `/asic_extract` (typically an ACN or ABN) plus additional ASIC attributes when available.
`matches[].identifier`	**object** Primary ASIC identifier (usually an ACN) returned during name searches.

Field	Description
matches[].identifier.numberHeading	**string** ASIC identifier type (`ACN` , `ABN` , etc.). **Example:** `ACN`
matches[].identifier.number	**integer** Identifier value for the company. **Example:** `123456780`
matches[].name	**object** Structured ASIC name block.
matches[].name.name	**string** Uppercase legal name of the organisation. **Example:** `ACME CORPORATION PTY LTD`
more_results	**boolean** Indicates that ASIC reported more matches than were returned (for example when `max_results` was reached). Refine your search and try again if this is `true` . **Example:** `false`

Sample response


```
{
    "message": "Ok",
    "function": "asic_extract_company_search",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "matches": [
        {
            "identifier": {
                "numberHeading": "ACN",
                "number": 123456780
            },
            "name": {
                "name": "ACME CORPORATION PTY LTD"
            },
            "status": {
                "code": "REGD,",
                "description": "Registered",
                "isRegistered": true,
                "effectiveFrom": "2020-01-15"
            }
        },
        {
            "identifier": {
                "numberHeading": "ACN",
                "number": 987654321
```

```
            },
            "name": {
                "name": "ACME CORPORATION PTY LTD"
            },
            "status": {
                "code": "DRGD,",
                "description": "Deregistered,",
                "isRegistered": false
            }
        }
    ],
    "more_results": false
}
```

Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

ASIC ID Check

POST `/asic_id_check`

Verifies a person against the ASIC officeholder and shareholder register using their name and date of birth. Returns a small `match_results` summary indicating whether the supplied details match an ASIC record.

Use this endpoint when you need to confirm a person is on the ASIC register. If you need the actual ASIC record (officeholdings, shareholdings, addresses, etc.) use `/asic_extract_person_search` and `/asic_extract` instead.

Match rules

- First name, last name and `birth_date` must all match an ASIC record for the check to pass.
- `middle_name`, when supplied, is reported on but does **not** gate the result. A check can pass with `middle_name: no_match`. ASIC's middle-name records are inconsistent - it is common for them to be missing entirely, recorded as an initial only, or only partially recorded - so we treat the middle name as a soft signal rather than a hard requirement.

Name matching

Name comparisons are case-insensitive and use the following normalisation rules on both the supplied value and the ASIC record:

- **Case.** `Smith`, `SMITH`, and `smith` are treated as the same name.
- **Accents and non-ASCII characters.** Transliterated to ASCII before comparison. `Müller` is treated as `Muller`, `André` as `Andre`.
- **Apostrophes.** Ignored. `O'Donnell` and `ODonnell` are treated as the same name. Both straight (`'`) and curly (`’`) apostrophes are normalised.
- **Hyphens.** Significant. `Smith-Jones` and `Smith Jones` (or `SmithJones`) are **not** treated as the same name - they should be supplied exactly as the ASIC record has them.
- **Punctuation and brackets.** Dots and parenthesised content are stripped before comparison (`John (Jack)` is treated as `John`, `St.` as `St`).
- **Whitespace.** Repeated whitespace is collapsed to single spaces and leading/trailing whitespace is trimmed.

There is no fuzzy matching - all comparisons are exact after the normalisation rules above.

Input expectations

- Names should be provided as they appear on ASIC records. A combined maximum of three given names is allowed across `first_name` and `middle_name`.
- `birth_date` must be an ISO `YYYY-MM-DD` date after 1900-01-01.

Response

The `match_results` object has two keys:

- `person`: `match` when the supplied first name, last name and date of birth match an ASIC record; otherwise `no_match`. `no_match` also covers the case where the search criteria are too broad to evaluate against a single record.
- `middle_name`: `match` when the supplied middle name matches the ASIC record, `no_match` when it does not (or the ASIC record has no middle name on file), or `not_provided` when no middle name was supplied.

Sandbox environment

The sandbox dataset is deterministic. Use any of the following combinations to receive a passing match:

First Name	Middle Name	Last Name	Birth Date	Notes
John	Michael	Doe	1980-05-15	Standard three-part name.
Maria		Scott	1971-11-02	No middle name on the ASIC record.
Patrick		O'Connor	1965-07-19	Apostrophe in the surname (also matches `OConnor`).
Mary-Jane		Parker	1983-02-28	Hyphenated first name.

Request body

Details of the individual to check.

Field	Description
`first_name` REQUIRED	**string [max 100 characters]** The person's first given name (one to three given names when combined with `middle_name`). **Example:** `John`
`middle_name`	**string [max 100 characters] or null** Optional middle names. Combined with `first_name` this cannot exceed three words. **Example:** `Michael`
`last_name` REQUIRED	**string [max 100 characters]** The person's family name as recorded with ASIC. **Example:** `Doe`

Field	Description
`birth_date` REQUIRED	**string (date)** The person's date of birth in `YYYY-MM-DD` format (must be after 1900-01-01). **Example:** `1980-05-15`

Sample request

```
{
    "first_name": "John",
    "middle_name": "Michael",
    "last_name": "Doe",
    "birth_date": "1980-05-15"
}
```

Responses

200 **ID check completed. The `match_results` object contains the verification outcome.** (application/json)

Field	Description
`message`	**string** Always `Ok` when the request succeeds. **Example:** `Ok`
`function`	**string** The API function that handled the request. **Example:** `asic_id_check`
`api_reference`	**string (uuid)** Audit reference for this ID check. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`match_results`	**object** The verification outcome. Carries no PII.
`match_results.` `person`	**string** `match` when the first name, last name and date of birth match an ASIC record; otherwise `no_match`. **Enum:** `match`, `no_match` **Example:** `match`

Field	Description
match_results. middle_name	string `match` when the supplied middle name matches the ASIC record's middle name, `no_match` when it does not (or the ASIC record has no middle name on file), or `not_provided` when no middle name was supplied in the request. **Enum:** `match`, `no_match`, `not_provided` **Example:** `match`

Sample response

```
{
    "message": "Ok",
    "function": "asic_id_check",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "match_results": {
        "person": "match",
        "middle_name": "match"
    }
}
```

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

ASIC Extract

POST `/asic_extract`

Requests an ASIC company or person extract. The response confirms the job was queued and returns an `api_reference`. Poll `GET /asic_extract/{api_reference}` until the extract payload is ready.

Request scenarios

- **Company extract** – set `type` to `company`, provide either an ABN or ACN, and choose an `extract_type`:
  - `current`: Requests an ASIC Current Extract.
  - `historical`: Requests an ASIC Current and Historical Extract.
  - `relational`: Request an ASIC Relational Extract.
- **Person extract** – set `type` to `person` and supply the `person_id`, `search_id`, legal `name`, and `birth_date` exactly as returned by `/asic_extract_person_search`. Linked identifiers from that search should be listed in `additional_person_ids` so the resulting extract stitches together all associated roles/shareholdings. The primary `person_id` must not appear inside `additional_person_ids`.

Sandbox dataset

The sandbox accepts deterministic identifiers so you can exercise both request types without billing. Use one of the following combinations:

Companies

Name	ABN	ACN	extract_type	Notes
Tech Innovators Ltd	32111111114	111111114	current	Current Extract
Green Energy Solutions	11123456780	123456780	historical	Current and Historical Extract
Urban Development Corp	54222222228	222222228	relational	Relational Extract
Creative Media Agency	33234567894	234567894	current	Current Extract simulating multiple polling for completion

People

Name	Birth Date	Extract Type	person_id	Search ID	Additional Person Id
JOHN MICHAEL DOE	1980-05-15	historical	123456780	11223344	123456781
JANE ALICE SMITH	1987-03-22	current	123456790	55667788	123456791
LUCAS BENJAMIN CARTER	1992-08-09	current	998877660	77889900	(none)

Request body

Parameters describing the ASIC entity you wish to extract.

Field	Description
`type` REQUIRED	**string** "company" for organisation extracts or "person" for individual extracts. **Enum:** `company`, `person` **Example:** `company`
`extract_type` REQUIRED	**string** Company extracts accept `current`, `historical`, or `relational`. Person extracts accept `current` or `historical` only. **Enum:** `current`, `historical`, `relational` **Example:** `current`
`abn`	**string or null** Required when `type` is `company` and an ACN is not supplied. Provide 11 digits (whitespace ignored). **Example:** `32111111114`

Field	Description
`acn`	**string or null** Required when `type` is `company` and an ABN is not supplied. Provide 9 digits (whitespace ignored). **Example:** `111111114`
`person_id`	**string or null** Required for person extracts. Copy the `person_id` provided by `/asic_extract_person_search`. **Example:** `123456780`
`additional_person_ids`	**array of strings or null** Optional list of related ASIC person identifiers. Do **not** repeat the primary `person_id`. Use the additional IDs returned by `/asic_extract_person_search` (or derived from sandbox keys containing `|`).
`search_id`	**string or null** Required for person extracts. Copy the `search_id` returned by `/asic_extract_person_search`. **Example:** `11223344`
`name`	**string or null** Required for person extracts. Uppercase legal name matching the search result. **Example:** `JOHN MICHAEL DOE`
`birth_date`	**string (date) or null** Required for person extracts. ISO `YYYY-MM-DD` date of birth taken from the search result. **Example:** `1980-05-15`

Sample request

```
{
    "type": "company",
    "extract_type": "current",
    "abn": "32111111114",
    "acn": "111111114",
    "person_id": "123456780",
    "additional_person_ids": [
        "123456781"
    ],
    "search_id": "11223344",
    "name": "JOHN MICHAEL DOE",
    "birth_date": "1980-05-15"
```

```
}
```

Responses

200 **Extract request accepted.** (application/json)

Field	Description
`message`	**string** Always `Ok` on success. **Example:** `Ok`
`function`	**string** The API function that handled the request. **Example:** `asic_extract`
`api_reference`	**string (uuid)** Audit reference for this extract request. Poll `/asic_extract/{api_reference}` to retrieve results. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Sample response

```
{
    "message": "Ok",
    "function": "asic_extract",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95"
}
```

501 **Summary extracts are not implemented.** (application/json)

Field	Description
`message`	**string** Static error returned when `summary=true` is requested. **Example:** `Summary ASIC extracts are not supported yet.`

Sample response

```
{
    "message": "Summary ASIC extracts are not supported yet."
}
```

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

Retrieve ASIC Extract

GET `/asic_extract/{api_reference}`

Polls the status of an ASIC extract that was previously queued via `POST /asic_extract`. Provide the `api_reference` returned from the original request to determine whether the extract is ready, download the JSON payload, or stream a PDF copy. This retrieval call is not billable and can be repeated until the extract is ready.

- While the extract is still compiling the API responds with HTTP `202`. Keep polling with an exponential backoff.
- Once ready, the endpoint returns HTTP `200` with the full JSON extract. The JSON payload includes the exact ASIC structure from the upstream dataset (for both company and person extracts) plus the `asic_extract_show` `api_reference` for auditing.
- Append `?pdf=true` to stream a formatted PDF copy of the same extract. When `pdf=true`, the response body is a PDF (`application/pdf`) instead of JSON.
- The service polls upstream data for up to 30 seconds; if no extract is available at that point you will receive a `202` response. Continue polling with the same `api_reference`.
- An extract can only be retrieved for 24 hours after the original `POST /asic_extract` request. Once that window has passed the endpoint responds with HTTP `410` and you need to submit a new extract request.

Sandbox usage

Use the sandbox company/person identifiers listed under `POST /asic_extract` to create a request, then poll this endpoint with the returned `api_reference`. Two fixtures are configured to **always** return a `202` on the first poll to help you test retry logic:

- Company: Creative Media Agency (`ABN 33234567894` , `ACN 234567894`)
- Person: Lucas Benjamin Carter (`person_id 998877660` , `search_id 77889900`)

Other sandbox extracts typically complete within a few seconds but still require polling – you may see a `202` response before the sample payload is ready.

Path parameters

Field	Description
`api_reference` REQUIRED	string (uuid) The audit reference returned when the extract was queued via `POST /asic_extract`. Example: `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Query parameters

Field	Description

`pdf`	**boolean** When set to `true`, returns the extract as a PDF document instead of JSON. Omit (or send `false`) to receive the JSON payload. **Default:** `false`

Responses

200 **Extract is ready. Returns JSON by default or a PDF when `pdf=true`.** (application/json, application/pdf)

Field	Description
`message`	**string** Always `Ok` when the extract payload is returned. **Example:** `Ok`
`function`	**string** Name of the API function that handled the request. **Example:** `asic_extract_show`
`api_reference`	**string (uuid)** Audit reference for **this** retrieval call (not the original queue reference). **Example:** `7cbb094c-0f27-42f4-9f36-9cf0b2f6d5ff`
`extract`	**object (dynamic)** Full ASIC extract payload. The structure mirrors the original request type: • Company extracts contain `entity_type: company`, `entity` metadata, `asic_extracts`, `directors`, `shareholders`, etc. • Person extracts contain `entity_type: person`, an `entity` block with the subject details, and related roles/shareholdings.

Sample response


```
{
    "message": "Ok",
    "function": "asic_extract_show",
    "api_reference": "7cbb094c-0f27-42f4-9f36-9cf0b2f6d5ff",
    "extract": {
        "id": "ee4c241c-93d3-4439-9a00-3fb551759168",
        "entity_type": "company",
        "entity": {
            "abn": "32111111114",
            "acn": "111111114",
            "name": "TECH INNOVATORS LTD",
            "abr_status": "Active"
        },
        "asic_extracts": [
```

```
            {
                "id": "ee4c241c-0001-4439-9a00-3fb551759168",
                "type": "Current",
                "directors": [
                    {
                        "name": "JANE DOE",
                        "type": "Director"
                    }
                ],
                "addresses": [
                    {
                        "type": "Registered Office",
                        "address": "100 SAMPLE STREET SYDNEY 2000 NSW"
                    }
                ]
            }
        ]
    }
}
```


Also available as a PDF (application/pdf)

202 **Extract is still being prepared (returned when no data is available after the 30?second polling window).**
(application/json)

Field	Description
`message`	**string** Status message indicating the extract is not ready yet. **Example:** `Extract is still being processed. Please try again later.`

Sample response


```
{
    "message": "Extract is still being processed. Please try again later."
}
```


404 **No extract was found for the provided `api_reference`, or it belongs to another account/environment.**
(application/json)

Field	Description
`message`	**string** Error message describing that the record could not be located. **Example:** `Not Found.`

Sample response


```
{
    "message": "Not Found."
}
```

410 **The extract can no longer be retrieved. Extracts are only available for retrieval for 24 hours after the original `POST /asic_extract` request. Submit a new extract request to obtain a current extract.** (application/json)

Field	Description
`message`	**string** Error message describing that the extract has expired. **Example:** `This extract has expired. Please request a new one`

Sample response

```
{
    "message": "This extract has expired. Please request a new one"
}
```

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

Business Check

POST `/business_check`

Performs a business lookup by ACN (9 digits) or ABN (11 digits), and combines ABR and ASIC data where available.

Input

- `number` must be a valid ABN or ACN.

Output

Returns:

- business identity and status details
- address details
- ASIC dates and recent document metadata
- business names and historical name details

Request body

Business check request details.

Field	Description
`number` REQUIRED	string ABN (11 digits) or ACN (9 digits). **Example:** `91605303875`

Sample request

```
{
    "number": "91605303875"
}
```

Responses

200 **Business check result.** (application/json)

Field	Description
`message`	string **Example:** `Ok`

Field	Description
`api_reference`	**string (uuid)** **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`match`	**one of 2 shapes** The matched business. An empty array `[]` when the number is not known to the ABR or ASIC.

Field	Description

BUSINESS CHECK MATCH

Field	Description
name	string or null Example: `SEMVIA FINANCE PTY LTD`
type	string or null Example: `Australian Private Company`
abn	string or null Example: `91605303875`
abn_status	string or null Example: `active`
abn_gst_registered	boolean Example: `true`
acn	string or null Example: `605303875`
acn_status	string or null Example: `registered`
address_locality	string or null Example: `MELBOURNE`
address_state	string or null Example: `VIC`
address_postcode	string or null Example: `3000`
asic_reg_date	string or null Example: `2015-04-15`
asic_review_date	string or null Example: `2026-04-15`
asic_deregistered_date	string or null
asic_recent_documents	array of objects
asic_recent_documents[]. form_code	string Example: `484`
asic_recent_documents[]. description	string Example: `CHANGE TO COMPANY DETAILS`

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "match": {
        "name": "SEMVIA FINANCE PTY LTD",
        "type": "Australian Private Company",
        "abn": "91605303875",
        "abn_status": "active",
        "abn_gst_registered": true,
        "acn": "605303875",
        "acn_status": "registered",
        "address_locality": "MELBOURNE",
        "address_state": "VIC",
        "address_postcode": "3000",
        "asic_reg_date": "2015-04-15",
        "asic_review_date": "2026-04-15",
        "asic_deregistered_date": null,
        "asic_recent_documents": [
            {
                "form_code": "484",
                "description": "CHANGE TO COMPANY DETAILS",
                "date": "2021-07-07",
                "document_number": "7EBJ24000",
                "sub_documents": [
                    {
                        "form_code": "484N",
                        "description": "CHANGES TO (MEMBERS) SHARE HOLDINGS"
                    }
                ]
            }
        ],
        "business_names": [
            {
                "name": "Payexpress",
                "from": "2015-07-16",
                "to": null,
                "type": "business_name"
            }
        ],
        "name_details": [
            {
                "type": "organisation",
                "entity_name": "SEMVIA FINANCE PTY LTD",
                "from": "2015-04-24",
                "to": null
            }
        ]
    }
}
```


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

Business Check Bulk

POST `/business_checks`

Performs bulk business checks using ABN/ACN values.

Limits

- `requests` must contain between 1 and 100 items.

Request body

Bulk business check request details.

Field	Description
`requests` REQUIRED	**array of objects [1..100 items]**
`requests[].` `check_id`	**string [max 255 characters] or null** **Example:** `req-1001`

`requests[].`
`number`
REQUIRED
	string
	ABN (11 digits) or ACN (9 digits).
	Example: `605303875`

Sample request

```
{
    "requests": [
        {
            "check_id": "req-1001",
            "number": "605303875"
        }
    ]
}
```

Responses

200 **Bulk business check result.** (application/json)

Field	Description
`message`	**string** **Example:** `Ok`
`api_reference`	**string (uuid)** **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`results`	**array of objects**
`results[].` `check_id`	**string or null** **Example:** `req-1001`

Field	Description
results[]. error	**string or null** **Example:** `The number is not a valid ACN.`
results[]. match	**one of 2 shapes** The matched business. An empty array `[]` when the number is not known to the ABR or ASIC.

Field	Description

BUSINESS CHECK MATCH

Field	Description
name	string or null Example: `SEMVIA FINANCE PTY LTD`
type	string or null Example: `Australian Private Company`
abn	string or null Example: `91605303875`
abn_status	string or null Example: `active`
abn_gst_registered	boolean Example: `true`
acn	string or null Example: `605303875`
acn_status	string or null Example: `registered`
address_locality	string or null Example: `MELBOURNE`
address_state	string or null Example: `VIC`
address_postcode	string or null Example: `3000`
asic_reg_date	string or null Example: `2015-04-15`
asic_review_date	string or null Example: `2026-04-15`
asic_deregistered_date	string or null
asic_recent_documents	array of objects
asic_recent_documents[]. form_code	string Example: `484`
asic_recent_documents[]. description	string Example: `CHANGE TO COMPANY DETAILS`

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "results": [
        {
            "check_id": "req-1001",
            "error": "The number is not a valid ACN.",
            "match": {
                "name": "SEMVIA FINANCE PTY LTD",
                "type": "Australian Private Company",
                "abn": "91605303875",
                "abn_status": "active",
                "abn_gst_registered": true,
                "acn": "605303875",
                "acn_status": "registered",
                "address_locality": "MELBOURNE",
                "address_state": "VIC",
                "address_postcode": "3000",
                "asic_reg_date": "2015-04-15",
                "asic_review_date": "2026-04-15",
                "asic_deregistered_date": null,
                "asic_recent_documents": [
                    {
                        "form_code": "484",
                        "description": "CHANGE TO COMPANY DETAILS",
                        "date": "2021-07-07",
                        "document_number": "7EBJ24000",
                        "sub_documents": []
                    }
                ],
                "business_names": [
                    {
                        "name": "Payexpress",
                        "from": "2015-07-16",
                        "to": null,
                        "type": "business_name"
                    }
                ],
                "name_details": [
                    {
                        "type": "organisation",
                        "entity_name": "SEMVIA FINANCE PTY LTD",
                        "from": "2015-04-24",
                        "to": null
                    }
                ]
            }
        }
    ]
}
```


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

Create a Business Intelligence Request

POST `/business_intelligence`

Experimental

The Business Intelligence API is still in the experimental stage and is subject to change.

The Business Intelligence API performs a deep dive into the business behind a given ABN.

It will return a summary of the business, including:

- The business name and other trading names
- The ABN and ACN (if applicable)
- The business address
- The business website
- The business ANZSIC code and description
- The AUSTRAC remitter details (if applicable)

The API will also return a list of all the domains that were checked as part of the request.

Sandbox Mode

When accessing the Business Intelligence API in sandbox mode, only ABNs ending in 1, 2 or 3 are permitted. Attempting to use any other ABN will result in a 400 error. with the following message:

```
This service is only available for ABNs ending in 1, 2 or 3 in sandbox mode.
```

Note: Other ABNs will be accepted in sandbox mode, however the results may contain cached data. Results obtained in sandbox mode are not permitted to be used for production purposes.

Request body

Request details for the business intelligence report. Note that the query can take several minutes to complete. Best practice is to poll the status of the request using the `GET /business-intelligence/{request_uuid}` endpoint.

Field	Description
`abn`	string The ABN to request business intelligence for **Example:** `12345678901`

Sample request

```
{
    "abn": "12345678901"
}
```

Responses

200 **Details of the newly created business intelligence request** (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 for support queries. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`request_uuid`	**string (uuid)** The identifier for this business intelligence request. This will be used to retrieve the request results. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Sample response

```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "request_uuid": "fe4291ca-d831-4760-96df-c9cb03b3cd95"
}
```

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

Retrieve a Business Intelligence Request

GET `/business_intelligence/{request_uuid}`

Experimental

The Business Intelligence API is still in the experimental stage and is subject to change.

Path parameters

Field	Description

`request_uuid` REQUIRED	**string (uuid)** The identifier for the business intelligence request to retrieve **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Responses

200 **Details of the business intelligence request** (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 for support queries. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`status`	**string** The status of the business intelligence request • pending: The request has been created but not yet begun processing • processing: The request is currently being processed (See status_message for more details) • complete: The request has been completed • failed: The request has failed **Enum:** `pending`, `processing`, `complete`, `failed` **Example:** `pending`
`status_message`	**string** A message indicating what the current status of the request is **Example:** `Request is still processing.`
`data`	**object** The data returned by the request. This is available when the request is complete.
`data.` `abn`	**string** The ABN of the business **Example:** `12345678901`

Field	Description
data. abn_name	**object**
data. abn_name. entity_name	**string** The name of the business **Example:** `ACME Pty Ltd`
data. abn_name. from	**string (date)** The start date of for this entity name **Example:** `2020-01-01`
data. abn_name. to	**string (date) or null** The end date of for this entity name or null if it is the current entity name
data. abn_gst_status	**object** The history of the GST status of the business. Empty if the business is not registered for GST.
data. abn_gst_status. from	**string (date)** The start date of for this GST status **Example:** `2020-01-01`
data. abn_gst_status. to	**string (date) or null** The end date of for this GST status or null if it is the current GST status
data. abn_charity_status	**object** The history of the charity status of the business. Empty if the business is not a charity.
data. abn_charity_status. from	**string (date)** The start date of for this charity status **Example:** `2020-01-01`
data. abn_charity_status. to	**string (date) or null** The end date of for this charity status or null if it is the current charity status
data. abn_previous_names	**array of objects** The previous names of the business.

Field	Description
data. abn_previous_names[]. name	**string** The name of the business **Example:** `ACME Pty Ltd`
data. abn_previous_names[]. from	**string (date)** The start date of for this previous name **Example:** `2020-01-01`
data. abn_previous_names[]. to	**string (date) or null** The end date of for this previous name or null if it is the current previous name
data. abn_status	**string** The current status of the business **Example:** `Active`
data. abn_type	**string** The type of business **Example:** `Australian Private Company`
data. abn_reg_date	**string (date)** The date the ABN was registered **Example:** `2020-01-01`
data. abn_location	**string** The location of the business (State and Postcode) **Example:** `NSW 2000`
data. acn	**string** The ACN of the business (if applicable) **Example:** `123456789`
data. acn_status	**string** The status of the ACN (if applicable) **Example:** `Registered`

Field	Description
data. acn_reg_date	**string (date)** The date the ACN was registered (if applicable) **Example:** `2020-01-01`
data. acn_recent_documents	**array of objects** The recent ASIC documents of the business (if applicable)
data. acn_recent_documents[]. documentNumber	**string** The document number **Example:** `12345678901`
data. acn_recent_documents[]. dateReceived	**string (date)** The date the document was received **Example:** `2020-01-01`
data. acn_recent_documents[]. formCode	**string** The ASIC form code of the document **Example:** `488`
data. acn_recent_documents[]. numberOfPages	**integer** The number of pages in the document **Example:** `4`
data. acn_recent_documents[]. description	**string** The description of the document **Example:** `APPLICATION TO CHANGE REVIEW DATE OF AN ENTITY`
data. acn_recent_documents[]. additionalDescription	**array of objects** Additional description of the document
data. acn_recent_documents[]. additionalDescription[]. subformCode	**string** The subform code of the document **Example:** `488B`
data. acn_recent_documents[]. additionalDescription[]. subformDescription	**string** The description of the subform **Example:** `Change of review date`

Field	Description
data. business_names	**array of objects** The business names of the business
data. business_names[]. name	**string** The name of the business **Example:** `ACME Services`
data. business_names[]. from	**string (date)** The start date for this business name **Example:** `2020-01-01`
data. business_names[]. to	**string (date) or null** The end date for this business name or null if it is the current business name
data. business_names[]. type	**string** The type of business name **Example:** `business_name`
data. business_names[]. previous_state_number	**string** The previous state number of the business name (if applicable) **Example:** `12345678901`
data. business_names[]. previous_state_name	**string** The previous state name of the business name (if applicable) **Example:** `NSW`
data. domains	**array of strings** The domains of the business
data. websites	**array of objects** The websites of the business
data. websites[]. domain	**string** The domain of the website **Example:** `acme.com`

Field	Description
data. websites[]. type	**string** The type of website **Enum:** `active` , `archived` **Example:** `active`
data. websites[]. summary	**string** The summary of the website **Example:** `ACME is a leading provider of accounting and financial services.`
data. summary	**string** The summary of the business **Example:** `ACME is a leading provider of accounting and financial services.`
data. anzsic_code	**string** The ANZSIC code of the business **Example:** `6419`
data. anzsic_description	**string** The ANZSIC description of the business **Example:** `Other Auxiliary Finance and Investment Services`
data. remitter_details	**array of objects** The AUSTRAC remitter details of the business
data. remitter_details[]. id	**string** The ID of the remitter **Example:** `12345678901`
data. remitter_details[]. legalName	**string** The legal name of the remitter **Example:** `ACME Pty Ltd`
data. remitter_details[]. tradingNames	**array of strings** The trading names of the remitter

Field	Description
data. remitter_details[]. abn	**string** The ABN of the remitter **Example:** `12345678901`
data. remitter_details[]. acn	**string** The ACN of the remitter **Example:** `123456789`
data. remitter_details[]. arbn	**string** The ARBN of the remitter **Example:** `12345678901`
data. remitter_details[]. address	**string** The address of the remitter **Example:** `123 Main St, Sydney NSW 2000`
data. log	**array of strings** The log of the business intelligence request

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "status": "pending",
    "status_message": "Request is still processing.",
    "data": {
        "abn": "12345678901",
        "abn_name": {
            "entity_name": "ACME Pty Ltd",
            "from": "2020-01-01",
            "to": null
        },
        "abn_gst_status": {
            "from": "2020-01-01",
            "to": null
        },
        "abn_charity_status": {
            "from": "2020-01-01",
            "to": null
        },
        "abn_previous_names": [
            {
                "name": "ACME Pty Ltd",
                "from": "2020-01-01",
                "to": null
            }
```

```
        ],
        "abn_status": "Active",
        "abn_type": "Australian Private Company",
        "abn_reg_date": "2020-01-01",
        "abn_location": "NSW 2000",
        "acn": "123456789",
        "acn_status": "Registered",
        "acn_reg_date": "2020-01-01",
        "acn_recent_documents": [
            {
                "documentNumber": "12345678901",
                "dateReceived": "2020-01-01",
                "formCode": "488",
                "numberOfPages": 4,
                "description": "APPLICATION TO CHANGE REVIEW DATE OF AN ENTITY",
                "additionalDescription": [
                    {
                        "subformCode": "488B",
                        "subformDescription": "Change of review date"
                    }
                ]
            }
        ],
        "business_names": [
            {
                "name": "ACME Services",
                "from": "2020-01-01",
                "to": null,
                "type": "business_name",
                "previous_state_number": "12345678901",
                "previous_state_name": "NSW"
            }
        ],
        "domains": [
            "acme.com",
            "acme.com.au"
        ],
        "websites": [
            {
                "domain": "acme.com",
                "type": "active",
                "summary": "ACME is a leading provider of accounting and financial services."
            }
        ],
        "summary": "ACME is a leading provider of accounting and financial services.",
        "anzsic_code": "6419",
        "anzsic_description": "Other Auxiliary Finance and Investment Services",
        "remitter_details": [
            {
                "id": "12345678901",
                "legalName": "ACME Pty Ltd",
                "tradingNames": [
                    "ACME Pty Ltd"
                ],
                "abn": "12345678901",
                "acn": "123456789",
                "arbn": "12345678901",
                "address": "123 Main St, Sydney NSW 2000"
            }
        ],
        "log": [
            "Performing ABN lookup",
```

```
            "Performing domain lookup",
            "..."
        ]
    }
}
```

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

Caspar Person Search

POST `/caspar_person_search`

The Caspar Person Search returns a list of people and contact details based on the supplied search parameters. The search can be performed using a combination of name, address, phone number and email address.

Search Types

The person search can be used in a number of ways using different parameters and parameter combinations:

- **Name Search**: Searching for a name (or partial name) will return all people matching that name along with their address and any associated phone numbers and email addresses. An optional DOB range can also be included.
- **Phone Search**: Searching for a phone number (or partial phone number) will return all people connected to that phone number along with their address and any additional associated phone numbers and email addresses.
- **Email Search**: Searching for an email address will return all people connected to that email along with their address and any additional associated phone numbers and email addresses.
- **IP Address Search**: Searching for an IP address will return all people who have an online record associated with that IP address. Only IPv4 addresses are supported; IPv6 addresses are not currently supported.
- **Social Search**: Searching for a social media profile URL will return all people connected to that social profile. Supported social networks include Facebook, LinkedIn, Twitter, YouTube, Instagram, Indeed, Github, Meetup, Quora, Gravatar, Foursquare, and Pinterest. The URL can include or omit the `http://` / `https://` scheme and an optional leading `www.`, for example, `facebook.com/person123`, `www.facebook.com/person123`, and `https://www.facebook.com/person123` will all match the same person.
- **Address Search**: Searching for an address will return all people (current and historical) at that address along with any associated phone numbers and email addresses. It is also possible to search for an entire street and return all addresses and people on that street.

Search Operation

By default, the search returns records which match **all** of the supplied elements using a logical AND:

> Name **AND** Address **AND** Phone **AND** Email **AND** IP Address **AND** Social

However, many end users prefer a mode where the name and address are searched together, but records matching just the phone, email, IP address, or social profile are returned as well. This is accomplished by enabling the `option_smartsearch` option, in which case the logic becomes:

> (Name **AND** Address) **OR** Phone **OR** Email **OR** IP Address **OR** Social

Minimum Search Requirements

At least one of the following must be provided:

- `name_last` or `name_combined`
- `phone`
- `email`
- `ip_address`
- `social_url`
- `full_address` or `street_address`

A name search requires at least the last name. A name and partial address search is permitted for:

Search	Records Returned
Name and state	All occurrences of the name in the state
Name and suburb	All occurrences of the name in the suburb
Name and postcode	All occurrences of the name in the postcode
Name and street (+ suburb/state/postcode)	All occurrences of the name on the street

DOB Searching

A name search can be combined with DOB bounds (`dob_from` and/or `dob_to`) to restrict results to records where a DOB falls within the specified range. Supplying only one bound will return results above or below the specified date.

Smart Options

Option	Effect
`option_smartname`	Matches initials, first-name equivalents (Robert = Bob/Rob), and similar-sounding last names (Smith = Smyth)
`option_smartemail`	Matches the email username across different domains and TLDs
`option_smartphone`	Matches partial phone numbers across area codes and mobile prefixes
`option_smartaddress`	Matches similar or misspelled addresses in the selected area or neighbouring suburbs
`option_smartsearch`	Changes search logic to: (Name AND Address) OR Phone OR Email OR IP Address OR Social

Sort Order

Result sorting is controlled by the `sort_by` and `sort_order` parameters. If no sort parameters are supplied, results are returned in a default order that takes into account the search parameters and attempts to place the most relevant results first.

Secondary Person Search

A secondary person can be included in the search using the `secondary_name_*` and `secondary_dob_*` parameters. This will search for both people at the same address.

Sandbox environment data

When simulating queries in the sandbox environment, the following records can be used to return results:

Sample records

Title	First Name	Middle Name	Last Name	Street Address	Suburb	State	Postcode	Phone	Email
MS	Emma	Sandra	Russell	4 Westmill Dr	Hoppers Crossing	VIC	3029	0427519643	
	Shaun		Pound	18 Ardisia Ct	Burleigh Heads	QLD	4220	0436991031, 0755687356	pdffyr41@denceads.com.au
MR	William		Angell	16 Hex St	West Footscray	VIC	3012		ml1972@listratence.net.au

Sample marketing_opt_in flags

When `option_marketing_opt_in=true` is sent, the sample records above resolve to the following opt-in flags so that callers can observe both `true` and `false` values:

Person	Contact	marketing_opt_in
Emma Sandra Russell	phone 0427519643	`true`
Shaun Pound	phone 0436991031	`true`
Shaun Pound	phone 0755687356	`false`
Shaun Pound	email pdffyr41@denceads.com.au	`true`
William Angell	email ml1972@listratence.net.au	`false`

Addresses with deep history

Searching for the following addresses will return multiple residents demonstrating deep address history:

Address	Expected Records
375 ARGENT ST, BROKEN HILL NSW 2880	10 residents
UNIT 4/88 BROOK ST, COOGEE NSW 2034	Multiple residents
42 HOFF ST, MOUNT GRAVATT EAST QLD 4122	Multiple residents

Request body

The search criteria for finding people and their contact details.

Field	Description
`name_combined`	**string** The full name to search for as a single string (e.g. "MARY SALLY JONES") **Example:** `MARY JONES`

Field	Description
`name_first`	**string** First name of the person to search **Example:** `MARY`
`name_middle`	**string** Middle name of the person to search **Example:** `SALLY`
`name_last`	**string** Last (family) name of the person to search **Example:** `JONES`
`dob_from`	**string (date)** Lower bound date of birth (YYYY-MM-DD) **Example:** `1980-01-01`
`dob_to`	**string (date)** Upper bound date of birth (YYYY-MM-DD) **Example:** `1995-12-31`
`secondary_name_first`	**string** First name of a second person to search (same address) **Example:** `JOHN`
`secondary_name_middle`	**string** Middle name of the second person to search
`secondary_name_last`	**string** Last (family) name of the second person to search **Example:** `JONES`
`secondary_dob_from`	**string (date)** Lower bound DOB of the second person (YYYY-MM-DD)
`secondary_dob_to`	**string (date)** Upper bound DOB of the second person (YYYY-MM-DD)

Field	Description
`street_address`	**string** First line of the street address **Example:** `35 YARALLA ST`
`suburb`	**string** Suburb of the address **Example:** `CONCORD WEST`
`state`	**string** Australian state of the address **Enum:** `ACT`, `NSW`, `NT`, `QLD`, `SA`, `TAS`, `VIC`, `WA` **Example:** `NSW`
`postcode`	**string** Postcode of the address **Example:** `2138`
`suburb_state_postcode`	**string** A single string containing the suburb, state and postcode (address line 2) **Example:** `CONCORD WEST NSW 2138`
`full_address`	**string** A single string containing the complete address **Example:** `35 YARALLA ST CONCORD WEST NSW 2138`
`phone`	**string** Phone number of the person to search **Example:** `0299995000`
`email`	**string (email)** Email address of the person to search **Example:** `mary_jones89@example.com`

Field	Description
`ip_address`	**string (ipv4)** IPv4 address to search for. Returns people who have an online record associated with this IP address. IPv6 addresses are not currently supported. **Example:** `203.0.113.42`
`social_url`	**string** Social media profile URL to search for. Returns people connected to the specified social profile. The URL can include or omit the `http://` / `https://` scheme and an optional leading `www.`. Supported social networks: Facebook, LinkedIn, Twitter, YouTube, Instagram, Indeed, Github, Meetup, Quora, Gravatar, Foursquare, and Pinterest. **Example:** `facebook.com/person123`
`option_smartname`	**boolean** Enable smart name matching (initials, equivalents, similar-sounding) **Default:** `false`
`option_smartemail`	**boolean** Enable smart email matching (cross-domain, cross-TLD) **Default:** `false`
`option_smartphone`	**boolean** Enable smart phone matching (partial numbers, cross-area-code) **Default:** `false`
`option_smartaddress`	**boolean** Enable smart address matching (similar/misspelled addresses, neighbouring suburbs) **Default:** `false`
`option_smartsearch`	**boolean** Enable smart search mode which changes the search logic to: (Name AND Address) OR Phone OR Email **Default:** `false`

Field	Description
option_marketing_opt_in	**boolean** Include a `marketing_opt_in` block on each record indicating whether each phone number and email address is registered as opted-in for marketing for this specific person. Opt-in is determined per-person, per-contact: the same email or phone may be opted-in for one person but not for another (an email shared by partners, for example, can be marketing-opted-in for only one of them). A value of `true` means an explicit opt-in record exists for that contact under that person; `false` means that no opt-in record exists. Note that the opt-in status on a phone number does not negate the requirement to perform a DNC check on the phone number before attempting to call the person. **Default:** `false`
first_result	**integer [min 0]** Index of first result to return for pagination (0-based) **Default:** `0` **Example:** `0`
max_results	**integer [1..30]** Maximum number of results to return (1-30) **Default:** `10` **Example:** `10`
sort_by	**string** Field to sort results by **Enum:** `name` , `dob` , `phone` , `email` , `state` , `suburb` , `postcode` , `street_address` , `address_combined` , `updated` **Example:** `name`
sort_order	**string** Sort direction **Enum:** `ASC` , `DESC` **Example:** `ASC`

Sample request

```
{
    "name_combined": "MARY JONES",
```

```
    "name_first": "MARY",
    "name_middle": "SALLY",
    "name_last": "JONES",
    "dob_from": "1980-01-01",
    "dob_to": "1995-12-31",
    "secondary_name_first": "JOHN",
    "secondary_name_middle": "string",
    "secondary_name_last": "JONES",
    "secondary_dob_from": "2024-01-01",
    "secondary_dob_to": "2024-01-01",
    "street_address": "35 YARALLA ST",
    "suburb": "CONCORD WEST",
    "state": "NSW",
    "postcode": "2138",
    "suburb_state_postcode": "CONCORD WEST NSW 2138",
    "full_address": "35 YARALLA ST CONCORD WEST NSW 2138",
    "phone": "0299995000",
    "email": "mary_jones89@example.com",
    "ip_address": "203.0.113.42",
    "social_url": "facebook.com/person123",
    "option_smartname": true,
    "option_smartemail": true,
    "option_smartphone": true,
    "option_smartaddress": true,
    "option_smartsearch": true,
    "option_marketing_opt_in": true,
    "first_result": 0,
    "max_results": 10,
    "sort_by": "name",
    "sort_order": "ASC"
}
```


Responses

200 **Successful search response with matching person records.** (application/json)

Field	Description
`message`	**string** A message indicating the result of the request: • `Ok` - Search was successful • `No match` - No records found for the search criteria • `Found too many results` - The search matched too many records. Try providing more specific search criteria. **Example:** `Ok`
`function`	**string** The API function that handled the request. **Example:** `caspar_person_search`

Field	Description
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`records`	**array of objects** Array of matching person records.
`records[].title`	**string** Title of the matched person **Example:** `MS`
`records[].first_name`	**string** First name of the matched person **Example:** `MARY`
`records[].middle_name`	**string** Middle name of the matched person **Example:** `SALLY`
`records[].last_name`	**string** Last name of the matched person **Example:** `JONES`
`records[].dob`	**string** Date of birth (year only, format YYYY) if available **Example:** `1989`
`records[].dob_range_from`	**string** DOB range minimum (year only, format YYYY) if no exact DOB is available **Example:**
`records[].dob_range_to`	**string** DOB range maximum (year only, format YYYY) if no exact DOB is available **Example:**

Field	Description
records[]. deceased	**string** Deceased flag **Enum:** `Y`, `N` **Example:** `N`
records[]. deceased_date	**string** Deceased date (year only, format YYYY) if available **Example:**
records[]. gender	**string** Gender of the matched person **Enum:** `MALE`, `FEMALE`, `X`, `UNKNOWN` **Example:** `FEMALE`
records[]. phones	**array of strings** Array of phone numbers associated with the matched person
records[]. emails	**array of strings** Array of email addresses associated with the matched person
records[]. socials	**array of objects** Social media accounts associated with the matched person (if available)
records[]. ip_addresses	**array of strings** IP addresses associated with the matched person (if available)
records[]. connectivity	**object** Cached connectivity check results for any phones and emails in the result set. Each entry is keyed by the phone number or email address itself. Arrays may be empty when no cached connectivity data is available. Real-time connectivity can be obtained via the `email_ping` and `phone_ping` endpoints.
records[]. connectivity. phones	**object (dynamic)** Cached phone ping results keyed by phone number. Only populated when a cached ping result exists for a phone number in the result set.
records[]. connectivity. emails	**object (dynamic)** Cached email ping results keyed by email address. Only populated when a cached ping result exists for an email address in the result set.

Field	Description
records[]. marketing_opt_in	**object** Per-contact marketing opt-in status for this person. Only present when `option_marketing_opt_in=true` was sent on the request. `true` indicates the contact is registered as opted-in for marketing for this specific person; `false` indicates either no opt-in record exists, the contact is not in the local marketing dataset, or this person could not be matched locally. Opt-in status is per-person-per-contact - the same email or phone may resolve to `true` for one person and `false` for another.
records[]. marketing_opt_in. phones	**object (dynamic)** Marketing opt-in flag keyed by phone number. Contains an entry for every phone number in the record's `phones` array.
records[]. marketing_opt_in. emails	**object (dynamic)** Marketing opt-in flag keyed by email address. Contains an entry for every email address in the record's `emails` array.
records[]. address_id	**string** Address persistent identifier **Example:** `GANSW716615055`
records[]. street_address	**string** The street/postal address as one string **Example:** `35 YARALLA ST`
records[]. suburb	**string** Suburb of the matched address **Example:** `CONCORD WEST`
records[]. state	**string** State of the matched address **Example:** `NSW`
records[]. postcode	**string** Postcode of the matched address **Example:** `2138`

Field	Description
records[]. address_parsed	**string** Whether the address passed the address parser (Y/N) **Enum:** `Y`, `N` **Example:** `Y`
records[]. address_valid	**string** Whether the address exists in the address table (Y/N) **Enum:** `Y`, `N` **Example:** `Y`
records[]. address_primary_secondary	**string** Whether address is a primary or secondary dwelling (P/S or blank) **Example:**
records[]. date_start	**string** Date the person was first seen at this address (YYYY-MM) **Example:** `2016-06`
records[]. date_end	**string** Date the person was last seen at this address (YYYY-MM) **Example:** `2019-07`
records[]. legal_parcel_id	**string** Legal parcel identifier for the address property **Example:** `15//DP240258`
records[]. meshblock_category	**string** ABS meshblock category for the address (e.g. Residential, Commercial, Parkland) **Example:** `Residential`
records[]. latitude	**string** Latitude coordinate of the address **Example:** `-33.72914350`

Field	Description
records[]. longitude	**string** Longitude coordinate of the address **Example:** `151.21036778`
records[]. score	**string** Match relevance score. For person search this is typically "0"; for autotrace results this is a meaningful ranking score. **Example:** `0`
records[]. realestate	**array of objects** Real estate events for this address
records[]. realestate[]. listing_type	**string** Type of listing: sale, sold, or rent **Enum:** `sale`, `sold`, `rent`
records[]. realestate[]. property_type	**string** Type of property (e.g. House, Apartment, Unit)
records[]. realestate[]. num_bedrooms	**string** Number of bedrooms (if available)
records[]. realestate[]. num_bathrooms	**string** Number of bathrooms (if available)
records[]. realestate[]. num_car_spaces	**string** Number of car spaces (if available)
records[]. realestate[]. date	**string (date)** Date of the listing (YYYY-MM-DD)
records[]. realestate[]. estate_agent	**string** Name of the estate agent
records[]. realestate[]. price	**string** Property price (where available)
records[]. judgements	**array of objects** Judgements or events for the person at this address

Field	Description
records[]. judgements[]. name	**string** The name for which the judgement is recorded
records[]. judgements[]. creditor	**string** Name of the creditor
records[]. judgements[]. event	**string** Description of the event
records[]. judgements[]. date	**string (date)** Date of listing (YYYY-MM-DD)
records[]. business	**array of objects** ABN records possibly associated with the person at the matched postcode
records[]. business[]. abn	**string** ABN of the matching entity **Example:** `12345678901`
records[]. business[]. status	**string** ABN status: ACT (Active) or CAN (Cancelled) **Enum:** `ACT`, `CAN`
records[]. business[]. effective_from	**string (date)** Date the status became effective (YYYY-MM-DD)
records[]. address_info	**array of objects** Additional information about the address
records[]. address_info[]. property_type	**string** Type of property: agedcare, prison, or emg_accom **Enum:** `agedcare`, `prison`, `emg_accom`
records[]. address_info[]. property_detail	**string** Description of the property

Field	Description
records[]. demographic	**object** SEIFA geo-demographic indices for the address area, based on SA1 level statistical analysis. Decile ranked from 0 to 10, where 10 indicates the top 10% of areas for that indicator and 0 means no data is available.
records[]. demographic. IRSAD_decile	**string** Index of Relative Socio-economic Advantage and Disadvantage (0-10). Focuses on financial aspects related to buying power, income and wealth. A low score indicates relative disadvantage (e.g. low income households); a high score indicates relative advantage (e.g. high income, home ownership). **Example:** `9`
records[]. demographic. IER_decile	**string** Index of Economic Resources (0-10). Focuses on financial aspects related to buying power, income and wealth. A low score indicates a relative lack of economic resources (e.g. low income, low rent); a high score indicates greater access to economic resources (e.g. high income, home ownership). **Example:** `8`
records[]. demographic. IEO_decile	**string** Index of Education and Occupation (0-10). Focuses on the educational and occupational level of the area. A low score indicates lower education and occupation status (e.g. no qualifications, low skilled or unemployed); a high score indicates higher education and occupation status (e.g. higher qualifications, highly skilled occupations). **Example:** `7`
records[]. court_record	**string or null** Whether a court record exists for this person (Y/N or null if not checked) **Enum:** `Y`, `N` **Example:** `N`
records[]. social_record	**string or null** Whether a social media record exists for this person (Y/N or null if not checked) **Enum:** `Y`, `N` **Example:** `N`

Field	Description
records[]. employment_record	**string or null** Whether an employment record exists for this person (Y/N or null if not checked) **Enum:** `Y`, `N` **Example:** `N`
records[]. sources	**array of strings** Data sources that contributed to this record (e.g. government, proprietary_records, competition). May be empty.
total_records_available	**integer** Total number of matching records available. **Example:** `1`

Sample response


```
{
    "message": "Ok",
    "function": "caspar_person_search",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "records": [
        {
            "title": "MS",
            "first_name": "MARY",
            "middle_name": "SALLY",
            "last_name": "JONES",
            "dob": "1989",
            "dob_range_from": "",
            "dob_range_to": "",
            "deceased": "N",
            "deceased_date": "",
            "gender": "FEMALE",
            "phones": [
                "0299995000",
                "0491999888"
            ],
            "emails": [
                "mary_jones89@example.com"
            ],
            "socials": [],
            "ip_addresses": [
                "string"
            ],
            "connectivity": {
                "phones": {
                    "0299995000": {
                        "check_date": "2020-03-23",
                        "status": "connected",
                        "carrier": "",
                        "geo_area": "Sydney",
                        "dnc_result": "",
```

```
                "dnc_check_date": "",
                "dnc_reference": ""
            },
            "0491999888": {
                "check_date": "2020-08-04",
                "status": "disconnected",
                "carrier": "Telstra",
                "geo_area": "",
                "dnc_result": "N",
                "dnc_check_date": "2020-08-04",
                "dnc_reference": ""
            }
        },
        "emails": {
            "mary_jones89@example.com": {
                "check_date": "2020-08-04",
                "status": "undeliverable"
            }
        }
    },
    "marketing_opt_in": {
        "phones": {
            "0299995000": true,
            "0491999888": false
        },
        "emails": {
            "mary_jones89@example.com": true
        }
    },
    "address_id": "GANSW716615055",
    "street_address": "35 YARALLA ST",
    "suburb": "CONCORD WEST",
    "state": "NSW",
    "postcode": "2138",
    "address_parsed": "Y",
    "address_valid": "Y",
    "address_primary_secondary": "",
    "date_start": "2016-06",
    "date_end": "2019-07",
    "legal_parcel_id": "15//DP240258",
    "meshblock_category": "Residential",
    "latitude": "-33.72914350",
    "longitude": "151.21036778",
    "score": "0",
    "realestate": [
        {
            "listing_type": "sale",
            "property_type": "string",
            "num_bedrooms": "string",
            "num_bathrooms": "string",
            "num_car_spaces": "string",
            "date": "2024-01-01",
            "estate_agent": "string",
            "price": "string"
        }
    ],
    "judgements": [
        {
            "name": "string",
            "creditor": "string",
            "event": "string",
```

```
                    "date": "2024-01-01"
                }
            ],
            "business": [
                {
                    "abn": "12345678901",
                    "status": "ACT",
                    "effective_from": "2024-01-01"
                }
            ],
            "address_info": [
                {
                    "property_type": "agedcare",
                    "property_detail": "string"
                }
            ],
            "demographic": {
                "IRSAD_decile": "9",
                "IER_decile": "8",
                "IEO_decile": "7"
            },
            "court_record": "N",
            "social_record": "N",
            "employment_record": "N",
            "sources": [
                "government",
                "proprietary_records"
            ]
        }
    ],
    "total_records_available": 1
}
```


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

Caspar Person Autotrace

POST `/caspar_person_autotrace`

The Caspar Person Autotrace performs an automatic trace on the supplied person and associated contact details, returning possible matches ranked by a match score. The more information provided about the target person, the better the results and score ranking will be.

Minimum Trace Requirements

At a minimum, a last name must be provided (`name_last` or `name_combined`). However, the quality of results will be lower without additional identifying information. The more information supplied (name, DOB, address, phone, email), the better the match scoring.

Match Scoring

Each result includes a `score` field - a unitless integer where higher scores indicate a better match. The score takes into account multiple parameters including the distance between the supplied address and the matched address. If no address is supplied, the score may not be available.

Relatives

Setting `option_relatives` to `true` will include possible relatives in the trace results. Relatives are people who have an association with the person being searched but do not match the person's full name. Each record includes a `match_type` field indicating whether it is a `primary` match or a `relative`.

Sort Order

Result sorting is controlled by the `sort_by` and `sort_order` parameters. If no sort parameters are supplied, results are ordered by descending score, which is the most useful ordering for autotrace results.

Performance

A single autotrace can perform tens to hundreds of internal searches to gather and rank results, which may take several seconds. Plan timeouts accordingly, particularly when performing multiple autotrace queries in sequence.

Sandbox environment data

When simulating queries in the sandbox environment, the following records can be used to return results. Each example demonstrates multiple trace results with varying match scores.

Multi-result examples

The following records return multiple trace results, demonstrating how autotrace ranks and returns possible matches:

First Name	Last Name	Address	Trace Results	Description
Bernard	Williams	1 MOJAVE DR, BURLEIGH WATERS QLD 4220	6 results	Strong primary match (score 1000) with 5 additional matches across QLD, SA, NSW, TAS, and WA
Kenneth	Hawkins	UNIT 46/10-12 FRENCH AV, BANKSTOWN NSW 2200	4 results	Four equally-scored matches (score 250) in the same state (NSW)
Peter	Clacher	15 KEARSLEY ST, BELLBIRD NSW 2325	4 results	Primary match (score 1000) with diminishing scores across NSW, QLD, and WA
Paul	Hayes	UNIT 4/118 ELIZABETH DR, LIVERPOOL NSW 2170	4 results	Primary match (score 1000) with additional matches including phone and email details
Andrew	Sharp	181-183 STATION RD, BURPENGARY QLD 4505	3 results	Primary match (score 1000) plus a name-variant match (Drew Sharp, score 16)

Single-result examples

The following records return a single strong match and are useful for testing basic autotrace functionality:

Title	First Name	Last Name	Address	Phone	Email
MR	Owen	Wyllie	4 BUTCHERBIRD CL, ELI WATERS QLD 4655	0741241873	bc68@pardswit.org.au
MR	Peter	Jenkin	2 TANAMI CL, BELROSE NSW 2085	0294511557, 0447336754	peter508@moraitive.net.au

Title	First Name	Last Name	Address	Phone	Email
MS	Emma	Russell	4 WESTMILL DR, HOPPERS CROSSING VIC 3029	0427519643	

Request body

The details of the person to trace and any known contact information.

Field	Description
`name_combined`	**string** The full name to search for as a single string (e.g. "MARY SALLY JONES") **Example:** `MARY JONES`
`name_first`	**string** First name of the person to trace **Example:** `MARY`
`name_middle`	**string** Middle name of the person to trace **Example:** `SALLY`
`name_last`	**string** Last (family) name of the person to trace **Example:** `JONES`
`dob`	**string (date)** Date of birth of the person to trace (YYYY-MM-DD) **Example:** `1989-08-12`
`phone1`	**string** Known phone number of the person to trace **Example:** `0299995000`
`phone2`	**string** Known phone number of the person to trace
`phone3`	**string** Known phone number of the person to trace

Field	Description
`phone4`	**string** Known phone number of the person to trace
`email1`	**string (email)** Known email address of the person to trace **Example:** `mary_jones89@example.com`
`email2`	**string (email)** Known email address of the person to trace
`email3`	**string (email)** Known email address of the person to trace
`email4`	**string (email)** Known email address of the person to trace
`street_address`	**string** First line of the street address **Example:** `35 YARALLA ST`
`suburb`	**string** Suburb of the address **Example:** `CONCORD WEST`
`state`	**string** Australian state of the address **Enum:** `ACT`, `NSW`, `NT`, `QLD`, `SA`, `TAS`, `VIC`, `WA` **Example:** `NSW`
`postcode`	**string** Postcode of the address **Example:** `2138`
`suburb_state_postcode`	**string** A single string containing the suburb, state and postcode (address line 2) **Example:** `CONCORD WEST NSW 2138`

Field	Description
`full_address`	**string** A single string containing the complete address **Example:** `35 YARALLA ST CONCORD WEST NSW 2138`
`option_relatives`	**boolean** Include possible relatives in the trace results **Default:** `false`
`first_result`	**integer [min 0]** Index of first result to return for pagination (0-based) **Default:** `0` **Example:** `0`
`max_results`	**integer [1..30]** Maximum number of results to return (1-30) **Default:** `10` **Example:** `10`
`sort_by`	**string** Field to sort results by **Enum:** `score` , `name` , `dob` , `phone` , `email` , `state` , `suburb` , `postcode` , `street_address` , `address_combined` , `updated` **Example:** `score`
`sort_order`	**string** Sort direction **Enum:** `ASC` , `DESC` **Example:** `DESC`

Sample request

```
{
    "name_combined": "MARY JONES",
    "name_first": "MARY",
    "name_middle": "SALLY",
    "name_last": "JONES",
    "dob": "1989-08-12",
    "phone1": "0299995000",
    "phone2": "string",
```

```
    "phone3": "string",
    "phone4": "string",
    "email1": "mary_jones89@example.com",
    "email2": "user@example.com",
    "email3": "user@example.com",
    "email4": "user@example.com",
    "street_address": "35 YARALLA ST",
    "suburb": "CONCORD WEST",
    "state": "NSW",
    "postcode": "2138",
    "suburb_state_postcode": "CONCORD WEST NSW 2138",
    "full_address": "35 YARALLA ST CONCORD WEST NSW 2138",
    "option_relatives": true,
    "first_result": 0,
    "max_results": 10,
    "sort_by": "score",
    "sort_order": "DESC"
}
```


Responses

200 **Successful autotrace response with matching person records ranked by score.** (application/json)

Field	Description
`message`	**string** A message indicating the result of the request: • `Ok` - Trace was successful • `No match` - No records found for this trace • `Found too many results` - The trace found too many results and was unable to provide useful scoring. Try providing more specific search criteria. **Example:** `Ok`
`function`	**string** The API function that handled the request. **Example:** `caspar_person_autotrace`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`records`	**array of objects** Array of matching person records, ordered by score (highest first) unless a different sort is specified.

Field	Description
records[]. title	**string** Title of the matched person **Example:** `MS`
records[]. first_name	**string** First name of the matched person **Example:** `MARY`
records[]. middle_name	**string** Middle name of the matched person **Example:** `SALLY`
records[]. last_name	**string** Last name of the matched person **Example:** `JONES`
records[]. dob	**string** Date of birth (year only, format YYYY) if available **Example:** `1989`
records[]. dob_range_from	**string** DOB range minimum (year only, format YYYY) if no exact DOB is available **Example:**
records[]. dob_range_to	**string** DOB range maximum (year only, format YYYY) if no exact DOB is available **Example:**
records[]. deceased	**string** Deceased flag **Enum:** `Y`, `N` **Example:** `N`
records[]. deceased_date	**string** Deceased date (year only, format YYYY) if available **Example:**

Field	Description
records[]. gender	**string** Gender of the matched person **Enum:** `MALE`, `FEMALE`, `X`, `UNKNOWN` **Example:** `FEMALE`
records[]. phones	**array of strings** Array of phone numbers associated with the matched person
records[]. emails	**array of strings** Array of email addresses associated with the matched person
records[]. socials	**array of objects** Social media accounts associated with the matched person (if available)
records[]. ip_addresses	**array of strings** IP addresses associated with the matched person (if available)
records[]. connectivity	**object** Cached connectivity check results for any phones and emails in the result set. Each entry is keyed by the phone number or email address itself. Arrays may be empty when no cached connectivity data is available. Real-time connectivity can be obtained via the `email_ping` and `phone_ping` endpoints.
records[]. connectivity. phones	**object (dynamic)** Cached phone ping results keyed by phone number. Only populated when a cached ping result exists for a phone number in the result set.
records[]. connectivity. emails	**object (dynamic)** Cached email ping results keyed by email address. Only populated when a cached ping result exists for an email address in the result set.
records[]. address_id	**string** Address persistent identifier **Example:** `GANSW716615055`
records[]. street_address	**string** The street/postal address as one string **Example:** `35 YARALLA ST`

Field	Description
records[]. suburb	**string** Suburb of the matched address **Example:** `CONCORD WEST`
records[]. state	**string** State of the matched address **Example:** `NSW`
records[]. postcode	**string** Postcode of the matched address **Example:** `2138`
records[]. address_parsed	**string** Whether the address passed the address parser (Y/N) **Enum:** `Y`, `N` **Example:** `Y`
records[]. address_valid	**string** Whether the address exists in the address table (Y/N) **Enum:** `Y`, `N` **Example:** `Y`
records[]. address_primary_secondary	**string** Whether address is a primary or secondary dwelling (P/S or blank) **Example:**
records[]. date_start	**string** Date the person was first seen at this address (YYYY-MM) **Example:** `2016-06`
records[]. date_end	**string** Date the person was last seen at this address (YYYY-MM) **Example:** `2019-07`

Field	Description
records[]. legal_parcel_id	**string** Legal parcel identifier for the address property **Example:** `15//DP240258`
records[]. meshblock_category	**string** ABS meshblock category for the address (e.g. Residential, Commercial, Parkland) **Example:** `Residential`
records[]. latitude	**string** Latitude coordinate of the address **Example:** `-33.72914350`
records[]. longitude	**string** Longitude coordinate of the address **Example:** `151.21036778`
records[]. score	**string** A unitless integer match score. Higher scores indicate a better match from the autotrace process. The score is based on multiple parameters including the distance between the supplied address and the matched address. **Example:** `1000`
records[]. match_type	**string** Relationship indicator to the searched record: - `primary` - Direct match against the primary record - `relative` - Match is a possible relative of the primary record **Enum:** `primary`, `relative` **Example:** `primary`
records[]. realestate	**array of objects** Real estate events for this address
records[]. realestate[]. listing_type	**string** Type of listing: sale, sold, or rent **Enum:** `sale`, `sold`, `rent`

Field	Description
records[]. realestate[]. property_type	**string** Type of property (e.g. House, Apartment, Unit)
records[]. realestate[]. num_bedrooms	**string** Number of bedrooms (if available)
records[]. realestate[]. num_bathrooms	**string** Number of bathrooms (if available)
records[]. realestate[]. num_car_spaces	**string** Number of car spaces (if available)
records[]. realestate[]. date	**string (date)** Date of the listing (YYYY-MM-DD)
records[]. realestate[]. estate_agent	**string** Name of the estate agent
records[]. realestate[]. price	**string** Property price (where available)
records[]. judgements	**array of objects** Judgements or events for the person at this address
records[]. judgements[]. name	**string** The name for which the judgement is recorded
records[]. judgements[]. creditor	**string** Name of the creditor
records[]. judgements[]. event	**string** Description of the event
records[]. judgements[]. date	**string (date)** Date of listing (YYYY-MM-DD)
records[]. business	**array of objects** ABN records possibly associated with the person at the matched postcode

Field	Description
records[]. business[]. abn	**string** ABN of the matching entity **Example:** `12345678901`
records[]. business[]. status	**string** ABN status: ACT (Active) or CAN (Cancelled) **Enum:** `ACT`, `CAN`
records[]. business[]. effective_from	**string (date)** Date the status became effective (YYYY-MM-DD)
records[]. address_info	**array of objects** Additional information about the address
records[]. address_info[]. property_type	**string** Type of property: agedcare, prison, or emg_accom **Enum:** `agedcare`, `prison`, `emg_accom`
records[]. address_info[]. property_detail	**string** Description of the property
records[]. demographic	**object** SEIFA geo-demographic indices for the address area, based on SA1 level statistical analysis. Decile ranked from 0 to 10, where 10 indicates the top 10% of areas for that indicator and 0 means no data is available.
records[]. demographic. IRSAD_decile	**string** Index of Relative Socio-economic Advantage and Disadvantage (0-10). Focuses on financial aspects related to buying power, income and wealth. A low score indicates relative disadvantage (e.g. low income households); a high score indicates relative advantage (e.g. high income, home ownership). **Example:** `9`
records[]. demographic. IER_decile	**string** Index of Economic Resources (0-10). Focuses on financial aspects related to buying power, income and wealth. A low score indicates a relative lack of economic resources (e.g. low income, low rent); a high score indicates greater access to economic resources (e.g. high income, home ownership). **Example:** `8`

Field	Description
records[]. demographic. IEO_decile	**string** Index of Education and Occupation (0-10). Focuses on the educational and occupational level of the area. A low score indicates lower education and occupation status (e.g. no qualifications, low skilled or unemployed); a high score indicates higher education and occupation status (e.g. higher qualifications, highly skilled occupations). **Example:** `7`
records[]. court_record	**string or null** Whether a court record exists for this person (Y/N or null if not checked) **Enum:** `Y`, `N` **Example:** `N`
records[]. social_record	**string or null** Whether a social media record exists for this person (Y/N or null if not checked) **Enum:** `Y`, `N` **Example:** `N`
records[]. employment_record	**string or null** Whether an employment record exists for this person (Y/N or null if not checked) **Enum:** `Y`, `N` **Example:** `N`
records[]. sources	**array of strings** Data sources that contributed to this record (e.g. government, proprietary_records, competition). May be empty.
total_records_available	**integer** Total number of matching records available. **Example:** `4`

Sample response


```
{
    "message": "Ok",
    "function": "caspar_person_autotrace",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "records": [
        {
            "title": "MS",
            "first_name": "MARY",
            "middle_name": "SALLY",
```

```
            "last_name": "JONES",
            "dob": "1989",
            "dob_range_from": "",
            "dob_range_to": "",
            "deceased": "N",
            "deceased_date": "",
            "gender": "FEMALE",
            "phones": [
                "0299995000",
                "0491999888"
            ],
            "emails": [
                "mary_jones89@example.com"
            ],
            "socials": [],
            "ip_addresses": [
                "string"
            ],
            "connectivity": {
                "phones": {
                    "0299995000": {
                        "check_date": "2020-03-23",
                        "status": "connected",
                        "carrier": "",
                        "geo_area": "Sydney",
                        "dnc_result": "",
                        "dnc_check_date": "",
                        "dnc_reference": ""
                    },
                    "0491999888": {
                        "check_date": "2020-08-04",
                        "status": "disconnected",
                        "carrier": "Telstra",
                        "geo_area": "",
                        "dnc_result": "N",
                        "dnc_check_date": "2020-08-04",
                        "dnc_reference": ""
                    }
                },
                "emails": {
                    "mary_jones89@example.com": {
                        "check_date": "2020-08-04",
                        "status": "undeliverable"
                    }
                }
            },
            "address_id": "GANSW716615055",
            "street_address": "35 YARALLA ST",
            "suburb": "CONCORD WEST",
            "state": "NSW",
            "postcode": "2138",
            "address_parsed": "Y",
            "address_valid": "Y",
            "address_primary_secondary": "",
            "date_start": "2016-06",
            "date_end": "2019-07",
            "legal_parcel_id": "15//DP240258",
            "meshblock_category": "Residential",
            "latitude": "-33.72914350",
            "longitude": "151.21036778",
            "score": "1000",
            "match_type": "primary",
```

```
            "realestate": [
                {
                    "listing_type": "sale",
                    "property_type": "string",
                    "num_bedrooms": "string",
                    "num_bathrooms": "string",
                    "num_car_spaces": "string",
                    "date": "2024-01-01",
                    "estate_agent": "string",
                    "price": "string"
                }
            ],
            "judgements": [
                {
                    "name": "string",
                    "creditor": "string",
                    "event": "string",
                    "date": "2024-01-01"
                }
            ],
            "business": [
                {
                    "abn": "12345678901",
                    "status": "ACT",
                    "effective_from": "2024-01-01"
                }
            ],
            "address_info": [
                {
                    "property_type": "agedcare",
                    "property_detail": "string"
                }
            ],
            "demographic": {
                "IRSAD_decile": "9",
                "IER_decile": "8",
                "IEO_decile": "7"
            },
            "court_record": "N",
            "social_record": "N",
            "employment_record": "N",
            "sources": [
                "government",
                "proprietary_records"
            ]
        }
    ],
    "total_records_available": 4
}
```


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

ASIC Company Search

POST `/asic_company_search`

`asic_company_search` locates the ASIC identifiers you must supply when ordering a company risk report via `/company_risk`. The endpoint accepts any of the following search modes and automatically determines which one to use based on the `search` value:

Search mode	Input format	Typical use
ABN search	11 digits (spacing ignored)	When you already know the company's ABN
ACN search	9 digits (spacing ignored)	When you have the ACN / NNI
Name search	Free?form string matched against ASIC extract names	When you only know the legal company name

Behaviour

- Number searches return a single match (or an empty list) containing the ASIC record retrieved.
- Name searches can return multiple organisations. Use the optional `status` filter and `max_results` to keep the result set manageable. If the ASIC upstream indicates there are more results than returned, `more_results` is set to `true`.
- Copy either the `identifier.number` (usually an ACN) or the `abn` from a match when submitting a company extract request.

Status filters

When running a name search you can restrict the response to companies in a particular ASIC registration status:

status value	Meaning
`registered`	Only active/registered companies
`deregistered`	Only deregistered companies
`all` (default)	Return both registered and deregistered companies

Status filters are ignored for ABN / ACN lookups because those queries already return a single company.

Location filters

When running a name search you can restrict the response to a postcode or state. Results will only be included if they have a registered office address in the specified location. Postcode searches will include neighbouring postcodes.

If both a postcode and state are provided, only the postcode is applied.

Location filters are ignored for ABN / ACN lookups because those queries already return a single company.

Sandbox environment

The sandbox dataset is deterministic. Use one of the following sample organisations to receive predictable matches:

Company Name	ABN	ACN	Registration Status
Copper Finch Construction Pty Ltd	94782610486	782610486	Registered
Marble Kookaburra Interiors Pty Ltd	84655375652	655375652	Registered

Request body

Parameters describing the company search.

Field	Description
`search` REQUIRED	**string [max 100 characters]** Company name, ABN, or ACN to look up. Letter casing is ignored for name searches; whitespace is ignored for number searches. **Example:** `ACME CORPORATION PTY LTD`
`max_results`	**integer [1..90] or null** Maximum number of matches to return for name searches (1-90). Defaults to 20. Ignored for ABN/ACN lookups because those return at most one record. **Example:** `25`
`status`	**string or null** Optional name-search filter that limits responses to registered, deregistered, or all companies. If not supplied the API searches all companies. **Enum:** `registered`, `deregistered`, `all` **Example:** `registered`
`postcode`	**string [max 4 characters] or null** Optional name-search filter that limits responses to companies with a registered office in the specified postcode or neighbouring postcodes. Ignored for ABN/ACN lookups.
`state`	**string or null** Optional name-search filter that limits responses to companies with a registered office in the specified state. Use the two/three-letter state code (e.g. `VIC`, `NSW`). Ignored for ABN/ACN lookups.

Sample request

```
{
    "search": "ACME CORPORATION PTY LTD",
    "max_results": 25,
```

```
    "status": "registered",
    "postcode": "string",
    "state": "string"
}
```

Responses

200 **Company search completed successfully.** (application/json)

Field	Description
`message`	**string** Always `Ok` on success. **Example:** `Ok`
`function`	**string** The API function that handled the request. **Example:** `asic_company_search`
`api_reference`	**string (uuid)** Audit reference for the request. Use `/asic_extract/{api_reference}` to retrieve audit logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`matches`	**array of objects** A list of companies that match the search criteria. Each object contains the metadata you will need when calling `/asic_extract` (typically an ACN or ABN) plus additional ASIC attributes when available.
`matches[].identifier`	**object** Primary ASIC identifier (usually an ACN) returned during name searches.
`matches[].identifier.numberHeading`	**string** ASIC identifier type (`ACN` , `ABN` , etc.). **Example:** `ACN`
`matches[].identifier.number`	**integer** Identifier value for the company. **Example:** `123456780`
`matches[].name`	**object** Structured ASIC name block.

Field	Description
matches[].name.name	**string** Uppercase legal name of the organisation. **Example:** `ACME CORPORATION PTY LTD`
more_results	**boolean** Indicates that ASIC reported more matches than were returned (for example when `max_results` was reached). Refine your search and try again if this is `true`. **Example:** `false`

Sample response


```
{
    "message": "Ok",
    "function": "asic_company_search",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "matches": [
        {
            "identifier": {
                "numberHeading": "ACN",
                "number": 123456780
            },
            "name": {
                "name": "ACME CORPORATION PTY LTD"
            },
            "status": {
                "code": "REGD,",
                "description": "Registered",
                "isRegistered": true,
                "effectiveFrom": "2020-01-15"
            }
        },
        {
            "identifier": {
                "numberHeading": "ACN",
                "number": 987654321
            },
            "name": {
                "name": "ACME CORPORATION PTY LTD"
            },
            "status": {
                "code": "DRGD,",
                "description": "Deregistered,",
                "isRegistered": false
            }
        }
    ],
    "more_results": false
}
```


Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

Company Risk Report

POST `/company_risk`

Requests a company risk report using an ABN or ACN. The response confirms the job was queued and returns an `api_reference`. Poll `GET /company_risk/{api_reference}` until the report payload is ready.

Request rules

- Provide **either** `abn` or `acn` (not both).
- The report is generated in the background. Use the `api_reference` to retrieve it.

Sandbox dataset

Use the following sandbox companies to receive deterministic responses:

Company Name	ABN	ACN	Notes
Copper Finch Construction Pty Ltd	94782610486	782610486	Risk report returns immediately
Marble Kookaburra Interiors Pty Ltd	84655375652	655375652	Risk report may require additional polling

Request body

Parameters describing the company to report on.

Field	Description
`abn`	string or null Required when `acn` is not supplied. Provide 11 digits (whitespace ignored). **Example:** `94782610486`
`acn`	string or null Required when `abn` is not supplied. Provide 9 digits (whitespace ignored). **Example:** `782610486`

Sample request

```
{
    "abn": "94782610486",
    "acn": "782610486"
}
```

Responses

200 **Report request accepted.** (application/json)

Field	Description

`message`	**string** Always `Ok` on success. **Example:** `Ok`
`function`	**string** The API function that handled the request. **Example:** `company_risk`
`api_reference`	**string (uuid)** Audit reference for this report request. Poll `/company_risk/{api_reference}` for results. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Sample response

```
{
    "message": "Ok",
    "function": "company_risk",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95"
}
```

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

Retrieve Company Risk Report

GET `/company_risk/{api_reference}`

Polls the status of a company risk report that was queued via `POST /company_risk`. Provide the `api_reference` returned from the original request to determine whether the report is ready, download the JSON payload, or stream a PDF copy. This retrieval call is not billable and can be repeated until the report is ready.

- While the report is still compiling the API responds with HTTP `202`. Keep polling with an exponential backoff.
- Once ready, the endpoint returns HTTP `200` with the full JSON report plus the `company_risk_show` `api_reference` for auditing.
- Append `?pdf=true` to stream a formatted PDF copy of the same report. When `pdf=true`, the response body is a PDF (`application/pdf`) instead of JSON.
- The service polls upstream data for up to 30 seconds; if no report is available at that point you will receive a `202` response. Continue polling with the same `api_reference`.
- A report can only be retrieved for 24 hours after the original `POST /company_risk` request. Once that window has passed the endpoint responds with HTTP `410` and you need to submit a new report request.

Sandbox usage

Use the sandbox company identifiers listed under `POST /company_risk` to create a request, then poll this endpoint with the returned `api_reference`. Marble Kookaburra Interiors Pty Ltd (`ABN 84655375652` , `ACN 655375652`) is configured to return

a `202` on the first poll so you can test retry logic.

Path parameters

Field	Description
`api_reference` REQUIRED	**string (uuid)** The audit reference returned when the report was queued via `POST /company_risk`. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Query parameters

Field	Description
`pdf`	**boolean** When set to `true`, returns the report as a PDF document instead of JSON. Omit (or send `false`) to receive the JSON payload. **Default:** `false`

Responses

200 **Report is ready. Returns JSON by default or a PDF when `pdf=true`.** (application/json, application/pdf)

Field	Description
`message`	**string** Always `Ok` when the report payload is returned. **Example:** `Ok`
`function`	**string** Name of the API function that handled the request. **Example:** `company_risk_show`
`api_reference`	**string (uuid)** Audit reference for **this** retrieval call (not the original queue reference). **Example:** `7cbb094c-0f27-42f4-9f36-9cf0b2f6d5ff`
`report`	**object (dynamic)** Full company risk report payload.

Sample response

```
{
    "message": "Ok",
```

```
    "function": "company_risk_show",
    "api_reference": "7cbb094c-0f27-42f4-9f36-9cf0b2f6d5ff",
    "report": {
        "id": "7e6b9b1f-3f02-4c8c-8c45-5b1a9a2ce222",
        "entity_type": "company",
        "entity": {
            "abn": "84655375652",
            "acn": "655375652",
            "name": "MARBLE KOOKABURRA INTERIORS PTY LTD"
        },
        "cases": [
            {
                "case_number": "C-2025-0918",
                "case_type": "Goods and services",
                "court_name": "VIC CAT"
            }
        ]
    }
}
```


Also available as a PDF (application/pdf)

202 **Report is still being prepared (returned when no data is available after the 30?second polling window).**
(application/json)

Field	Description
`message`	**string** Status message indicating the report is not ready yet. **Example:** `Report is still being processed. Please try again later.`

Sample response


```
{
    "message": "Report is still being processed. Please try again later."
}
```


404 **No report was found for the provided `api_reference`, or it belongs to another account/environment.**
(application/json)

Field	Description
`message`	**string** Error message describing that the record could not be located. **Example:** `Not Found.`

Sample response


```
{
    "message": "Not Found."
}
```

410 **The report can no longer be retrieved. Reports are only available for retrieval for 24 hours after the original `POST /company_risk` request. Submit a new report request to obtain a current report.** (application/json)

Field	Description
`message`	**string** Error message describing that the report has expired. **Example:** `This report has expired. Please request a new one`

Sample response

```
{
    "message": "This report has expired. Please request a new one"
}
```

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

Court Check

POST `/court_check`

Searches court records against a individual's or company's name and state.

Searches can be made against civil records, criminal records, or both.

There are two types of searches:

- Summary - This returns the number of records and type that can be found against the individual or company.
- Detailed - This returns the details of the records that can be found against the individual or company.

Sandbox environment

When performing a court check in the sandbox environment, the following record will return a match:

Name Searches:

First Name	Last Name	State
Michael	Raymond	VIC

Company Searches:

Company	State
Samplemart	VIC

These samples can be queried with criminal, civil or both record listings.

Request body

The details of the entity to search court records for.

Field	Description
`type`	string The type of entity to search for. **Enum:** `name`, `company` **Example:** `name`

Field	Description
`report`	**string** The type of report to generate. **Enum:** `summary`, `detail` **Example:** `summary`
`listing`	**string** The type of court records to search for. **Enum:** `civil`, `criminal`, `all` **Example:** `civil`
`name_first`	**string** The first name of the individual when performing a name search. This should not be used in conjunction with name_full in detailed reports. If name_full is set, then name_first will be ignored in detailed reports. **Example:** `Michael`
`name_middle`	**string** The middle names of the individual when performing a name search. This should not be used in conjunction with name_full in detailed reports. If name_full is set, then name_middle will be ignored in detailed reports.
`name_last`	**string** The last name of the individual when performing a name search. This should not be used in conjunction with name_full in detailed reports. If name_full is set, then name_last will be ignored in detailed reports. **Example:** `Raymond`
`name_full`	**string** The full name of the individual when performing a name search in detail reports. This is used to search for the individual in the court records. It should be in the form "Last name, First name Middle name". Eg: "DOE, JOHN JAMES" If you would rather supply the name in its natural order and have it split for you, use name_full_unstructured instead. This field should not be used in summary reports.

Field	Description
`name_full_unstructured`	**string [max 80 characters]** The full name of the individual in its natural order, "First name Middle name Last name". Eg: "John James Doe" The API makes a best effort attempt to split this into the first, middle and last name for you, so you do not have to format the name yourself the way name_full requires. This can only be used in detail reports for name searches. Using it in any other type of report will return an error. This field should not be used in conjunction with name_full, name_first, name_middle or name_last. Doing so will return an error. If the name cannot be split into at least a first and a last name, the value you supplied is used as name_full instead. The usual name requirements for a detail report still apply to it, so a value that is not a usable name will still be rejected. **Example:** `Michael Raymond`
`company`	**string** The name of the company when performing a company search.
`state`	**string** The state to search in. If the state is omitted, then all states will be searched. **Enum:** `NSW`, `VIC`, `QLD`, `WA`, `SA`, `TAS`, `ACT`, `NT` **Example:** `VIC`
`max_results`	**integer** The maximum number of results to return. **Example:** `10`
`first_result`	**integer** Specifies the first result to be returned in the list of records. This is useful when displaying results in a paginated manner. **Example:** `0`

Sample request

```
{
    "type": "name",
    "report": "summary",
    "listing": "civil",
    "name_first": "Michael",
    "name_middle": "string",
    "name_last": "Raymond",
    "name_full": "string",
    "name_full_unstructured": "Michael Raymond",
    "company": "string",
    "state": "VIC",
    "max_results": 10,
```

```
    "first_result": 0
}
```

Responses

200 **The result of the court check.** (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`
`function`	**string** The function that was called. **Example:** `court_check`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`records`	**array of objects** An array of the court records that were found. This will be an empty list if no records are found.
`total_records_available`	**integer** The total number of records that are available for the search, regardless of the specified max_results. **Example:** `1`

Sample response

```
{
    "message": "Ok",
    "function": "court_check",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "records": [
        {
            "full_name": "RAYMOND, MICHAEL",
            "total": 2,
            "listing": "civil",
            "type": "individual"
        }
    ],
    "total_records_available": 1
}
```

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

Court Check Multiple

POST `/court_check_multiple`

This is a bulk version of the court check endpoint that allows multiple checks to be performed in a single request.

This endpoint accepts up to 20 requests in the `requests` array and applies shared settings (`report` , `listing` , pagination and optional `start_date`) to each item.

The types of searches that can be performed are the same as the single court check endpoint:

- Summary - This returns the number of records and type that can be found against the individual or company.
- Detailed - This returns the details of the records that can be found against the individual or company.

Sandbox environment

When performing a court check in the sandbox environment, the following record will return a match:

Name Searches:

First Name	Last Name	State
Michael	Raymond	VIC

Company Searches:

Company	State
Samplemart	VIC

These samples can be queried with criminal, civil or both record listings.

Request body

The details of the entities to search for.

Field	Description
`report`	**string** The type of report to generate. **Enum:** `summary` , `detail` **Example:** `summary`

Field	Description
`listing`	**string** The type of court records to search for. **Enum:** `civil`, `criminal`, `all` **Example:** `all`
`max_results`	**integer [1..50]** The maximum number of results to return. **Example:** `10`
`first_result`	**integer** Specifies the first result to be returned. **Example:** `1`
`start_date`	**string** Optional earliest court date in `Y-m-d` format. **Example:** `2024-01-01`
`requests`	**array of objects [1..20 items]** The list of court checks to perform.
`requests[].` `check_id`	**string** Optional client reference returned in the matching result item. **Example:** `check_123`
`requests[].` `type`	**string** The type of entity to search for. **Enum:** `name`, `company` **Example:** `name`
`requests[].` `name_first`	**string [max 50 characters]** Required for `type=name` with `report=summary`. **Example:** `Michael`
`requests[].` `name_middle`	**string [max 50 characters]** Optional middle name for `type=name`. An empty string will filter results to those with no middle name, while omitting this field or null will not filter based on middle name. **Example:**

Field	Description
requests[]. name_last	**string [max 50 characters]** Required for `type=name` with `report=summary`. **Example:** `Raymond`
requests[]. name_full	**string [max 50 characters]** Required for `type=name` with `report=detail`, unless `name_full_unstructured` is supplied instead. Must be formatted as "Last name, First name Middle name". **Example:** `RAYMOND, MICHAEL`
requests[]. name_full_unstructured	**string [max 50 characters]** An alternative to `name_full` for `type=name` with `report=detail`. Accepts the name in its natural order, "First name Middle name Last name", and makes a best effort attempt to split it into the first, middle and last name for you. Cannot be used together with `name_full`, `name_first`, `name_middle` or `name_last`, and cannot be used with `report=summary` or `type=company`. If the name cannot be split into at least a first and a last name, the value you supplied is used as `name_full` instead, and the usual name requirements for a detail report still apply to it. **Example:** `Michael Raymond`
requests[]. company	**string [max 512 characters]** Required for `type=company`. **Example:** `Acme Services Pty Ltd`
requests[]. state	**string** Optional state scope for the request item. **Enum:** `NSW`, `VIC`, `QLD`, `WA`, `SA`, `TAS`, `ACT`, `NT`, `FED` **Example:** `NSW`

Sample request

```
{
    "report": "summary",
    "listing": "all",
    "max_results": 10,
    "first_result": 1,
    "start_date": "2024-01-01",
    "requests": [
        {
            "check_id": "check_123",
            "type": "name",
            "name_first": "Michael",
            "name_middle": "",
            "name_last": "Raymond",
```

```
            "name_full": "RAYMOND, MICHAEL",
            "name_full_unstructured": "Michael Raymond",
            "company": "Acme Services Pty Ltd",
            "state": "NSW"
        }
    ]
}
```

Responses

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

Field	Description
`message`	**string** A message indicating the result of the request. **Example:** `Ok`
`function`	**string** The function that was called. **Example:** `court_check_multiple`
`api_reference`	**string (uuid)** A unique identifier for this request. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`results`	**array of objects** Result item for each request object.
`results[].` `check_id`	**string or null** Echoed check id from the request item. **Example:** `check_123`
`results[].` `records`	**array of objects** The returned court records for this request item.
`results[].` `total_records_available`	**integer** Total records available for this request item. **Example:** `0`
`results[].` `error`	**string** Present when an item-level error occurs.

Sample response

```
{
    "message": "Ok",
    "function": "court_check_multiple",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "results": [
        {
            "check_id": "check_123",
            "records": [],
            "total_records_available": 0,
            "error": "string"
        }
    ]
}
```


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

ADC Check

POST `/adc_check`

The ADC Check API allows you to check if an individual is recorded as deceased against the ADC (Australian Death Check) service.

> **Note:** If you are looking for **ADC Check (Custodial)**, please refer to the Deceased Check endpoint.

ADC Check

The returned records are checked against the ADC (Australian Death Check) service and any matching records are returned in the 'matches' field of the response. Note that the ADC check is performed based on the name and date of birth of the individual. The results will also include the state of the individual, and this should be taken into account when determining if the result is a match.

When searching a name, the first and last names must match exactly. The optional middle name(s) can be partial matches. It is recommended that the full name (including middle names) are used when searching for a name.

Examples:

Assuming the ADC has the following record: Name: John Henry Smith

Different search names and the expected matched name result:

Searched First Name	Searched Middle Name	Searched Last Name	Matched First Name	Matched Last Name	name_match
John	Henry	Smith	John Henry	Smith	exact
John	Henry Peter	Smith	John Henry	Smith	partial
John	Peter Henry	Smith	John Henry	Smith	partial
John	Peter	Smith	John Henry	Smith	partial
John		Smith	John Henry	Smith	partial

Sandbox environment data

When simulating queries in the sandbox environment, the following records will return a match:

Reference	First Name	Last Name	State	Date of Birth	Date of Death	Date of Death Range
123400001	John	Smith	VIC	1998-03-21	2023-07-24	
123400002	John	Doe	NSW	1971-01-23	2003-08-22	

Reference	First Name	Last Name	State	Date of Birth	Date of Death	Date of Death Range
123400003	John Andrew	Doe	WA	1971-01-23	2011-11-03	
123400004	Mary	Smith	NSW	1965-02-17		Between 31/12/2018 and 2/2/2019

Request body

The details of the record to search

Field	Description
`first_name`	**string** The first name of the individual **Example:** `John`
`middle_name`	**string or null** The optional middle name of the individual **Example:** `Andrew`
`last_name`	**string** The last name of the individual **Example:** `Smith`
`birth_date`	**string (date)** The date of birth of the individual **Example:** `1998-03-21`

Sample request

```
{
    "first_name": "John",
    "middle_name": "Andrew",
    "last_name": "Smith",
    "birth_date": "1998-03-21"
}
```

Responses

200 **Successful 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. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`matches`	**array of objects** Array of matched records.
`matches[].` `first_name`	**string** The first name (and middle name if applicable) of the matched individual. **Example:** `John Andrew`
`matches[].` `last_name`	**string** The last name of the matched individual. **Example:** `Smith`
`matches[].` `date_of_birth`	**string or null** The dob of the matched individual. **Example:** `1998-03-21`
`matches[].` `date_of_death`	**string or null** The date of death of the matched individual. **Example:** `2023-07-24`
`matches[].` `date_of_death_range`	**string or null** The date range of death of the matched individual. **Example:** `Between 31/12/2018 and 2/2/2019`
`matches[].` `state`	**string** The state of the matched individual. **Enum:** `ACT`, `NSW`, `NT`, `QLD`, `SA`, `TAS`, `VIC`, `WA` **Example:** `VIC`

matches[]. reference	**string** The ADC reference number of the matched record. **Example:** `123400001`
matches[]. name_match	**string** The type of match for the name. - exact: The name is an exact match. - partial: The name is a partial match (The last name is the same but the first / middle name differ). **Enum:** `exact`, `partial` **Example:** `exact`

Sample response

```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "matches": [
        {
            "first_name": "John Andrew",
            "last_name": "Smith",
            "date_of_birth": "1998-03-21",
            "date_of_death": "2023-07-24",
            "date_of_death_range": "Between 31/12/2018 and 2/2/2019",
            "state": "VIC",
            "reference": "123400001",
            "name_match": "exact"
        }
    ]
}
```

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

ADC Check Bulk

POST `/adc_checks`

The ADC Check Bulk API allows you to check if an individual is recorded as deceased against the ADC (Australian Death Check) service.

This endpoint allows multiple requests to be simultaneously made in a single query by providing a list of requests in the requests array.

ADC Check

The returned records are checked against the ADC (Australian Death Check) service and any matching records are returned in the 'matches' field of the response. Note that the ADC check is performed based on the name and date of birth of the individual. The results will also include the state of the individual, and this should be taken into account when determining if the result is a

match.

When searching a name, the first and last names must match exactly. The optional middle name(s) can be partial matches. It is recommended that the full name (including middle names) are used when searching for a name.

Examples:

Assuming the ADC has the following record: Name: John Henry Smith

Different search names and the expected matched name result:

Searched First Name	Searched Middle Name	Searched Last Name	Matched First Name	Matched Last Name	name_match
John	Henry	Smith	John Henry	Smith	exact
John	Henry Peter	Smith	John Henry	Smith	partial
John	Peter Henry	Smith	John Henry	Smith	partial
John	Peter	Smith	John Henry	Smith	partial
John		Smith	John Henry	Smith	partial

Sandbox environment data

When simulating queries in the sandbox environment, the following records will return a match:

Reference	First Name	Last Name	State	Date of Birth	Date of Death	Date of Death Range
123400001	John	Smith	VIC	1998-03-21	2023-07-24	
123400002	John	Doe	NSW	1971-01-23	2003-08-22	
123400003	John Andrew	Doe	WA	1971-01-23	2011-11-03	
123400004	Mary	Smith	NSW	1965-02-17		Between 31/12/2018 and 2/2/2019

Request body

The details of the record to search

Field	Description
`requests`	**array of objects** The list of requests to make.
`requests[].` `check_id`	**string** The unique identifier for this request. This can be used match the request in the results. **Example:** `abcd123456789`

Field	Description
requests[]. first_name	**string** The first name of the individual **Example:** `John`
requests[]. middle_name	**string or null** The optional middle name of the individual **Example:** `Andrew`
requests[]. last_name	**string** The last name of the individual **Example:** `Smith`
requests[]. birth_date	**string (date)** The date of birth of the individual **Example:** `1998-03-21`

Sample request

```
{
    "requests": [
        {
            "check_id": "abcd123456789",
            "first_name": "John",
            "middle_name": "Andrew",
            "last_name": "Smith",
            "birth_date": "1998-03-21"
        }
    ]
}
```

Responses

200 **Successful 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`

Field	Description
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`results`	**array of objects** A list of results for each of the requests made.
`results[].` `check_id`	**string** The unique identifier for this request. This can be used match the request in the results. **Example:** `abcd123456789`
`results[].` `matches`	**array of objects** Array of matched records.

`results[].`
`matches[].`
`first_name`
	string
	
	The first name (and middle name if applicable) of the matched individual.
	
	Example: `John Andrew`

`results[].`
`matches[].`
`last_name`
	string
	
	The last name of the matched individual.
	
	Example: `Smith`

`results[].`
`matches[].`
`date_of_birth`
	string or null
	
	The dob of the matched individual.
	
	Example: `1998-03-21`

`results[].`
`matches[].`
`date_of_death`
	string or null
	
	The date of death of the matched individual.
	
	Example: `2023-07-24`

`results[].`
`matches[].`
`date_of_death_range`
	string or null
	
	The date range of death of the matched individual.
	
	Example: `Between 31/12/2018 and 2/2/2019`

Field	Description
results[]. matches[]. state	**string** The state of the matched individual. **Enum:** `ACT` , `NSW` , `NT` , `QLD` , `SA` , `TAS` , `VIC` , `WA` **Example:** `VIC`
results[]. matches[]. reference	**string** The ADC reference number of the matched record. **Example:** `123400001`
results[]. matches[]. name_match	**string** The type of match for the name. - exact: The name is an exact match. - partial: The name is a partial match (The last name is the same but the first / middle name differ). **Enum:** `exact` , `partial`

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "results": [
        {
            "check_id": "abcd123456789",
            "matches": [
                {
                    "first_name": "John Andrew",
                    "last_name": "Smith",
                    "date_of_birth": "1998-03-21",
                    "date_of_death": "2023-07-24",
                    "date_of_death_range": "Between 31/12/2018 and 2/2/2019",
                    "state": "VIC",
                    "reference": "123400001",
                    "name_match": "exact"
                }
            ]
        }
    ]
}
```


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

Validate with the AEC

POST `/dvs/aec`

Validates a name, birthdate and address against the Australian Document Verification Service (DVS) using the Australian Electoral Commission.

Sandbox environment

When simulating queries in the sandbox environment, a specific set of details will return a specific result. The response will be determined by the suburb:

DVS sandbox responses for this document

Request body

The name, address and birthdate to validate

Field	Description
`preflight`	**boolean** When set to true, the request will only perform a preflight check to validate the input parameters. No actual verification will be performed. **Example:** `false`
`consent`	**boolean** The individual's consent to perform the check **Example:** `true`
`first_name`	**string** The first name of the individual **Example:** `John`
`last_name`	**string** The last name of the individual **Example:** `Smith`

Field	Description
`birth_date`	**string (date)** The date of birth of the individual **Example:** `1980-01-01`
`state`	**string** The Australian state of the individual's address **Enum:** `NSW`, `VIC`, `QLD`, `SA`, `WA`, `TAS`, `NT`, `ACT` **Example:** `VIC`
`postcode`	**string** The postcode of the individual's address **Example:** `3000`
`suburb`	**string** The suburb of the individual's address **Example:** `Melbourne`
`street_address`	**string** The street address of the individual. Cannot be used in conjunction with street_number, street_name, street_type and flat_unit_number. **Example:** `The Rialto 5/123 Smith St`
`street_number`	**string** The street number of the individual's address **Example:** `123`
`street_name`	**string** The street name of the individual's address **Example:** `Smith St`

Field	Description
street_type	**string** The street type of the individual's address **Enum:** `AC RD` , `ACCESS ROAD` , `ACCESS` , `ALY` , `ALLEY` , `AMBLE` , `APP` , `APPROACH` , `ARC` , `ARCADE` , `ART` , `ARTERIAL ROAD` , `ART LNK` , `ARTERIAL ROAD LINK` , `AVE` , `AVENUE` , `AVE C` , `AVENUE CENTRAL` , `AVE E` , `AVENUE EAST` , `AVE EXT` , `AVENUE EXTENSION` , `AVE N` , `AVENUE NORTH` , `AVE S` , `AVENUE SOUTH` , `AVE W` , `AVENUE WEST` , `BANK` , `BAY` , `BCH` , `BEACH` , `BEND` , `BLVD` , `BOULEVARDE` , `BLVD E` , `BOULEVARDE EAST` , `BLVD N` , `BOULEVARDE NORTH` , `BLVD S` , `BOULEVARDE SOUTH` , `BLVD W` , `BOULEVARDE WEST` , `BRACE` , `BRAE` , `BRDWLK` , `BOARDWALK` , `BRDWY` , `BROADWAY` , `BRETT` , `BRK` , `BREAK` , `BROW` , `BULL` , `BWL` , `BOWL` , `BYPS` , `BYPASS` , `BYWAY` , `CRES N` , `CRESCENT NORTH` , `CRES S` , `CRESCENT SOUTH` , `CRES W` , `CRESCENT WEST` , `CREST` , `CRIEF` , `CRS` , `CROSS` , `CRS W` , `CROSS WEST` , `CRSNG` , `CROSSING` , `CT` , `COURT` , `CT E` , `COURT EAST` , `CT N` , `COURT NORTH` , `CT S` , `COURT SOUTH` , `CT W` , `COURT WEST` , `CTS` , `COURTS` , `CTYD` , `COURTYARD` , `CUT` , `CUTTING` , `DALE` , `DELL` , `DENE` , `DEV` , `DEVIATION` , `DEV RD` , `DEVIATION ROAD` , `DIST` , `DISTRIBUTOR` , `DIVDE` , `DIVIDE` , `DOCK` , `DOMAIN` , `DOWNS` , `DR` , `DRIVE` , `DR E` , `DRIVE EAST` , `DR EXT` , `DRIVE EXTENDED` , `DR N` , `DRIVE NORTH` , `DR S` , `DRIVE SOUTH` , `DR W` , `DRIVE WEST` , `EDGE` , `ELBOW` , `END` , `ENT` , `ENTRANCE` , `ESMNT` , `EASEMENT` , `ESP` , `ESPLANADE` , `ESP E` , `ESPLANADE EAST` , `ESP N` , `ESPLANADE NORTH` , `ESP S` , `ESPLANADE SOUTH` , `CAUS` , `CAUSEWAY` , `CHASE` , `CIR` , `CIRCLE` , `CIR E` , `CIRCLE EAST` , `CIR N` , `CIRCLE NORTH` , `CIR S` , `CIRCLE SOUTH` , `CIR W` , `CIRCLE WEST` , `CL` , `CLOSE` , `CL E` , `CLOSE EAST` , `CL N` , `CLOSE NORTH` , `CL S` , `CLOSE SOUTH` , `CL W` , `CLOSE WEST` , `CLSTR` , `CLUSTER` , `CMMN` , `COMMON` , `CMMNS` , `COMMONS` , `CNCRSE` , `CONCOURSE` , `CNR` , `CORNER` , `CNTR` , `CENTRE` , `CORSO` , `COURSE` , `COVE` , `CPS` , `COPSE` , `CRCLT` , `CIRCLET` , `CRCS` , `CIRCUS` , `CRCS E` , `CIRCUS EAST` , `CRCS N` , `CIRCUS NORTH` , `CRCS S` , `CIRCUS SOUTH` , `CRCS W` , `CIRCUS WEST` , `CRCT` , `CIRCUIT` , `CRCT E` , `CIRCUIT EAST` , `CRCT N` , `CIRCUIT NORTH` , `CRCT S` , `CIRCUIT SOUTH` , `CRCT W` , `CIRCUIT WEST` , `CRES` , `CRESCENT` , `CRES E` , `CRESCENT EAST` , `ESP W` , `ESPLANADE WEST` , `EST` , `ESTATE` , `EXPWY` , `EXPRESSWAY` , `FAWY` , `FAIRWAY` , `FLAT` , `FOLW` , `FOLLOW` , `FORD` , `FR` , `FRONTAGE` , `FWY` , `FREEWAY` , `GAP` , `GATE` , `GATEWAY` , `GDN` , `GARDEN` , `GDNS` , `GARDENS` , `GLADE` , `GLEN` , `GLY` , `GULLY` , `GR` , `GROVE` , `GR E` , `GROVE EAST` , `GR N` , `GROVE NORTH` , `GR S` , `GROVE SOUTH` , `GR W` , `GROVE WEST` , `GRANGE` , `GREEN` , `HAVEN` , `HDWY` , `HIDEAWAY` , `HILL` , `HLLW` , `HOLLOW` , `HRBR` , `HARBOUR` , `HTH` , `HEATH` , `HTS` , `HEIGHTS` , `HWY` , `HIGHWAY` , `HWY E` , `HIGHWAY EAST` , `HWY N` , `HIGHWAY NORTH` , `HWY S` , `HIGHWAY SOUTH` , `HWY W` , `HIGHWAY WEST` , `INLET` , `IS` , `ISLAND` , `JN` , `JUNCTION` , `KEY` , `KEYS` , `KNOLL` , `LA` , `LANE` , `LA E` , `LANE EAST` , `LA N` , `LANE NORTH` , `LA S` , `LANE SOUTH` , `LA W` , `LANE WEST` , `LINE` , `LINK` , `LKT` , `LOOKOUT` , `LNDG` , `LANDING` , `LOOP` , `MALL` , `MART` , `MEAD` , `MEW` , `MEWS` , `MNDR` , `MEANDER` , `MTRWY` , `MOTORWAY` , `NOOK` , `OUTLK` , `OUTLOOK` , `OUTLT` , `OUTLET` , `OVAL` , `OVRPS` , `OVERPASS` , `PASS` , `PDE` , `PARADE` , `PDE E` , `PARADE EAST` , `PDE N` , `PARADE NORTH` , `PDE S` , `PARADE SOUTH` , `PDE W` , `PARADE WEST` , `PK` , `PARK` , `PK E` , `PARK EAST` , `PK N` , `PARK NORTH` , `PK S` , `PARK SOUTH` , `PK W` , `PARK WEST` , `PKT` , `POCKET` , `PKWY` , `PARKWAY` , `PL` , `PLACE` , `PL E` , `PLACE EAST` , `PL N` , `PLACE NORTH` , `PL S` , `PLACE SOUTH` , `PL W` , `PLACE WEST` , `PLAZA` , `PNSLA` , `PENINSULA` , `PORT` , `PREC` , `PRECINCT` , `PRMNDE` , `PROMENADE` , `PSGE` , `PASSAGE` , `PT` , `POINT` , `PTH` , `PATH` , `PTHWY` , `PATHWAY` , `PUR` , `PURSUIT` , `QDRNT` , `QUADRANT` , `QUAY` , `QUAY E` , `QUAY EAST` , `QUAY N` , `QUAY NORTH` , `QUAY S` , `QUAY SOUTH` , `QUAY W` , `QUAY WEST` , `QUAYS` , `RAMBLE` , `RCH` , `REACH` , `RD` , `ROAD` , `RD C` , `ROAD CENTRAL` , `RD E` , `ROAD EAST` , `RD EXT` , `ROAD EXTENDED` , `RD N` , `ROAD NORTH` , `RD S` , `ROAD SOUTH` , `RD SE` , `ROAD SOUTH EAST` , `RD W` , `ROAD WEST` , `RDS` , `ROADS` , `RDWY` , `ROADWAY` , `RESERVE` ,

Field	Description
`flat_unit_number`	**string** The flat or unit number of the individual's address **Example:** `5`
`habitation_building_name`	**string** The building name of the individual's address **Example:** `The Rialto`

Sample request

```
{
    "preflight": false,
    "consent": true,
    "first_name": "John",
    "last_name": "Smith",
    "birth_date": "1980-01-01",
    "state": "VIC",
    "postcode": "3000",
    "suburb": "Melbourne",
    "street_address": "The Rialto 5/123 Smith St",
    "street_number": "123",
    "street_name": "Smith St",
    "street_type": "ST",
    "flat_unit_number": "5",
    "habitation_building_name": "The Rialto"
}
```

Responses

200 **Result of the document verification check** (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`
`function`	**string** The function that was called **Example:** `dvs_aec`

Field	Description
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**object** The check data
`data.` `response_type`	**string** The type of response **Example:** `AECResponse`
`data.` `activity_id`	**string** A reference ID for this activity. This ID can be used to track the activity in the logs. **Example:** `sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d`
`data.` `verification_request_number`	**string** A unique reference number for this request **Example:** `dec306ed-7cff-41c0-b02f-889e430f5309`
`data.` `verification_result_code`	**string** The result of the verification - `Y` - The document is valid - `N` - The document was not matched - `D` - The document is invalid or not electronically captured - `S` - An error occurred during the check **Enum:** `Y`, `N`, `S`, `D` **Example:** `N`
`data.` `additional_information`	**array of objects** An array of expanded responses returned from the DVSHub or Document Issuer for N or D results

`data.`
`additional_information[].`
`message`
	string
	
	A message indicating the reason for the N or D result.
	
	Example: `Address not known or invalid.`

Field	Description
data. additional_information[]. code	**string** A code number associated with the corresponding message. **Example:** `EC-001`

Sample response

```
{
    "message": "Ok",
    "function": "dvs_aec",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": {
        "response_type": "AECResponse",
        "activity_id": "sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d",
        "verification_request_number": "dec306ed-7cff-41c0-b02f-889e430f5309",
        "verification_result_code": "N",
        "additional_information": [
            {
                "message": "Address not known or invalid.",
                "code": "EC-001"
            }
        ]
    }
}
```

202 **A successful preflight check. A 400 will be returned if the preflight check fails.** (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`
function	**string** The function that was called **Example:** `dvs_aec`
api_reference	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
data	**null** For preflight checks this is null

Sample response

```
{
    "message": "Ok",
    "function": "dvs_aec",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": null
}
```


Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

Validate an ASIC/MSIC Card

POST `/dvs/asic`

Validates a ASIC/MSIC card against the Australian Document Verification Service (DVS).

Sandbox environment

When simulating queries in the sandbox environment, a specific set of ASIC cards will return a specific result. The response will be determined by the card number:

DVS sandbox responses for this document

Request body

The ASIC or MSIC card details to validate

Field	Description
`preflight`	**boolean** When set to true, the request will only perform a preflight check to validate the input parameters. No actual verification will be performed. **Example:** `false`
`consent`	**boolean** The individual's consent to perform the check **Example:** `true`
`full_name`	**string** The full name of the document holder **Example:** `John Smith`
`birth_date`	**string (date)** The date of birth of the document holder **Example:** `1980-01-01`

Field	Description
`card_number`	**string** The card number on the card to be validated **Example:** `ABC123456`
`card_expiry`	**string** The expiry date of the card to be validated **Pattern:** `^[0-9]{4}-[0-9]{2}$` **Example:** `2024-02`
`card_type`	**string** The type of card to be validated **Enum:** `ASIC`, `MSIC` **Example:** `ASIC`

Sample request

```
{
    "preflight": false,
    "consent": true,
    "full_name": "John Smith",
    "birth_date": "1980-01-01",
    "card_number": "ABC123456",
    "card_expiry": "2024-02",
    "card_type": "ASIC"
}
```

Responses

200 **Result of the document verification check** (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`
`function`	**string** The function that was called **Example:** `dvs_asic`

Field	Description
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**object** The check data
`data.` `response_type`	**string** The type of response **Example:** `AsicMsicResponse`
`data.` `activity_id`	**string** A reference ID for this activity. This ID can be used to track the activity in the logs. **Example:** `sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d`
`data.` `verification_request_number`	**string** A unique reference number for this request **Example:** `dec306ed-7cff-41c0-b02f-889e430f5309`
`data.` `verification_result_code`	**string** The result of the verification - `Y` - The document is valid - `N` - The document was not matched - `D` - The document is invalid or not electronically captured - `S` - An error occurred during the check **Enum:** `Y`, `N`, `S`, `D` **Example:** `N`
`data.` `additional_information`	**array of objects** An array of expanded responses returned from the DVSHub or Document Issuer for N or D results

`data.`
`additional_information[].`
`message`
	string
	
	A message indicating the reason for the N or D result.
	
	Example: `Name on Card does not match`

Field	Description
data. additional_information[]. code	**string** A code number associated with the corresponding message. **Example:** `SI-001`

Sample response


```
{
    "message": "Ok",
    "function": "dvs_asic",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": {
        "response_type": "AsicMsicResponse",
        "activity_id": "sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d",
        "verification_request_number": "dec306ed-7cff-41c0-b02f-889e430f5309",
        "verification_result_code": "N",
        "additional_information": [
            {
                "message": "Name on Card does not match",
                "code": "SI-001"
            }
        ]
    }
}
```


202 **A successful preflight check. A 400 will be returned if the preflight check fails.** (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`
function	**string** The function that was called **Example:** `dvs_aec`
api_reference	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
data	**null** For preflight checks this is null

Sample response

```
{
    "message": "Ok",
    "function": "dvs_aec",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": null
}
```


Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

Validate a birth certificate

POST `/dvs/birth_cert`

Validates a birth certificate against the Australian Document Verification Service (DVS).

Sandbox environment

When simulating queries in the sandbox environment, a specific set of birth certificates will return a specific result. The response will be determined by the certificate number:

DVS sandbox responses for this document

Verification notes

- For ACT, NSW, NT, SA, VIC, WA: Either a Registration Number or Certificate Number must be provided. If both numbers are provided both will be checked.
- For QLD and TAS: Either a Registration Date or Certificate Number must be provided. If both are provided both will be checked. A Registration Number is not required.

> All states recommend that the Certificate Number is used for verification where possible as this is more reliable than the Registration Number or Registration Date. If a birth certificate fails to verify using the Registration Number or Registration Date, it is recommended to try again using the Certificate Number.

Certificate Number and private sector callers

Private sector DVS users, which includes all Global Data API customers, **must** supply the Certificate Number for NT, SA and ACT certificates issued on or after the following dates:

State	Certificate Number required for certificates issued from
NT	12 July 1999
SA	1 November 1999
ACT	1 May 2002

This is a DVS rule rather than an API validation rule, and it is deliberately not enforced here. The validation above accepts either number, because a certificate issued before these dates does not carry a Certificate Number at all and must still be able to verify on its Registration Number.

The consequence is that a request can pass validation and still come back unverified. If you are checking an NT, SA or ACT certificate issued on or after the date above and you do not send `certificate_number`, expect a `D` result (no match on the data supplied) rather than a validation error.

Request body

The birth certificate details to validate

Field	Description
`preflight`	**boolean** When set to true, the request will only perform a preflight check to validate the input parameters. No actual verification will be performed. **Example:** `false`
`consent`	**boolean** The individual's consent to perform the check **Example:** `true`
`first_name`	**string** The first name of the document holder **Example:** `John`
`last_name`	**string** The last name of the document holder **Example:** `Smith`
`birth_date`	**string (date)** The date of birth of the document holder **Example:** `1980-01-01`
`registration_number`	**string** The registration number on the certificate to be validated **Example:** `1234567`
`certificate_number`	**string** The certificate number on the certificate to be validated **Example:** `123456789`

Field	Description
`registration_date`	**string (date)** The registration date on the certificate to be validated (only relevant for QLD or TAS) **Example:** `2021-02-28`
`state`	**string** The state of issue for the birth certificate **Enum:** `NSW`, `VIC`, `QLD`, `TAS`, `ACT`, `NT`, `SA`, `WA` **Example:** `VIC`

Sample request

```
{
    "preflight": false,
    "consent": true,
    "first_name": "John",
    "last_name": "Smith",
    "birth_date": "1980-01-01",
    "registration_number": "1234567",
    "certificate_number": "123456789",
    "registration_date": "2021-02-28",
    "state": "VIC"
}
```

Responses

200 **Result of the document verification check** (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`
`function`	**string** The function that was called **Example:** `dvs_birth_cert`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Field	Description
`data`	**object** The check data
`data.` `response_type`	**string** The type of response **Example:** `BirthCertificateResponse`
`data.` `activity_id`	**string** A reference ID for this activity. This ID can be used to track the activity in the logs. **Example:** `sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d`
`data.` `verification_request_number`	**string** A unique reference number for this request **Example:** `dec306ed-7cff-41c0-b02f-889e430f5309`
`data.` `verification_result_code`	**string** The result of the verification - `Y` - The document is valid - `N` - The document was not matched - `D` - The document is invalid or not electronically captured - `S` - An error occurred during the check **Enum:** `Y`, `N`, `S`, `D` **Example:** `N`
`data.` `additional_information`	**array of objects** An array of expanded responses returned from the DVSHub or Document Issuer for N or D results

`data.`
`additional_information[].`
`message`
	string
	
	A message indicating the reason for the N or D result.
	
	Example: `Family Name does not match.`

`data.`
`additional_information[].`
`code`
	string
	
	A code number associated with the corresponding message.
	
	Example: `BC-001`

Sample response

```
{
    "message": "Ok",
    "function": "dvs_birth_cert",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": {
        "response_type": "BirthCertificateResponse",
        "activity_id": "sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d",
        "verification_request_number": "dec306ed-7cff-41c0-b02f-889e430f5309",
        "verification_result_code": "N",
        "additional_information": [
            {
                "message": "Family Name does not match.",
                "code": "BC-001"
            }
        ]
    }
}
```


202 **A successful preflight check. A 400 will be returned if the preflight check fails.** (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`
`function`	**string** The function that was called **Example:** `dvs_aec`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**null** For preflight checks this is null

Sample response


```
{
    "message": "Ok",
    "function": "dvs_aec",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": null
}
```


Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

Validate a Centrelink concession card

POST `/dvs/centrelink`

Validates a Centrelink concession card against the Australian Document Verification Service (DVS).

Sandbox environment

When simulating queries in the sandbox environment, a specific set of Centrelink cards will return a specific result. The response will be determined by the CRN:

DVS sandbox responses for this document

Request body

The Centrelink card details to validate

Field	Description
`preflight`	**boolean** When set to true, the request will only perform a preflight check to validate the input parameters. No actual verification will be performed. **Example:** `false`
`consent`	**boolean** The individual's consent to perform the check **Example:** `true`
`full_name`	**string** The full name of the card holder **Example:** `John Smith`
`birth_date`	**string (date)** The date of birth of the card holder **Example:** `1980-01-01`
`crn`	**string** The crn number on the card to be validated **Example:** `123456789A`

Field	Description
`card_expiry`	**string (date)** The expiry date of the card to be validated **Example:** `2024-02-01`
`card_type`	**string** The type of card to be validated **Enum:** `HCC`, `PCC`, `SHC` **Example:** `HCC`

Sample request

```
{
    "preflight": false,
    "consent": true,
    "full_name": "John Smith",
    "birth_date": "1980-01-01",
    "crn": "123456789A",
    "card_expiry": "2024-02-01",
    "card_type": "HCC"
}
```

Responses

200 **Result of the document verification check** (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`
`function`	**string** The function that was called **Example:** `dvs_centrelink`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**object** The check data

Field	Description
data. response_type	**string** The type of response **Example:** `CentrelinkResponse`
data. activity_id	**string** A reference ID for this activity. This ID can be used to track the activity in the logs. **Example:** `sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d`
data. verification_request_number	**string** A unique reference number for this request **Example:** `dec306ed-7cff-41c0-b02f-889e430f5309`
data. verification_result_code	**string** The result of the verification • `Y` - The document is valid • `N` - The document was not matched • `D` - The document is invalid or not electronically captured • `S` - An error occurred during the check **Enum:** `Y`, `N`, `S`, `D` **Example:** `N`
data. additional_information	**array of objects** An array of expanded responses returned from the DVSHub or Document Issuer for N or D results
data. additional_information[]. message	**string** A message indicating the reason for the N or D result. **Example:** `Name does not match.`
data. additional_information[]. code	**string** A code number associated with the corresponding message. **Example:** `CC-001`

Sample response

```
{
    "message": "Ok",
    "function": "dvs_centrelink",
```

```
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": {
        "response_type": "CentrelinkResponse",
        "activity_id": "sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d",
        "verification_request_number": "dec306ed-7cff-41c0-b02f-889e430f5309",
        "verification_result_code": "N",
        "additional_information": [
            {
                "message": "Name does not match.",
                "code": "CC-001"
            }
        ]
    }
}
```


202 **A successful preflight check. A 400 will be returned if the preflight check fails.** (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`
`function`	**string** The function that was called **Example:** `dvs_aec`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**null** For preflight checks this is null

Sample response


```
{
    "message": "Ok",
    "function": "dvs_aec",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": null
}
```


Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

Validate a citizenship certificate

POST `/dvs/citizenship`

Validates a Citizenship certificate against the Australian Document Verification Service (DVS).

Sandbox environment

When simulating queries in the sandbox environment, a specific set of citizenship certificates will return a specific result. The response will be determined by the stock number:

DVS sandbox responses for this document

Request body

The citizenship certificate details to validate

Field	Description
`preflight`	**boolean** When set to true, the request will only perform a preflight check to validate the input parameters. No actual verification will be performed. **Example:** `false`
`consent`	**boolean** The individual's consent to perform the check **Example:** `true`
`first_name`	**string** The first name of the document holder **Example:** `John`
`last_name`	**string** The last name of the document holder **Example:** `Smith`
`birth_date`	**string (date)** The date of birth of the document holder **Example:** `1980-01-01`

Field	Description
`acquisition_date`	**string (date)** The acquisition date specified on the certificate **Example:** `1980-01-01`
`stock_number`	**string** The stock number specified on the certificate **Example:** `123/45678`

Sample request

```
{
    "preflight": false,
    "consent": true,
    "first_name": "John",
    "last_name": "Smith",
    "birth_date": "1980-01-01",
    "acquisition_date": "1980-01-01",
    "stock_number": "123/45678"
}
```

Responses

200 **Result of the document verification check** (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`
`function`	**string** The function that was called **Example:** `dvs_citizenship`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**object** The check data

Field	Description
data. response_type	**string** The type of response **Example:** `CitizenshipCertificateResponse`
data. activity_id	**string** A reference ID for this activity. This ID can be used to track the activity in the logs. **Example:** `sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d`
data. verification_request_number	**string** A unique reference number for this request **Example:** `dec306ed-7cff-41c0-b02f-889e430f5309`
data. verification_result_code	**string** The result of the verification - `Y` - The document is valid - `N` - The document was not matched - `D` - The document is invalid or not electronically captured - `S` - An error occurred during the check **Enum:** `Y`, `N`, `S`, `D` **Example:** `N`
data. additional_information	**array of objects** An array of expanded responses returned from the DVSHub or Document Issuer for N or D results
data. additional_information[]. message	**string** A message indicating the reason for the N or D result. **Example:** `Document invalid.`
data. additional_information[]. code	**string** A code number associated with the corresponding message. **Example:** `CC-001`

Sample response

```
{
    "message": "Ok",
    "function": "dvs_citizenship",
```

```
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": {
        "response_type": "CitizenshipCertificateResponse",
        "activity_id": "sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d",
        "verification_request_number": "dec306ed-7cff-41c0-b02f-889e430f5309",
        "verification_result_code": "N",
        "additional_information": [
            {
                "message": "Document invalid.",
                "code": "CC-001"
            }
        ]
    }
}
```


202 **A successful preflight check. A 400 will be returned if the preflight check fails.** (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`
`function`	**string** The function that was called **Example:** `dvs_aec`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**null** For preflight checks this is null

Sample response


```
{
    "message": "Ok",
    "function": "dvs_aec",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": null
}
```


Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

Validate a death certificate

POST `/dvs/death_cert`

Validates a death certificate against the Australian Document Verification Service (DVS).

Sandbox environment

When simulating queries in the sandbox environment, a specific set of death certificates will return a specific result. The response will be determined by the certificate number:

DVS sandbox responses for this document

Request body

The death certificate details to validate

Field	Description
`preflight`	**boolean** When set to true, the request will only perform a preflight check to validate the input parameters. No actual verification will be performed. **Example:** `false`
`consent`	**boolean** The individual's consent to perform the check **Example:** `true`
`first_name`	**string** The first name(s) stated on the document **Example:** `John`
`last_name`	**string** The last name stated on the document **Example:** `Smith`
`event_date`	**string (date)** The date of of the death **Example:** `1980-01-01`

Field	Description
`registration_number`	**string** The registration number on the certificate to be validated **Example:** `1234567`
`certificate_number`	**string** The certificate number on the certificate to be validated **Example:** `123456789`
`registration_date`	**string (date)** The registration date on the certificate to be validated (only relevant for QLD or TAS) **Example:** `2021-02-28`
`state`	**string** The state of issue for the death certificate **Enum:** `NSW`, `VIC`, `QLD`, `TAS`, `ACT`, `NT`, `SA`, `WA` **Example:** `VIC`

Sample request

```
{
    "preflight": false,
    "consent": true,
    "first_name": "John",
    "last_name": "Smith",
    "event_date": "1980-01-01",
    "registration_number": "1234567",
    "certificate_number": "123456789",
    "registration_date": "2021-02-28",
    "state": "VIC"
}
```

Responses

200 **Result of the document verification check** (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`

Field	Description
`function`	**string** The function that was called **Example:** `dvs_death_cert`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**object** The check data
`data.` `response_type`	**string** The type of response **Example:** `DeathCertificateResponse`
`data.` `activity_id`	**string** A reference ID for this activity. This ID can be used to track the activity in the logs. **Example:** `sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d`
`data.` `verification_request_number`	**string** A unique reference number for this request **Example:** `dec306ed-7cff-41c0-b02f-889e430f5309`
`data.` `verification_result_code`	**string** The result of the verification - `Y` - The document is valid - `N` - The document was not matched - `D` - The document is invalid or not electronically captured - `S` - An error occurred during the check **Enum:** `Y`, `N`, `S`, `D` **Example:** `N`
`data.` `additional_information`	**array of objects** An array of expanded responses returned from the DVSHub or Document Issuer for N or D results

Field	Description
data. additional_information[]. message	**string** A message indicating the reason for the N or D result. **Example:** `Given Name/s does not match.`
data. additional_information[]. code	**string** A code number associated with the corresponding message. **Example:** `DC-001`

Sample response

```
{
    "message": "Ok",
    "function": "dvs_death_cert",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": {
        "response_type": "DeathCertificateResponse",
        "activity_id": "sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d",
        "verification_request_number": "dec306ed-7cff-41c0-b02f-889e430f5309",
        "verification_result_code": "N",
        "additional_information": [
            {
                "message": "Given Name/s does not match.",
                "code": "DC-001"
            }
        ]
    }
}
```

202 **A successful preflight check. A 400 will be returned if the preflight check fails.** (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`
function	**string** The function that was called **Example:** `dvs_aec`
api_reference	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Field	Description
`data`	**null** For preflight checks this is null

Sample response

```
{
    "message": "Ok",
    "function": "dvs_aec",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": null
}
```

Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

Validate a drivers licence

POST `/dvs/drivers_licence`

Validates a drivers licence number against the Australian Document Verification Service (DVS).

N.B. Middle names will not be checked if not supplied so it is optional to supply it.

Victorian driver licences sometimes only print an initial for the middle name. However the full first middle name will be on record and must be entered if the middle name is supplied.

For this reason, DVS advise not to send the middle name for a licence check. This will work for all states and territories.

If a middle name is supplied then it will be checked. An initial entered for the middle name will likely result in a failed matched.

Sandbox environment

When simulating queries in the sandbox environment, a specific set of drivers licence numbers will return a specific result. The response will be determined by the drivers licence number:

DVS sandbox responses for this document

Request body

The licence details to validate

Field	Description
`preflight`	**boolean** When set to true, the request will only perform a preflight check to validate the input parameters. No actual verification will be performed. **Example:** `false`

Field	Description
`consent`	**boolean** The individual's consent to perform the check **Example:** `true`
`first_name`	**string** The first name of the licence holder **Example:** `John`
`middle_name`	**string or null** The middle name of the licence holder. This field is optional and can be left blank. It will not be checked if not supplied. However, if supplied, it will be checked. **Example:** `James`
`last_name`	**string** The last name of the licence holder **Example:** `Smith`
`birth_date`	**string (date)** The date of birth of the licence holder **Example:** `1980-01-01`
`licence_number`	**string** The drivers licence number to be validated **Example:** `1234567890`
`card_number`	**string** The card number of the licence to be validated **Example:** `P0001282`
`state`	**string** The state of issue for the drivers licence **Enum:** `NSW` , `VIC` , `QLD` , `SA` , `WA` , `TAS` , `NT` , `ACT` **Example:** `VIC`

Sample request

```
{
    "preflight": false,
    "consent": true,
    "first_name": "John",
    "middle_name": "James",
    "last_name": "Smith",
    "birth_date": "1980-01-01",
    "licence_number": "1234567890",
    "card_number": "P0001282",
    "state": "VIC"
}
```

Responses

200 **Result of licence validation** (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`
`function`	**string** The function that was called **Example:** `dvs_drivers_licence`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**object** The check data
`data.` `response_type`	**string** The type of response **Example:** `DriverLicenceResponse`
`data.` `activity_id`	**string** A reference ID for this activity. This ID can be used to track the activity in the logs. **Example:** `sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d`

Field	Description
data. verification_request_number	**string** A unique reference number for this request **Example:** `dec306ed-7cff-41c0-b02f-889e430f5309`
data. verification_result_code	**string** The result of the verification - `Y` - The licence is valid - `N` - The licence is invalid - `D` - The licence is unknown - `S` - An error occurred during the check **Enum:** `Y`, `N`, `S`, `D` **Example:** `N`
data. additional_information	**array of objects** An array of expanded responses returned from the DVSHub or Document Issuer for N or D results
data. additional_information[]. message	**string** A message indicating the reason for the N or D result. **Example:** `Document invalid.`
data. additional_information[]. code	**string** A code number associated with the corresponding message. **Example:** `DL-001`

Sample response


```
{
    "message": "Ok",
    "function": "dvs_drivers_licence",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": {
        "response_type": "DriverLicenceResponse",
        "activity_id": "sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d",
        "verification_request_number": "dec306ed-7cff-41c0-b02f-889e430f5309",
        "verification_result_code": "N",
        "additional_information": [
            {
                "message": "Document invalid.",
                "code": "DL-001"
            }
        ]
    }
```

```
}
```

202 **A successful preflight check. A 400 will be returned if the preflight check fails.** (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`
`function`	**string** The function that was called **Example:** `dvs_aec`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**null** For preflight checks this is null

Sample response

```
{
    "message": "Ok",
    "function": "dvs_aec",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": null
}
```

Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

Validate an ImmiCard

POST `/dvs/immicard`

Validates an ImmiCard against the Australian Document Verification Service (DVS).

Sandbox environment

When simulating queries in the sandbox environment, a specific set of ImmiCards will return a specific result. The response will be determined by the card number:

DVS sandbox responses for this document

Request body

The ImmiCard card details to validate

Field	Description
`preflight`	**boolean** When set to true, the request will only perform a preflight check to validate the input parameters. No actual verification will be performed. **Example:** `false`
`consent`	**boolean** The individual's consent to perform the check **Example:** `true`
`first_name`	**string** The first name of the card holder **Example:** `John`
`last_name`	**string** The last name of the card holder **Example:** `Smith`
`birth_date`	**string (date)** The date of birth of the card holder **Example:** `1980-01-01`
`card_number`	**string** The card number on the ImmiCard to be validated **Pattern:** `^[A-Za-z]{3}[0-9]{6}$` **Example:** `ABC123456`

Sample request

```
{
    "preflight": false,
    "consent": true,
    "first_name": "John",
    "last_name": "Smith",
    "birth_date": "1980-01-01",
    "card_number": "ABC123456"
}
```

Responses

200 **Result of the document verification check** (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`
`function`	**string** The function that was called **Example:** `dvs_immicard`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**object** The check data
`data.` `response_type`	**string** The type of response **Example:** `ImmiCardResponse`
`data.` `activity_id`	**string** A reference ID for this activity. This ID can be used to track the activity in the logs. **Example:** `sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d`
`data.` `verification_request_number`	**string** A unique reference number for this request **Example:** `dec306ed-7cff-41c0-b02f-889e430f5309`

Field	Description
data. verification_result_code	**string** The result of the verification - `Y` - The document is valid - `N` - The document was not matched - `D` - The document is invalid or not electronically captured - `S` - An error occurred during the check **Enum:** `Y`, `N`, `S`, `D` **Example:** `N`
data. additional_information	**array of objects** An array of expanded responses returned from the DVSHub or Document Issuer for N or D results
data. additional_information[]. message	**string** A message indicating the reason for the N or D result. **Example:** `Travel document could not be found.`
data. additional_information[]. code	**string** A code number associated with the corresponding message. **Example:** `IM-001`

Sample response

```
{
    "message": "Ok",
    "function": "dvs_immicard",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": {
        "response_type": "ImmiCardResponse",
        "activity_id": "sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d",
        "verification_request_number": "dec306ed-7cff-41c0-b02f-889e430f5309",
        "verification_result_code": "N",
        "additional_information": [
            {
                "message": "Travel document could not be found.",
                "code": "IM-001"
            }
        ]
    }
}
```

202 **A successful preflight check. A 400 will be returned if the preflight check fails.** (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`
`function`	**string** The function that was called **Example:** `dvs_aec`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**null** For preflight checks this is null

Sample response

```
{
    "message": "Ok",
    "function": "dvs_aec",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": null
}
```

Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

Validate a marriage certificate

POST `/dvs/marriage`

Validates a marriage certificate against the Australian Document Verification Service (DVS).

The registration field is state-specific: for **QLD** the `registration_date` is used, while for all **other states** the `registration_number` is used. If you supply the field that does not apply to the requested state, it is silently ignored and not forwarded to DVS. A supplied `registration_date` must still be a valid `YYYY-MM-DD` date even when it will be ignored, otherwise the request is rejected.

Sandbox environment

When simulating queries in the sandbox environment, a specific set of marriage certificates will return a specific result. The response will be determined by the certificate number:

DVS sandbox responses for this document

Request body

The marriage certificate details to validate

Field	Description
`preflight`	**boolean** When set to true, the request will only perform a preflight check to validate the input parameters. No actual verification will be performed. **Example:** `false`
`consent`	**boolean** The individual's consent to perform the check **Example:** `true`
`first_name1`	**string** The first name of the first person listed on the document **Example:** `John`
`last_name1`	**string** The last name of the first person listed on the document **Example:** `Smith`
`first_name2`	**string** The first name of the second person listed on the document **Example:** `Mary`
`last_name2`	**string** The last name of the second person listed on the document **Example:** `Jones`
`event_date`	**string (date)** The date of the marriage **Example:** `1980-01-01`
`registration_number`	**string** The registration number on the certificate to be validated. Used for all states except QLD; silently ignored for QLD certificates. **Example:** `1234567`

Field	Description
`certificate_number`	**string** The certificate number on the certificate to be validated **Example:** `123456789`
`registration_date`	**string (date)** The registration date on the certificate to be validated. Used for QLD certificates only; silently ignored for all other states. Must be a valid YYYY-MM-DD date even when ignored. **Example:** `2021-02-28`
`state`	**string** The state of issue for the marriage certificate **Enum:** `NSW`, `VIC`, `QLD`, `TAS`, `ACT`, `NT`, `SA`, `WA` **Example:** `VIC`

Sample request

```
{
    "preflight": false,
    "consent": true,
    "first_name1": "John",
    "last_name1": "Smith",
    "first_name2": "Mary",
    "last_name2": "Jones",
    "event_date": "1980-01-01",
    "registration_number": "1234567",
    "certificate_number": "123456789",
    "registration_date": "2021-02-28",
    "state": "VIC"
}
```

Responses

200 **Result of the document verification check** (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`

Field	Description
`function`	**string** The function that was called **Example:** `dvs_marriage`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**object** The check data
`data.` `response_type`	**string** The type of response **Example:** `MarriageCertificateResponse`
`data.` `activity_id`	**string** A reference ID for this activity. This ID can be used to track the activity in the logs. **Example:** `sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d`
`data.` `verification_request_number`	**string** A unique reference number for this request **Example:** `dec306ed-7cff-41c0-b02f-889e430f5309`
`data.` `verification_result_code`	**string** The result of the verification - `Y` - The document is valid - `N` - The document was not matched - `D` - The document is invalid or not electronically captured - `S` - An error occurred during the check **Enum:** `Y`, `N`, `S`, `D` **Example:** `N`
`data.` `additional_information`	**array of objects** An array of expanded responses returned from the DVSHub or Document Issuer for N or D results

Field	Description
data. additional_information[]. message	**string** A message indicating the reason for the N or D result. **Example:** `Family Name does not match.`
data. additional_information[]. code	**string** A code number associated with the corresponding message. **Example:** `MC-001`

Sample response

```
{
    "message": "Ok",
    "function": "dvs_marriage",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": {
        "response_type": "MarriageCertificateResponse",
        "activity_id": "sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d",
        "verification_request_number": "dec306ed-7cff-41c0-b02f-889e430f5309",
        "verification_result_code": "N",
        "additional_information": [
            {
                "message": "Family Name does not match.",
                "code": "MC-001"
            }
        ]
    }
}
```

202 **A successful preflight check. A 400 will be returned if the preflight check fails.** (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`
function	**string** The function that was called **Example:** `dvs_aec`
api_reference	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Field	Description
`data`	**null** For preflight checks this is null

Sample response

```
{
    "message": "Ok",
    "function": "dvs_aec",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": null
}
```

Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

Validate a Medicare card

POST `/dvs/medicare`

Validates a Medicare card against the Australian Document Verification Service (DVS).

Sandbox environment

When simulating queries in the sandbox environment, a specific set of medicare cards will return a specific result. The response will be determined by the card number:

DVS sandbox responses for this document

Request body

The medicare card details to validate

Field	Description
`preflight`	**boolean** When set to true, the request will only perform a preflight check to validate the input parameters. No actual verification will be performed. **Example:** `false`
`consent`	**boolean** The individual's consent to perform the check **Example:** `true`

Field	Description
`full_name1`	**string** Line 1 of the full name of the card holder **Example:** `Marguerite`
`full_name2`	**string** Line 2 of the full name of the card holder **Example:** `Eleanor`
`full_name3`	**string** Line 3 of the full name of the card holder **Example:** `Jayne`
`full_name4`	**string** Line 4 of the full name of the card holder **Example:** `Salisbury`
`birth_date`	**string (date)** The date of birth of the card holder **Example:** `1980-01-01`
`card_number`	**string** The card number on the card to be validated **Example:** `2951636283`
`individual_ref_number`	**string** The individual reference number on the card for the card holder **Pattern:** `^[1-9]{1}$` **Example:** `1`
`card_expiry`	**string** The expiry date of the card to be validated **Example:** `2024-02`

Field	Description
`card_type`	**string** The type of card to be validated **Enum:** `G`, `Y`, `B` **Example:** `G`

Sample request

```
{
    "preflight": false,
    "consent": true,
    "full_name1": "Marguerite",
    "full_name2": "Eleanor",
    "full_name3": "Jayne",
    "full_name4": "Salisbury",
    "birth_date": "1980-01-01",
    "card_number": "2951636283",
    "individual_ref_number": "1",
    "card_expiry": "2024-02",
    "card_type": "G"
}
```

Responses

200 **Result of the document verification check** (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`
`function`	**string** The function that was called **Example:** `dvs_medicare`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**object** The check data

Field	Description
data. response_type	**string** The type of response **Example:** `MedicareResponse`
data. activity_id	**string** A reference ID for this activity. This ID can be used to track the activity in the logs. **Example:** `sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d`
data. verification_request_number	**string** A unique reference number for this request **Example:** `dec306ed-7cff-41c0-b02f-889e430f5309`
data. verification_result_code	**string** The result of the verification - `Y` - The document is valid - `N` - The document was not matched - `D` - The document is invalid or not electronically captured - `S` - An error occurred during the check **Enum:** `Y`, `N`, `S`, `D` **Example:** `N`
data. additional_information	**array of objects** An array of expanded responses returned from the DVSHub or Document Issuer for N or D results
data. additional_information[]. message	**string** A message indicating the reason for the N or D result. **Example:** `Name does not match.`
data. additional_information[]. code	**string** A code number associated with the corresponding message. **Example:** `MD-001`

Sample response

```
{
    "message": "Ok",
    "function": "dvs_medicare",
```

```
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": {
        "response_type": "MedicareResponse",
        "activity_id": "sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d",
        "verification_request_number": "dec306ed-7cff-41c0-b02f-889e430f5309",
        "verification_result_code": "N",
        "additional_information": [
            {
                "message": "Name does not match.",
                "code": "MD-001"
            }
        ]
    }
}
```


202 **A successful preflight check. A 400 will be returned if the preflight check fails.** (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`
`function`	**string** The function that was called **Example:** `dvs_aec`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**null** For preflight checks this is null

Sample response


```
{
    "message": "Ok",
    "function": "dvs_aec",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": null
}
```


Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

Validate a change of name certificate

POST `/dvs/name_change`

Validates a change of name certificate against the Australian Document Verification Service (DVS).

Sandbox environment

When simulating queries in the sandbox environment, a specific set of change of name certificates will return a specific result. The response will be determined by the certificate number:

DVS sandbox responses for this document

Request body

The change of name certificate details to validate

Field	Description
`preflight`	**boolean** When set to true, the request will only perform a preflight check to validate the input parameters. No actual verification will be performed. **Example:** `false`
`consent`	**boolean** The individual's consent to perform the check **Example:** `true`
`first_name`	**string** The new first name of the document holder **Example:** `John`
`last_name`	**string** The new last name of the document holder **Example:** `Smith`
`birth_date`	**string (date)** The date of birth of the document holder **Example:** `1980-01-01`

Field	Description
`registration_number`	**string** The registration number on the certificate to be validated **Example:** `1234567`
`certificate_number`	**string** The certificate number on the certificate to be validated **Example:** `123456789`
`registration_date`	**string (date)** The registration date on the certificate to be validated (only relevant for QLD or TAS) **Example:** `2021-02-28`
`state`	**string** The state of issue for the change of name certificate **Enum:** `NSW`, `VIC`, `QLD`, `TAS`, `ACT`, `NT`, `SA`, `WA` **Example:** `VIC`

Sample request

```
{
    "preflight": false,
    "consent": true,
    "first_name": "John",
    "last_name": "Smith",
    "birth_date": "1980-01-01",
    "registration_number": "1234567",
    "certificate_number": "123456789",
    "registration_date": "2021-02-28",
    "state": "VIC"
}
```

Responses

200 **Result of the document verification check** (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`

Field	Description
`function`	**string** The function that was called **Example:** `dvs_name_change_cert`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**object** The check data
`data.` `response_type`	**string** The type of response **Example:** `ChangeOfNameCertificateResponse`
`data.` `activity_id`	**string** A reference ID for this activity. This ID can be used to track the activity in the logs. **Example:** `sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d`
`data.` `verification_request_number`	**string** A unique reference number for this request **Example:** `dec306ed-7cff-41c0-b02f-889e430f5309`
`data.` `verification_result_code`	**string** The result of the verification - `Y` - The document is valid - `N` - The document was not matched - `D` - The document is invalid or not electronically captured - `S` - An error occurred during the check **Enum:** `Y`, `N`, `S`, `D` **Example:** `N`
`data.` `additional_information`	**array of objects** An array of expanded responses returned from the DVSHub or Document Issuer for N or D results

Field	Description
data. additional_information[]. message	**string** A message indicating the reason for the N or D result. **Example:** `Registration number does not match.`
data. additional_information[]. code	**string** A code number associated with the corresponding message. **Example:** `NC-001`

Sample response

```
{
    "message": "Ok",
    "function": "dvs_name_change_cert",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": {
        "response_type": "ChangeOfNameCertificateResponse",
        "activity_id": "sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d",
        "verification_request_number": "dec306ed-7cff-41c0-b02f-889e430f5309",
        "verification_result_code": "N",
        "additional_information": [
            {
                "message": "Registration number does not match.",
                "code": "NC-001"
            }
        ]
    }
}
```

202 **A successful preflight check. A 400 will be returned if the preflight check fails.** (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`
function	**string** The function that was called **Example:** `dvs_aec`
api_reference	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Field	Description
`data`	**null** For preflight checks this is null

Sample response

```
{
    "message": "Ok",
    "function": "dvs_aec",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": null
}
```

Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

Validate Australian travel documents

POST `/dvs/passport`

Validates a Passport, Certificate of Identity, Document of Identity, or UN Convention Travel Document against the Australian Document Verification Service (DVS).

Sandbox environment

When simulating queries in the sandbox environment, a specific set of travel document numbers will return a specific result. The response will be determined by the travel document number:

DVS sandbox responses for this document

Request body

The document details to validate

Field	Description
`preflight`	**boolean** When set to true, the request will only perform a preflight check to validate the input parameters. No actual verification will be performed. **Example:** `false`
`consent`	**boolean** The individual's consent to perform the check **Example:** `true`

Field	Description
`first_name`	**string** The first name of the document holder as printed on the document **Example:** `John`
`last_name`	**string** The last name of the document holder as printed on the document **Example:** `Smith`
`birth_date`	**string (date)** The date of birth of the document holder **Example:** `1980-01-01`
`travel_document_number`	**string** The travel document number to be validated **Example:** `M1234567`
`gender`	**string** The gender of the document holder **Enum:** `M`, `F`, `X` **Example:** `M`

Sample request

```
{
    "preflight": false,
    "consent": true,
    "first_name": "John",
    "last_name": "Smith",
    "birth_date": "1980-01-01",
    "travel_document_number": "M1234567",
    "gender": "M"
}
```

Responses

200 **Result of the validation** (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`
`function`	**string** The function that was called **Example:** `dvs_passport`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**object** The check data
`data.` `response_type`	**string** The type of response **Example:** `PassportResponse`
`data.` `activity_id`	**string** A reference ID for this activity. This ID can be used to track the activity in the logs. **Example:** `sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d`
`data.` `verification_request_number`	**string** A unique reference number for this request **Example:** `dec306ed-7cff-41c0-b02f-889e430f5309`
`data.` `verification_result_code`	**string** The result of the verification - `Y` - The document is valid - `N` - The document was not matched - `D` - The document is invalid or not electronically captured - `S` - An error occurred during the check **Enum:** `Y`, `N`, `S`, `D` **Example:** `N`

data. additional_information	**array of objects** An array of expanded responses returned from the DVSHub or Document Issuer for N or D results
data. additional_information[]. message	**string** A message indicating the reason for the N or D result. **Example:** `Document Invalid.`
data. additional_information[]. code	**string** A code number associated with the corresponding message. **Example:** `PP-001`

Sample response

```
{
    "message": "Ok",
    "function": "dvs_passport",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": {
        "response_type": "PassportResponse",
        "activity_id": "sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d",
        "verification_request_number": "dec306ed-7cff-41c0-b02f-889e430f5309",
        "verification_result_code": "N",
        "additional_information": [
            {
                "message": "Document Invalid.",
                "code": "PP-001"
            }
        ]
    }
}
```

202 **A successful preflight check. A 400 will be returned if the preflight check fails.** (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`
function	**string** The function that was called **Example:** `dvs_aec`

Field	Description
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**null** For preflight checks this is null

Sample response

```
{
    "message": "Ok",
    "function": "dvs_aec",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": null
}
```

Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

Validate a registration by descent certificate

POST `/dvs/reg_by_descent`

Validates a Registration by Descent certificate against the Australian Document Verification Service (DVS).

Sandbox environment

When simulating queries in the sandbox environment, a specific set of Registration by Descent certificates will return a specific result. The response will be determined by the stock number:

DVS sandbox responses for this document

Request body

The Registration by Descent certificate details to validate

Field	Description
`preflight`	**boolean** When set to true, the request will only perform a preflight check to validate the input parameters. No actual verification will be performed. **Example:** `false`

Field	Description
`consent`	**boolean** The individual's consent to perform the check **Example:** `true`
`first_name`	**string** The first name of the document holder **Example:** `John`
`last_name`	**string** The last name of the document holder **Example:** `Smith`
`birth_date`	**string (date)** The date of birth of the document holder **Example:** `1980-01-01`
`acquisition_date`	**string (date)** The acquisition date specified on the certificate **Example:** `1980-01-01`
`stock_number`	**string** The stock number specified on the certificate **Example:** `123/45678`

Sample request

```
{
    "preflight": false,
    "consent": true,
    "first_name": "John",
    "last_name": "Smith",
    "birth_date": "1980-01-01",
    "acquisition_date": "1980-01-01",
    "stock_number": "123/45678"
}
```

Responses

200 **Result of the document verification check** (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`
`function`	**string** The function that was called **Example:** `dvs_reg_by_descent`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**object** The check data
`data.` `response_type`	**string** The type of response **Example:** `RegistrationbyDescentCertificateResponse`
`data.` `activity_id`	**string** A reference ID for this activity. This ID can be used to track the activity in the logs. **Example:** `sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d`
`data.` `verification_request_number`	**string** A unique reference number for this request **Example:** `dec306ed-7cff-41c0-b02f-889e430f5309`
`data.` `verification_result_code`	**string** The result of the verification - `Y` - The document is valid - `N` - The document was not matched - `D` - The document is invalid or not electronically captured - `S` - An error occurred during the check **Enum:** `Y`, `N`, `S`, `D` **Example:** `N`

data. additional_information	**array of objects** An array of expanded responses returned from the DVSHub or Document Issuer for N or D results
data. additional_information[]. message	**string** A message indicating the reason for the N or D result. **Example:** `Name or Date of Birth does not match.`
data. additional_information[]. code	**string** A code number associated with the corresponding message. **Example:** `RD-001`

Sample response


```
{
    "message": "Ok",
    "function": "dvs_reg_by_descent",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": {
        "response_type": "RegistrationbyDescentCertificateResponse",
        "activity_id": "sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d",
        "verification_request_number": "dec306ed-7cff-41c0-b02f-889e430f5309",
        "verification_result_code": "N",
        "additional_information": [
            {
                "message": "Name or Date of Birth does not match.",
                "code": "RD-001"
            }
        ]
    }
}
```


202 **A successful preflight check. A 400 will be returned if the preflight check fails.** (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`
function	**string** The function that was called **Example:** `dvs_aec`

Field	Description
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**null** For preflight checks this is null

Sample response

```
{
    "message": "Ok",
    "function": "dvs_aec",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": null
}
```

Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

Validate Australian visas

POST `/dvs/visa`

Validates an Australian visa using an individual's foreign passport against the Australian Document Verification Service (DVS).

Sandbox environment

When simulating queries in the sandbox environment, a specific set of visas will return a specific result. The response will be determined by the passport number:

DVS sandbox responses for this document

Request body

The document details to validate

Field	Description
`preflight`	**boolean** When set to true, the request will only perform a preflight check to validate the input parameters. No actual verification will be performed. **Example:** `false`

Field	Description
`consent`	**boolean** The individual's consent to perform the check **Example:** `true`
`first_name`	**string** The first name of the document holder as printed on the document **Example:** `John`
`last_name`	**string** The last name of the document holder as printed on the document **Example:** `Smith`
`birth_date`	**string (date)** The date of birth of the document holder **Example:** `1980-01-01`
`passport_number`	**string** The passport number to be validated **Example:** `M1234567`

Sample request

```
{
    "preflight": false,
    "consent": true,
    "first_name": "John",
    "last_name": "Smith",
    "birth_date": "1980-01-01",
    "passport_number": "M1234567"
}
```

Responses

200 **Result of the validation** (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`

Field	Description
`function`	**string** The function that was called **Example:** `dvs_visa`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**object** The check data
`data.` `response_type`	**string** The type of response **Example:** `VisaResponse`
`data.` `activity_id`	**string** A reference ID for this activity. This ID can be used to track the activity in the logs. **Example:** `sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d`
`data.` `verification_request_number`	**string** A unique reference number for this request **Example:** `dec306ed-7cff-41c0-b02f-889e430f5309`
`data.` `verification_result_code`	**string** The result of the verification - `Y` - The document is valid - `N` - The document was not matched - `D` - The document is invalid or not electronically captured - `S` - An error occurred during the check **Enum:** `Y`, `N`, `S`, `D` **Example:** `N`
`data.` `additional_information`	**array of objects** An array of expanded responses returned from the DVSHub or Document Issuer for N or D results

Field	Description
data. additional_information[]. message	**string** A message indicating the reason for the N or D result. **Example:** `Travel document could not be found.`
data. additional_information[]. code	**string** A code number associated with the corresponding message. **Example:** `VI-001`

Sample response

```
{
    "message": "Ok",
    "function": "dvs_visa",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": {
        "response_type": "VisaResponse",
        "activity_id": "sandbox-d06ed58a-b279-4f50-9b6e-5a0a26b4e87d",
        "verification_request_number": "dec306ed-7cff-41c0-b02f-889e430f5309",
        "verification_result_code": "N",
        "additional_information": [
            {
                "message": "Travel document could not be found.",
                "code": "VI-001"
            }
        ]
    }
}
```

202 **A successful preflight check. A 400 will be returned if the preflight check fails.** (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`
function	**string** The function that was called **Example:** `dvs_aec`
api_reference	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Field	Description
`data`	**null** For preflight checks this is null

Sample response

```
{
    "message": "Ok",
    "function": "dvs_aec",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": null
}
```

Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

Check email deliverability

GET `/email_ping`

Tests the deliverability of one or more supplied email addresses. This deliverability check will return a code indicating if the email address is deliverable or not.

> **Note:**
>
> 1. The order of the returned email addresses may not be the same as the supplied order.
> 2. Email addresses will be converted to lower case before being returned by the API.

Interpreting the results

The returned field ping_result gives the actual deliverability result. This indicates if the email address is valid and the mail server has indicated that it will accept delivery for that email account. The ping_result can be undetermined as is the case for greylisting, catch-all or other anti-spam measures.

The returned field info contains a plain test description of the check result. Examples include:

- Deliverable
- Undeliverable: Mailbox not found
- Undeliverable: No DNS entry
- Unknown: Greylisted
- Catch all
- Abuse
- Spam trap

The info field is designed to provide a human readable explanation of the result. The contents of the field can vary for different error scenarios and as such the field should not be matched programmatically.

A further ok_to_send field indicates if the emails should be sent. This differs from the ping_result field which shows if the email can be sent. An example which shows this difference is where the email address is a known spam trap or serial abuse reporter. In this case ping_result will be 'Y' (indicating that the email address is capable of receiving messages) but ok_to_send will be 'N' showing that an email should not be sent to the address.

Whether ping_result or ok_to_send is used to determine deliverability will be dependent on the intent of the API consumer.

Sandbox environment

When simulating queries in the sandbox environment, each email address will return a specific result depending on the email address tld:

Email Type	Response	Description

Any email ending in .asn.au		Simulate a timeout, will never return a result
Any email ending in .com.au:	Y	Simulate a slow result, takes 20 seconds but returns as deliverable
Any email ending in .org.au:	N	Simulate a slow result, takes 20 seconds but returns as not deliverable
Any email ending in .net.au:	U	Simulate a slow result, takes 20 seconds but returns as undetermined
Any email ending in .edu	I	Email is invalid
Any email ending in .org	N	Email is not deliverable
Any email ending in .net	U	Email is undetermined
Any other email (eg .com)	Y	Email is deliverable

Request body

The details of the record to enhance

Field	Description
`emails`	**string** One or more (comma separated) email addresses to be checked **Example:** `test@example.com`

Sample request

```
{
    "emails": "test@example.com"
}
```

Responses

200 **Result of email ping check** (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)** **Example:** `738d9a89-b6fe-4fc1-96be-1389b2a506a2`
`records`	**array of objects** The checked email addresses and their deliverability status

Field	Description
records[]. email	**string** The checked email address **Example:** `test@example.com`
records[]. ping_result	**string** The result of the deliverability check for the email address - `Y` - The email is deliverable - `N` - The email is not deliverable - `U` - The state of the email address cannot be detected - `I` - The email address is invalid **Enum:** `Y`, `N`, `U`, `I` **Example:** `Y`
records[]. ok_to_send	**string** Whether the email address is ok to send to - `Y` - Ok to send - `N` - Not ok to send - `U` - Unable to determine **Enum:** `Y`, `N`, `U` **Example:** `Y`
records[]. info	**string** Plain text description giving additional detail about the result **Example:** `Deliverable`

Sample response


```
{
    "message": "Ok",
    "api_reference": "738d9a89-b6fe-4fc1-96be-1389b2a506a2",
    "records": [
        {
            "email": "test@example.com",
            "ping_result": "Y",
            "ok_to_send": "Y",
            "info": "Deliverable"
        }
    ]
}
```


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

Global Data Check

POST `/globaldata_check`

Validates a person's identity details against the **Global Data Universe** and returns one or more candidate match rows with per-field outcomes.

The endpoint searches by **last_name** and any combination of optional identifiers - `birth_date`, `address`, up to four `phone` numbers, and up to four `email` addresses. Every supplied field is then evaluated against each candidate and reported separately. `last_name` is the only required field.

Match outcomes

For **first_name** and **middle_name** the response uses a graded ladder. The strongest applicable label is returned:

Label	Meaning
`match`	Exact equality after normalisation (case- and diacritic-insensitive).
`alias_match`	The two names share a known nickname / alias group (for example `Bob` / `Robert`, `Jenny` / `Jennifer`, `Tony` / `Antonio`).
`partial_match`	One side is exactly a single-letter initial that matches the leading letter of the other side (for example `P` vs `Peter`). Modelled on universe records that only have a first-letter initial captured. Longer-form abbreviations like `Pete` vs `Peter` are handled by `alias_match` instead.
`fuzzy_match`	The two names share a phonetic (metaphone) code and the same leading letter.
`no_match`	None of the above.
`no_record`	(Middle name only.) The caller supplied a middle name but the candidate has no middle name on file.
`not_used`	The caller supplied this field but it was dropped because it failed validation and `ignore_errors=true`.

Fields that the caller did not supply are **omitted from each candidate row entirely**. The `address` rollup key appears whenever any address part was supplied; individual address component keys (`street_address`, `suburb`, `state`, `postcode`) appear only when the caller actually supplied that part. When no address was supplied at all, the rollup and all four component keys are omitted together.

Which match types are accepted is controlled per name field via `first_name_matching`, `middle_name_matching`, and `last_name_matching` (defaults to `["exact"]` for each).

For **dob** the outcomes are `match`, `match_year` (year matches but day/month don't), `match_day_month_reversal` (year matches, day and month swapped), `no_record` (candidate has no DOB on file), `no_match`, or `not_used`.

For **phone** and **email**, each supplied slot is evaluated independently against the candidate's full set of phones / emails. If you supply `phone` and `phone3` (for example), the response carries a separate `phone` outcome and a separate `phone3` outcome, each `match` / `no_match` based on whether that specific value matches any of the candidate's phones on file. Slots

that the caller did not supply are omitted from the response. The same per-slot evaluation applies to `email`, `email2`, `email3`, and `email4`.

Australian phone numbers are accepted in either `0NSN` form (10 digits starting with `0`) or the equivalent E.164 form (`+61` followed by 9 digits). The leading `+` on the E.164 form is optional - `61400123456` is accepted as well as `+61400123456`, which avoids spurious rejections of numbers that have round-tripped through Excel / CSV imports that silently strip the `+`. All three shapes are normalised to the `0NSN` form internally before comparison.

Address matching

When an address is supplied, the response includes both an `address` rollup field and the four individual address component fields (`street_address`, `suburb`, `state`, `postcode`). The rollup is a single-glance summary of how well the supplied address matched the candidate; the component fields tell you exactly which parts agreed.

The rollup is derived from the components as follows:

`address` rollup	Meaning
`match`	Every supplied address component matches the candidate (parts the caller did not supply are not considered).
`match_street`	Same street number on the same street name (street-type variations like `ST` vs `RD` are allowed) but at least one of the other supplied parts (suburb / state / postcode) does not match.
`match_locality`	The street did not match, but the state matches and at least one of the suburb or postcode also matches. State must be supplied for `match_locality` to fire.
`no_match`	None of the above; the supplied address did not align with the candidate at any of the levels above.

The component fields each return `match` / `no_match` independently and are returned alongside the rollup so you can see which specific parts contributed to it. For example, `address: match_street` plus `street_address: match` and `postcode: no_match` tells you the candidate lives on the same street but in a different postcode area.

When the caller supplies only a subset of address parts (e.g. just `postcode`, or `suburb` + `postcode`), only those parts are evaluated and reported - the unsupplied address keys are omitted from the row entirely. The rollup `match` therefore means "every supplied part matched", not "all four parts matched".

Whichever input form was used (`gnaf_id`, `full_address`, or any combination of address parts), the supplied components are evaluated directly. With `gnaf_id` / `full_address` the address is resolved to canonical components first; with the parts form the supplied values are used as-is.

A given person may have several addresses on file (current and historical). `match_results` always returns **one row per person** - we do not duplicate the same person once per address. When evaluating, we score every address the person has against the supplied address and report the outcome for the **best-matching** one. Ties on rollup outcome are broken by preferring the most recently active address. For example, if a person has both an old `match_locality` address and a current `match` address against your input, you will see `address: match` in the response.

Best-candidate selection

Candidates are ranked in three steps, applied in order:

1. **Strong-discriminator hits rank first.** A candidate that produces an exact match against any of `phone`, `email`, full `dob`, or the full `address` (the rollup at `match`) outranks any candidate that does not, regardless of how well the other candidate scores on names alone. `dob: match_day_month_reversal` also counts as a strong-discriminator hit, **but only when the candidate also has `first_name` of `match` / `alias_match` AND `last_name` of `match`** - without name

corroboration, a reversed DOB is treated as a numeric coincidence and does not gate the candidate into the strong-discriminator group.

2. **Exact last-name wins within the strong-discriminator group.** Among candidates in the same group from step 1, candidates with `last_name: match` rank above candidates with `last_name: fuzzy_match` - a fuzzy last-name candidate never displaces an exact last-name candidate for the top slot.
3. **Weighted score breaks remaining ties.** Per-field weights are summed and the highest sum wins:

Field	match	partial outcomes
`first_name`	20	`alias_match` 16, `partial_match` 10, `fuzzy_match` 6
`middle_name`	1	`alias_match` 1, `partial_match` 1
`last_name`	12	`fuzzy_match` 8
`dob`	10	`match_day_month_reversal` 7, `match_year` 4
`address`	8	`match_street` 5, `match_locality` 3
each `phone` slot (`phone` / `phone2` / `phone3` / `phone4`)	4	-
each `email` slot (`email` / `email2` / `email3` / `email4`)	2	-

Each candidate row reports the per-field outcomes directly. Callers wanting to drive a routing rule should branch on the specific outcomes that matter for their use case (for example, `last_name: match` plus `dob: match` plus any phone match for high-confidence auto-approval) rather than on a single aggregate score.

Single vs multiple candidates

`match_results` is always an array, with one row per distinct **person** in the universe (never one row per person-and-address). By default the array contains the single best candidate (length 0 or 1). Set `return_multiple_candidates: true` to receive multiple candidates ordered best-first, with the same row schema.

When `return_multiple_candidates: true` is set, the response is capped at the **top 5 candidates by score**. If your search matches more than 5 records, only the strongest 5 are returned. Narrow your request (add a DOB, a phone, or an address) to lift the strongest candidates above the cap.

Two universe records that produce **identical per-field outcomes** against the request collapse into a single row in `match_results`. The count of rows therefore reflects the number of distinct outcome maps, not the number of underlying universe records considered. Adding more identifiers to the request makes candidates differ in their per-field outcomes and surface as separate rows.

`ignore_errors`

Useful when the data you are checking is not perfectly clean - for example, when the source system you are pulling from contains the occasional malformed phone number, invalid email, or junk date of birth.

By default, any individual field that fails validation will cause the whole request to be rejected with a `400` error, leaving you to either retry the call with the bad fields omitted or fix the data on your side first.

Setting `ignore_errors: true` removes that round-trip: the endpoint silently drops any individual field whose value fails validation (a malformed `birth_date` , an invalid phone number, an unresolvable address, etc.) and runs the match against

whatever valid input remains. The dropped fields are reported back in the response as `not_used` so you can see exactly which inputs were ignored. Only field-level value problems are soft-ignored - the request still has to be structurally well-formed (correct types, `last_name` present).

`include_deceased`

Records marked as deceased are excluded from the candidate pool by default. Setting `include_deceased: true` adds them back in.

Note that the response does **not** include any indicator of whether a returned candidate was deceased - matched candidates are returned in the same shape regardless. Most use cases (KYC, identity verification, contact validation) should leave this off; only enable it when you have a specific need to match against deceased records and your downstream workflow handles them appropriately.

Sandbox environment

When simulating queries in the sandbox environment, the following records can be searched for:

First Name	Middle Name	Last Name	Birth Date	Address	Phone	Email
JOHN	ANDREW	SMITH	1998-03-21	20 HARDY ST, LILYDALE 3140 VIC	0399999999, 0491222111	jsmith1998@example.com
ROBERT	PATRICK	BROWN	1971-03-18	8 WALTHAM ST, RICHMOND 3121 VIC	0399998888	
MARY	SALLY	JONES	1989-08-12	35 YARALLA ST, CONCORD WEST 2138 NSW	0299995000, 0491222444	mary_jones89@example.com
MARION		FILEWOOD	1955-09-07	375 ARGENT ST, BROKEN HILL 2880 NSW	0412024869, 0880885999	m47b7@rev-zone.net
AARON	ALBERT	FILEWOOD	1949-03-11	375 ARGENT ST, BROKEN HILL 2880 NSW	0880885999	
RUTH		WOLFE	2000-05-20	375 ARGENT ST, BROKEN HILL 2880 NSW	0455963579	
MATTHEW		SMITH	1999-10-21	(no address on file)	(no phone on file)	(no email on file)
JOAN		EVANS	1996-07-06	(no address on file)	(no phone on file)	(no email on file)
ROBERT		BROWN	(no DOB on file)	6 WATERVIEW CL, PORT MACQUARIE 2444 NSW	0491222333	

First Name	Middle Name	Last Name	Birth Date	Address	Phone	Email
R		BROWN	(no DOB on file)	22 BRABYN ST, WINDSOR 2756 NSW	(no phone on file)	RP_BROWN71@EXAMPLE.COM

Useful demonstrations against the sandbox data:

- **alias_match** - search `first_name = "Bob"`, `last_name = "Brown"`, `birth_date = "1971-03-18"` with `first_name_matching = ["exact","alias"]` matches Robert Brown with `first_name: alias_match`.
- **partial_match** - search `first_name = "Robert"`, `last_name = "Brown"` with `first_name_matching = ["exact","partial"]` may match the `R BROWN` record with `first_name: partial_match`.
- **fuzzy_match** - search `first_name = "Mathew"`, `last_name = "Smith"`, `birth_date = "1999-10-21"` with `first_name_matching = ["exact","fuzzy"]` matches Matthew Smith with `first_name: fuzzy_match`.
- **address as identifier** - search `last_name = "Smith"` plus the full address `20 HARDY ST, LILYDALE 3140 VIC` (no other identifier) matches John Smith.
- **last-name-only** - search `last_name = "Evans"` with `return_multiple_candidates = true` returns every Evans in the universe, including Joan Evans.
- **no_record DOB** - search `first_name = "Robert"`, `last_name = "Brown"`, `phone = "0491222333"`, `birth_date = "1985-06-04"` matches the second Robert Brown record by phone, with `dob: no_record` because that record has no DOB on file.

Request body

The details of the person to validate.

Field	Description
`first_name`	**string or null** The first name of the individual. Optional - omit, leave blank, or pass `null` if unknown. **Example:** `John`
`middle_name`	**string or null** The middle name of the individual. **Example:** `James`
`last_name` REQUIRED	**string** The last name of the individual. Required. **Example:** `Doe`
`birth_date`	**string (date) or null** The date of birth of the individual in `YYYY-MM-DD` format. **Example:** `1990-01-01`

Field	Description
`gnaf_id`	**string or null** A GNAF address identifier (e.g. `GAVIC421647320`). Both no-separator (`GAVIC421647320`, `GANSW710277749`) and underscore-separated (`GANT_717247498`, `GASA_414917272`) forms are accepted; the underscore is part of the source identifier and should be preserved rather than stripped. Mutually exclusive with `full_address` and the address-parts fields. **Example:** `GAVIC421647320`
`full_address`	**string or null** The address as a single free-text string. Mutually exclusive with `gnaf_id` and the address-parts fields. **Example:** `4/123 Fake Street, Fakeville VIC 3987`
`street_address`	**string or null** The street line of the address. Any combination of `street_address` / `suburb` / `state` / `postcode` is accepted - the parts do not all have to be supplied together. `gnaf_id` / `full_address` must not be supplied alongside any address part. The response only evaluates and reports per-field outcomes for the parts the caller actually supplied. **Example:** `4/123 Fake Street`
`suburb`	**string or null** The suburb of the address. May be supplied on its own or in combination with any other address parts. **Example:** `Fakeville`
`state`	**string or null** The Australian state of the address. May be supplied on its own or in combination with any other address parts. **Enum:** `NSW`, `VIC`, `QLD`, `SA`, `WA`, `TAS`, `NT`, `ACT` **Example:** `VIC`
`postcode`	**string or null** The postcode of the address. May be supplied on its own or in combination with any other address parts. **Example:** `3987`

Field	Description
`phone`	**string or null** A phone number for the individual. Accepts either `0NSN` (10 digits starting with `0`) or the equivalent E.164 form (`+61` followed by 9 digits). The leading `+` is optional - `61400123456` is treated the same as `+61400123456`. **Example:** `0400123456`
`phone2`	**string or null** An additional phone number. Same format as `phone`.
`phone3`	**string or null** An additional phone number. Same format as `phone`.
`phone4`	**string or null** An additional phone number. Same format as `phone`.
`email`	**string or null** An email address for the individual. **Example:** `john@example.com`
`email2`	**string or null** An additional email address.
`email3`	**string or null** An additional email address.
`email4`	**string or null** An additional email address.
`return_multiple_candidates`	**boolean** When `true`, return up to the top 5 candidates ordered best-first. When `false` (the default), return only the single best candidate. **Default:** `false` **Example:** `false`

Field	Description
`ignore_errors`	**boolean** When `true`, supplied fields that fail validation (bad DOB, invalid phone/email, unresolvable address) are dropped from the search and reported as `not_used` in the response, rather than producing a 400. **Default:** `false` **Example:** `false`
`include_deceased`	**boolean** When `true`, deceased records are included in the candidate pool. Defaults to `false`. **Default:** `false` **Example:** `false`
`first_name_matching`	**array of strings** Allow-list of accepted match types for `first_name`. Defaults to `["exact"]`. Add `alias`, `partial`, or `fuzzy` to also accept the corresponding outcome on the response ladder. **Enum:** `exact`, `alias`, `partial`, `fuzzy`
`middle_name_matching`	**array of strings** Allow-list of accepted match types for `middle_name`. Defaults to `["exact"]`. Middle name does not filter candidates - a missing or mismatched middle name will still return a candidate; this list only controls which positive outcomes are recognised. **Enum:** `exact`, `alias`, `partial`, `fuzzy`
`last_name_matching`	**array of strings** Allow-list of accepted match types for `last_name`. Defaults to `["exact"]`. Adding `fuzzy` widens the SQL candidate search to include records whose last-name metaphone matches and whose last name starts with the same letter. **Enum:** `exact`, `fuzzy`

Sample request

```
{
    "first_name": "John",
    "middle_name": "James",
    "last_name": "Doe",
    "birth_date": "1990-01-01",
    "gnaf_id": "GAVIC421647320",
    "full_address": "4/123 Fake Street, Fakeville VIC 3987",
    "street_address": "4/123 Fake Street",
    "suburb": "Fakeville",
```

```
    "state": "VIC",
    "postcode": "3987",
    "phone": "0400123456",
    "phone2": "string",
    "phone3": "string",
    "phone4": "string",
    "email": "john@example.com",
    "email2": "string",
    "email3": "string",
    "email4": "string",
    "return_multiple_candidates": false,
    "ignore_errors": false,
    "include_deceased": false,
    "first_name_matching": [
        "exact",
        "alias"
    ],
    "middle_name_matching": [
        "exact"
    ],
    "last_name_matching": [
        "exact"
    ]
}
```

Responses

200 **Result of the validation check.** (application/json)

Field	Description
`message` REQUIRED	**string** A message indicating the result of the request. This will be `Ok` if the request was successful. **Example:** `Ok`
`api_reference` REQUIRED	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Field	Description
`match_results` REQUIRED	**array of objects** Candidate rows returned by the search, **one row per matched person** (never one row per person-and-address). Always an array. When `return_multiple_candidates` is `false` (the default) the array contains zero or one row (the single best candidate); when `true` it contains up to 5 candidates ordered best-first. When the request supplied an address and the candidate has several addresses on file, the address-component outcomes (`address` rollup plus any supplied `street_address` / `suburb` / `state` / `postcode` parts) reflect the **best-matching** address from the candidate's address history, with the most recently active address used to break ties. Each row only includes the fields that were actually evaluated. Fields that the caller did not supply are omitted from the row entirely (they are not returned as `null` or `not_provided`). The `address` rollup appears whenever any address part was supplied; the individual `street_address` / `suburb` / `state` / `postcode` keys appear only when the caller supplied that part. When no address was supplied at all, the rollup and all four component keys are omitted together.
`match_results[].` `first_name`	**string** First name match outcome (uses the graded ladder). Omitted when not supplied in the request. **Enum:** `match` , `alias_match` , `partial_match` , `fuzzy_match` , `no_match` , `not_used` **Example:** `match`
`match_results[].` `middle_name`	**string** Middle name match outcome. `no_record` means a middle name was supplied but the candidate has none on file. Omitted when not supplied in the request. **Enum:** `match` , `alias_match` , `partial_match` , `fuzzy_match` , `no_record` , `no_match` , `not_used` **Example:** `match`
`match_results[].` `last_name`	**string** Last name match outcome. **Enum:** `match` , `fuzzy_match` , `no_match` , `not_used` **Example:** `match`

Field	Description
match_results[]. dob	**string** Birth date match outcome. `match_year` and `match_day_month_reversal` are returned for partial date matches; `no_record` means the candidate has no DOB on file. Omitted when not supplied in the request. **Enum:** `match`, `match_year`, `match_day_month_reversal`, `no_record`, `no_match`, `not_used` **Example:** `match`
match_results[]. address	**string** Roll-up address outcome derived from the four address components. `match_street` requires the same street number on the same street name (street-type variations allowed); `match_locality` requires state to match (if supplied) and at least one of suburb or postcode to match. Omitted when no address was supplied in the request. **Enum:** `match`, `match_street`, `match_locality`, `no_match`, `not_used` **Example:** `match`
match_results[]. street_address	**string** Street-line component of the address comparison. Omitted when no address was supplied. **Enum:** `match`, `no_match`, `not_used` **Example:** `match`
match_results[]. suburb	**string** Suburb component of the address comparison. Omitted when no address was supplied. **Enum:** `match`, `no_match`, `not_used` **Example:** `match`
match_results[]. state	**string** State component of the address comparison. Omitted when no address was supplied. **Enum:** `match`, `no_match`, `not_used` **Example:** `match`

Field	Description
match_results[]. postcode	**string** Postcode component of the address comparison. Omitted when no address was supplied. **Enum:** `match`, `no_match`, `not_used` **Example:** `match`
match_results[]. phone	**string** Match outcome for the supplied `phone` value: `match` if any of the candidate's phones on file equals it, otherwise `no_match`. Omitted when `phone` was not supplied in the request. **Enum:** `match`, `no_match`, `not_used` **Example:** `match`
match_results[]. phone2	**string** Match outcome for the supplied `phone2` value, evaluated independently of the other phone slots. Omitted when `phone2` was not supplied. **Enum:** `match`, `no_match`, `not_used` **Example:** `match`
match_results[]. phone3	**string** Match outcome for the supplied `phone3` value, evaluated independently of the other phone slots. Omitted when `phone3` was not supplied. **Enum:** `match`, `no_match`, `not_used` **Example:** `match`
match_results[]. phone4	**string** Match outcome for the supplied `phone4` value, evaluated independently of the other phone slots. Omitted when `phone4` was not supplied. **Enum:** `match`, `no_match`, `not_used` **Example:** `match`

Field	Description
match_results[]. email	**string** Match outcome for the supplied `email` value: `match` if any of the candidate's emails on file equals it (case-insensitive), otherwise `no_match`. Omitted when `email` was not supplied in the request. **Enum:** `match`, `no_match`, `not_used` **Example:** `no_match`
match_results[]. email2	**string** Match outcome for the supplied `email2` value, evaluated independently of the other email slots. Omitted when `email2` was not supplied. **Enum:** `match`, `no_match`, `not_used` **Example:** `match`
match_results[]. email3	**string** Match outcome for the supplied `email3` value, evaluated independently of the other email slots. Omitted when `email3` was not supplied. **Enum:** `match`, `no_match`, `not_used` **Example:** `match`
match_results[]. email4	**string** Match outcome for the supplied `email4` value, evaluated independently of the other email slots. Omitted when `email4` was not supplied. **Enum:** `match`, `no_match`, `not_used` **Example:** `match`

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "match_results": [
        {
            "first_name": "match",
            "middle_name": "match",
            "last_name": "match",
            "dob": "match",
            "address": "match",
            "street_address": "match",
            "suburb": "match",
            "state": "match",
            "postcode": "match",
            "phone": "match",
            "phone2": "match",
```

```
            "phone3": "match",
            "phone4": "match",
            "email": "no_match",
            "email2": "match",
            "email3": "match",
            "email4": "match"
        }
    ]
}
```

500 **Unexpected error.** (application/json)

Field	Description
`message`	**string** A message indicating the error. **Example:** `An error occurred while processing your request.`

Sample response

```
{
    "message": "An error occurred while processing your request."
}
```

Standard error responses: 400, 402, 403 (see Common error responses).

Extract ID Document Data & Face Image

POST `/id_extract`

Performs OCR / data extraction on an identity document image and (when available) returns parsed fields plus a cropped face portrait (`photo`). Supported document types are:

- `passport`
- `licence` (Australian driver licence – requires front & back images)
- `medicare` (Australian Medicare card – front image only)

The response structure under `result` varies by `document_type` (see schema & examples below). All successful responses include a base64-encoded face image when a face is detected (`photo`).

Inputs

Supply images as base64 **data URI** strings (e.g. `...`) or raw base64. Only JPEG format is accepted for this endpoint. Maximum file size per image is 16 MB. For `licence` , both front (`document_photo`) and back (`document_back`) images are required.

Field	Required	Type	Notes
`document_type`	Yes	enum	One of `passport` , `licence` , `medicare`
`document_photo`	Yes	string	Base64 JPEG (front)
`document_back`	Conditionally	string	Base64 JPEG (back) – required when `document_type = licence`
`photo_border_percent`	No	integer	Border padding around the extracted face photo as a percentage of the detected face box (0-50, default 15)

Output (result object)

Depending on `document_type` , typical extracted fields include:

- Passport: `first_name` , `last_name` , `birth_date` (YYYY-MM-DD), `expiry_date` (YYYY-MM-DD), `gender` , `passport_number` , `country` , `photo` , `face_occluded` .
- Licence: `first_name` , `middle_name` , `last_name` , `birth_date` , `address` , `expiry_date` (YYYY-MM-DD), `conditions` , `licence_state` , `card_number` , `photo` , `face_occluded` .
- Medicare: `card_color` , `card_number` , `expiry_date` (YYYY-MM or YYYY-MM-DD), `people` (array of holders with `ref_number` & `name` lines).

Additional or fewer fields may appear depending on document quality and internal extraction logic.

Error Handling

- Validation errors (missing / invalid fields, size limits, unsupported mime) return HTTP 400 with a descriptive `message` .

- Known processing errors from the document analysis engine (e.g. `no_face_found`) return HTTP 400 with that message.
- Unknown processing failures return HTTP 500 with `An error occurred while processing the document.`
- Entitlement / auth / rate limit errors use standard Global Data API response references.

Auditing & Privacy

For compliance, sensitive fields and raw images are obfuscated in internal audit logs (masked with `X`). The API response itself returns the actual extracted values so you should persist / redact them according to your own data handling policies.

Request body

ID document type and base64-encoded image(s) to extract.

Field	Description
`document_type` REQUIRED	**string** Type of document supplied. **Enum:** `passport`, `licence`, `medicare` **Example:** `passport`
`document_photo` REQUIRED	**string** Base64 (optionally data URI) JPEG image of the front of the document. **Example:** `...`
`document_back`	**string** Base64 (optionally data URI) JPEG image of the back of the document (licence only). **Example:** `...`
`photo_border_percent`	**integer [0..50]** Border padding added around the detected face box in the extracted face photo (`photo`), expressed as a percentage of the detected face box dimensions. Use a larger value to keep more spacing around the face in the cropped photo. Not applicable to `medicare`. **Default:** `15` **Example:** `25`

Sample request

```
{
    "document_type": "passport",
    "document_photo": "...",
    "document_back": "...",
    "photo_border_percent": 25
}
```

Responses

200 **Document successfully processed and data extracted.** (application/json)

Field	Description
`message`	**string** Always `Ok` on success. **Example:** `Ok`
`api_reference`	**string (uuid)** Unique audit reference for this request. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`result`	**one of 3 shapes** Extracted document data (structure depends on `document_type`).

Field	Description

PASSPORT EXTRACTION RESULT

Field	Description
photo	string Base64 encoded cropped face image. Example: `iVBORw0KGgoAAAANSUhEUg...`
face_occluded	one of 2 shapes Whether face occlusion was detected for the cropped face image. Returns `false` when the face is not considered occluded. When the face has been considered occluded, this field returns a percentage confidence that the face is occluded. This is not an indication of the proportion of the face that is occluded. BOOLEAN boolean, one of `false`, e.g. `false` NUMBER number (float), e.g. `97.5`
first_name	string Example: `JOHN MICHAEL`
last_name	string Example: `SMITH`
birth_date	string (date) Example: `1990-01-01`
expiry_date	string (date) Example: `2030-01-01`
gender	string Example: `M`
passport_number	string Example: `M1234567A`
country	string Example: `AUS`

DRIVER LICENCE EXTRACTION RESULT

Field	Description
photo	string Base64 encoded cropped face image.

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "result": {
        "photo": "iVBORw0KGgoAAAANSUhEUg...",
        "face_occluded": false,
        "first_name": "JOHN MICHAEL",
        "last_name": "SMITH",
        "birth_date": "1990-01-01",
        "expiry_date": "2030-01-01",
        "gender": "M",
        "passport_number": "M1234567A",
        "country": "AUS"
    }
}
```


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

Create an ID Pass

POST `/idpass/register`

ID Pass

Please review the full ID Pass documentation here:

ID Pass documentation

ID Pass is powerful and secure platform for verifying customer identities in compliance with KYC and AML regulations. The API provides an efficient way to integrate identity verification workflows into your applications, ensuring smooth user experiences while maintaining the highest security and data privacy standards.

- **Document Verification:** Supports Australian Driver's Licenses, Passports, Foreign Passports with an Australian Visa, Centrelink Cards, Medicare Cards, Australian Birth Certificates and New Zealand Driver Licences. Integrates with the Australian Document Verification Service (DVS) for Australian documents and with the New Zealand licence issuer for New Zealand Driver Licences.
- **Biometric Face Verification:** Liveness detection ensures authenticity. Matches selfies to the ID document for added security.
- **Secure and Configurable:** Fully encrypted data storage and transfer. Configurable link validity, retention settings, and webhook notifications.

API Workflow

- **Create an ID Pass:** Generate a secure verification link for your users with customizable configurations. Control parameters such as allowed document types, liveness checks, and link expiration.
- **Monitor Progress:** Use webhooks to track verification events such as "in_progress," "complete," or "failed."
- **Retrieve Results:** Access detailed verification results, including document validation, biometric comparison, and logs. Optionally decrypt sensitive information locally using a provided cipher key.

Sandbox environment

When creating an ID Pass in the sandbox environment, the actual DVS checks for the identity documents will be simulated and the specific result (pass, fail, etc) will determined by the ID number in the same way as other sandbox DVS queries.

Sample documents are available for you to test with the ID Pass API:

- DVS sample documents

Data encryption

All data stored for the ID Pass is encrypted at rest to ensure the security and privacy of sensitive information. Additionally, particularly sensitive fields are encrypted using a unique cipher key that is generated for each ID Pass.

The cipher key is securely included in the ID Pass register response and is deleted from our systems once the ID Pass is completed by the end user.

This design ensures that we have no capability to decrypt the data once the cipher key has been deleted.

Important notes on encryption

- **Irretrievability of Data:** If the cipher key is lost, there is no way to recover or decrypt the associated ID Pass data. It is you responsibility to securely store and manage the cipher key.
- **Compliance:** This encryption approach ensures compliance with privacy regulations by minimizing exposure of sensitive data.
- **Decryption:** To access sensitive fields (e.g., validation results, documents, or images), you must supply the cipher key when retrieving the ID Pass data. Without the cipher key, encrypted data will be returned for local decryption.

Request body

The configuration for the ID Pass to create

Field	Description
`link_validity_days`	**integer [1..14] or null** Number of days for which the link should remain available **Default:** `7` **Example:** `3`
`image_retention_days`	**integer [0..365] or null** Number of days for which the images from the check will be stored, up to a maximum of 365 days. The stored images include the probe image from the face liveness check (if liveness was requested) and the front / back /photo images from any identity documents used during the check. A setting of 0 will cause the images to be deleted once the check has been completed and prevents the ID Pass Details endpoint from ever returning image data for the check. Images are deleted automatically once the retention period has passed. **Default:** `0` **Example:** `5`

Field	Description
`check_liveness`	**boolean or null** Whether a face liveness check should be performed. **Default:** `false` **Example:** `true`
`document_1_allowed_types`	**array of strings or null** The types of documents which are allowed for the first document check. - `licence` - Australian Drivers Licence - `passport` - Australian Passport - `medicare` - Medicare Card - `visa` - Foreign Passport (for Australian Visa holders) - `centrelink` - Centrelink Card (manual entry, no OCR) - `birth_certificate` - Australian Birth Certificate (manual entry, no OCR) - `nz_licence` - New Zealand Driver Licence (front and back). **Enum:** `licence`, `passport`, `medicare`, `visa`, `centrelink`, `birth_certificate`, `nz_licence`
`document_2_allowed_types`	**array of strings or null** The types of documents which are allowed for the second document check. - `licence` - Australian Drivers Licence - `passport` - Australian Passport - `medicare` - Medicare Card - `visa` - Foreign Passport (for Australian Visa holders) - `centrelink` - Centrelink Card (manual entry, no OCR) - `birth_certificate` - Australian Birth Certificate (manual entry, no OCR) - `nz_licence` - New Zealand Driver Licence (front and back). **Enum:** `licence`, `passport`, `medicare`, `visa`, `centrelink`, `birth_certificate`, `nz_licence`

Field	Description
`document_3_allowed_types`	**array of strings or null** The types of documents which are allowed for the third document check. - `licence` - Australian Drivers Licence - `passport` - Australian Passport - `medicare` - Medicare Card - `visa` - Foreign Passport (for Australian Visa holders) - `centrelink` - Centrelink Card (manual entry, no OCR) - `birth_certificate` - Australian Birth Certificate (manual entry, no OCR) - `nz_licence` - New Zealand Driver Licence (front and back). **Enum:** `licence`, `passport`, `medicare`, `visa`, `centrelink`, `birth_certificate`, `nz_licence`
`document_verification`	**string or null** The verification strategy to use for identity documents. - `dvs` - Full verification of Australian documents via the Australian Document Verification Service (DVS); a New Zealand Driver Licence is verified with the New Zealand licence issuer and returned in the same result shape. Requires a DVS Identity (OAC) on the account for live environments. - `basic` - Basic format validation only. Checks that document fields match the expected format (e.g. card number patterns, Medicare checksums) without performing a DVS check. Does not require a DVS Identity. - `idsp` - Identity Service Provider mode. Document verification is performed by Global Data on the customer's behalf. Requires liveness detection to be enabled and the account to be enrolled as an IDSP Service Client. Does not require a DVS Identity (OAC) on the account. **Enum:** `dvs`, `basic`, `idsp` **Default:** `dvs` **Example:** `dvs`
`require_id_photo`	**boolean or null** Whether the user should be required to supply an ID photo. **Default:** `false` **Example:** `true`

Field	Description
`id_photo_purpose`	**string [3..255 characters] or null** The purpose of the ID photo. This text will be included on the page where the user is asked to supply the ID photo. For example, passing "the SampleCo Membership Card" will display the text "This image will be used for the SampleCo Membership Card" on the page where the user is asked to supply the ID photo. Note that the id_photo_purpose is required if require_id_photo is true and must be between 3 and 255 characters. **Example:** `the SampleCo Membership Card`
`webhook_url`	**string (url) or null** Webhook URL which the ID Pass system will call to notify events **Example:** `https://example.com/webhook`
`webhook_events`	**array of strings or null** The events for which the webhook should be called - `opened` - Opened - `in_progress` - In Progress - `complete` - Completed - `expired` - Expired - `failed` - Failed - `cancelled` - Cancelled **Enum:** `opened`, `in_progress`, `complete`, `expired`, `failed`, `cancelled`
`return_url`	**string (url) or null** URL to link to from the ID Pass complete page. This can be used to return the user to your application or website after they complete the ID Pass.
`requested_identity`	**object or null** Optional object containing your customer's identity details to be verified. At least one of `first_name` or `last_name` must be provided when this object is supplied. `middle_name` alone is not valid. During verification comparison, a submitted single hyphen (`-`) for first or last name is treated as missing for matching against missing requested identity values.
`requested_identity.` `first_name`	**string or null** Optional first name of your customer to be verified. Provide `first_name`, `last_name`, or both. **Example:** `John`

Field	Description
`requested_identity.middle_name`	**string or null** Optional middle name of your customer to be verified. This field does not satisfy the required-name rule on its own. **Example:** `Andrew`
`requested_identity.last_name`	**string or null** Optional last name of your customer to be verified. Provide `first_name`, `last_name`, or both. **Example:** `Johnson`
`requested_identity.date_of_birth`	**string (date) or null** Optional date of birth of your customer to be verified. If provided, this is matched against the verified identity date of birth. If omitted or null, only name matching is performed and you should verify the returned `verified_identity.date_of_birth` yourself. **Example:** `1980-01-01`

Sample request

```
{
    "link_validity_days": 3,
    "image_retention_days": 5,
    "check_liveness": true,
    "document_1_allowed_types": [
        "licence",
        "passport"
    ],
    "document_2_allowed_types": [
        "licence",
        "passport"
    ],
    "document_3_allowed_types": [
        "licence",
        "passport"
    ],
    "document_verification": "dvs",
    "require_id_photo": true,
    "id_photo_purpose": "the SampleCo Membership Card",
    "webhook_url": "https://example.com/webhook",
    "webhook_events": [
        "opened"
    ],
    "return_url": "https://example.com",
    "requested_identity": {
        "first_name": "John",
        "middle_name": "Andrew",
        "last_name": "Johnson",
        "date_of_birth": "1980-01-01"
    }
```

```
}
```

Responses

200 **Details of the newly created ID Pass** (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 for support queries. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`id`	**string (uuid)** The identifier for this ID Pass.
`cipher_key`	**string** The cipher key fot this ID Pass. All PII will be encrypted with this key. **Example:** `IHm81bkcMsTL7J1ilxhDNE59+p5OQkvUJ3mZQhlUNHA=`
`link`	**string (url)** The ID Pass link to be provided to your user **Example:** `https://idpass.globaldata.net.au/idpass?token=abcd123456`

Sample response

```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "id": "00000000-0000-0000-0000-000000000000",
    "cipher_key": "IHm81bkcMsTL7J1ilxhDNE59+p5OQkvUJ3mZQhlUNHA=",
    "link": "https://idpass.globaldata.net.au/idpass?token=abcd123456"
}
```

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

Read an ID Pass

POST `/idpass/details`

Read the details from an ID Pass

Please review the full ID Pass documentation here:

ID Pass documentation

Data encryption

All data stored for the ID Pass is encrypted at rest to ensure the security and privacy of sensitive information. Additionally, particularly sensitive fields are encrypted using a unique cipher key that is generated for each ID Pass.

The cipher key is securely included in the ID Pass register response and is deleted from our systems once the ID Pass is completed by the end user.

This design ensures that we have no capability to decrypt the data once the cipher key has been deleted.

Important notes on encryption

- **Irretrievability of Data:** If the cipher key is lost, there is no way to recover or decrypt the associated ID Pass data. It is you responsibility to securely store and manage the cipher key.
- **Compliance:** This encryption approach ensures compliance with privacy regulations by minimizing exposure of sensitive data.
- **Decryption:** To access sensitive fields (e.g., validation results, documents, or images), you must supply the cipher key when retrieving the ID Pass data. Without the cipher key, encrypted data will be returned for local decryption.

Recalling Encrypted Data

You can include the cipher key (returned during ID Pass creation) in your request to handle encrypted data as follows:

- **With Cipher Key:** If the cipher key is provided, sensitive fields will be decrypted and returned in plain text.
- **Without Cipher Key:** If no cipher key is supplied, the encrypted data will be returned for local decryption using the provided cipher key.
- **Compliance Considerations** Ensure your decision to include the cipher key aligns with your compliance and security policies.

Fields Encrypted

- **Validation Results:** Field: validation_summary
- **Requested Identity:** Field: config.requested_identity
- **Document Details:** Field: documents
- **Captured Images:** Field: images

Encryption and Decryption:

The data is encrypted using the AES-256-CBC cipher. Use the following pseudocode to decrypt the data locally:

Decrypting Data

```
function decryptData(encryptedData, cipherKey, iv):
  // Decode the cipher key and initialization vector (IV)
  decodedKey = base64Decode(cipherKey)
  decodedIV = base64Decode(iv)

  // Decrypt the data using AES-256-CBC
  decryptedData = AES256_CBC_Decrypt(encryptedData, decodedKey, decodedIV)

  return decryptedData
```

Validating Integrity with MAC

```
function validateMAC(data, iv, cipherKey):
  // Concatenate the IV and encrypted data
  combinedData = iv + data

  // Generate a MAC using HMAC-SHA256
  mac = HMAC_SHA256(combinedData, base64Decode(cipherKey))

  return mac
```

Request body

The details of the ID Pass to retrieve

Field	Description
`id`	**string (uuid)** The id of the ID Pass to retrieve **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`cipher_key`	**string or null** The cipher_key for this ID Pass. If the cipher_key is provided then the response will be read from the system and decrypted before returning. Alternatively, if no cipher_key is supplied, then the encrypted data will be returned and can be decrypted locally. **Example:** `IHm81bkcMsTL7J1ilxhDNE59+p5OQkvUJ3mZQhlUNHA=`
`return_logs`	**boolean or null** Optionally return the debug log from the ID Pass. The log contains information about each step which was attempted by the user and can be useful to help understand user issues. Logs are returned in the logs response parameter. **Example:** `true`

Field	Description
`return_images`	**boolean or null** Optionally return the images captured during the ID Pass process. This will include the probe image extracted from the liveness face verification (if requested) and the front / back and photo images from the documents which were validated as well as the optional ID photo image. Note that images are only stored for the period in the image_retention_days parameter from the ID Pass creation. If image retention days is set to 0 then the images will be deleted once the ID Pass is complete and this endpoint will never return image data for the ID Pass, even while it is still in progress. Once the retention period has passed and the images have been deleted, the images response parameter is no longer included. Images are returned in the images response parameter. **Example:** `true`
`return_documents`	**boolean or null** Optionally return detailed information about each of the documents which was assessed during the ID Pass process. This includes the full OCR information captured as well as the information which was validated. **Example:** `true`

Sample request

```
{
    "id": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "cipher_key": "IHm81bkcMsTL7J1ilxhDNE59+p5OQkvUJ3mZQhlUNHA=",
    "return_logs": true,
    "return_images": true,
    "return_documents": true
}
```

Responses

200 **Details of the ID Pass** (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`

Field	Description
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request for support queries. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`id`	**string (uuid)** The identifier for this ID Pass.
`link`	**string (url)** The ID Pass link to be provided to your user **Example:** `https://idpass.globaldata.net.au/idpass?token=abcd123456`
`status`	**string** The status of the ID Pass - `new` - The ID Pass has not yet been used - `opened` - The USer has proceeded past the welcome page - `in_progress` - The user has given consent to proceed - `complete` - The ID Pass has been completed - `expired` - The ID Pass has expired (reached link_validity_days) - `failed` - The ID Pass has failed - `cancelled` - The ID Pass has been cancelled **Enum:** `new`, `opened`, `in_progress`, `complete`, `expired`, `failed`, `cancelled` **Example:** `new`
`verification_status`	**string** The verification status of the ID Pass: - `passed` - The user passed the ID Pass process - `failed` - The ID failed to verify - `null` - The ID Pass is in progress **Enum:** `passed`, `failed` **Example:** `passed`
`verification_description`	**string** Human readable reason for the verification_status result **Example:** `Document 1 had biographic identity edits before verification The identity was verified successfully`

Field	Description
`config`	**object** The original configuration parameters used to create the ID Pass. See the ID Pass Register definition for these values.
`config.` `link_validity_days`	**integer** Number of days for which the link should remain available. **Example:** `3`
`config.` `image_retention_days`	**integer** Number of days for which the images from the check will be stored. **Example:** `5`
`config.` `check_liveness`	**boolean** Whether a face liveness check should be performed. **Example:** `true`
`config.` `document_1_allowed_types`	**array of strings** The types of documents which are allowed for the first document check.
`config.` `document_2_allowed_types`	**array of strings** The types of documents which are allowed for the second document check.
`config.` `document_3_allowed_types`	**array of strings** The types of documents which are allowed for the third document check.
`config.` `webhook_url`	**string** Webhook URL which the ID PAss system will call to notify events **Example:** `https://example.com/webhook`
`config.` `webhook_events`	**array of strings** The events for which the webhook should be called
`config.` `return_url`	**string** Optional URL to link to from the ID Pass complete page. **Example:** `https://example.com/return`

Field	Description
config. privacy_policy_url	**string** The privacy policy URL which will be displayed on the consent page. This is taken from your account configuration. Contact support to make any changes to this link. **Example:** `https://example.com/privacy`
config. requester_name	**string** Your company name as displayed on the consent page. This is taken from your account configuration. Contact support to make any changes to this field. **Example:** `Sample Company`
config. requested_identity	**one of 2 shapes** The customer identity details supplied at registration. When a cipher_key is provided this object is decrypted; otherwise the encrypted payload is returned for local decryption.

Field	Description
	DECRYPTED IDENTITY

Field	Description
`first_name`	**string** The first name of the person to be verified **Example:** `John`
`middle_name`	**string** The middle name of the person to be verified **Example:** `Andrew`
`last_name`	**string** The last name of the person to be verified **Example:** `Smith`
`date_of_birth`	**string (date)** The date of birth of the person to be verified **Example:** `1990-03-21`

ENCRYPTED IDENTITY

Field	Description
`iv`	**string** The initialisation vector for decrypting the requested identity
`mac`	**string** The message authentication code for the encrypted identity payload
`data`	**string** The encrypted requested identity data

`created_at`	**string** The date/time when the ID Pass was created **Example:** `2024-05-16T01:26:08+00:00`

Field	Description
`consent_given_at`	**string** The date/time when the user checked the consent box **Example:** `2024-05-16T01:35:11+00:00`
`consent_text`	**string** The text of the consent which was displayed to the user **Example:** `I confirm that I am authorised to provide my personal details...`
`completed_at`	**string** The date/time when the user completed the ID Pass process **Example:** `2024-05-16T01:38:28+00:00`
`ip_address`	**string** The IP address used to complete the ID Pass **Example:** `203.22.251.3`
`logs`	**array of strings** Array of log messages returned when the return_logs option is set
`validation_summary`	**one of 3 shapes** This includes the result for each document passed to validation and the identity which was verified. When a cipher_key is provided this object is decrypted; otherwise the encrypted payload is returned for local decryption. Returns an empty array if the validation summary has no data. The `date_of_birth` completion described below applies to the decrypted object only. Where no `cipher_key` was supplied the encrypted payload is returned exactly as stored, so decrypting it yourself gives a partial date in its stored `1980-03-00` / `1980-00-00` form and no `partial_date_of_birth` key.

Field	Description

DECRYPTED VALIDATION SUMMARY

Field	Description
`document_verification`	**string** The document verification strategy that was used for this ID Pass. Mirrors the `config.document_verification` value. **Enum:** `dvs` , `idsp` , `basic` **Example:** `dvs`
`liveness_result`	**object** The result of the liveness check. Only present when `check_liveness` is true.
`liveness_result.validated`	**boolean** Whether the liveness check was successful **Example:** `true`
`liveness_result.confidence`	**string** The confidence score for the liveness check, formatted as a string to three decimal places **Example:** `99.811`
`liveness_result.liveness_attempts`	**integer** The number of liveness attempts **Example:** `1`
`liveness_result.bounding_box`	**object** The bounding box of the liveness check
`liveness_result.bounding_box.top`	**number** The top coordinate of the bounding box (percentage of image height) **Example:** `0.1`
`liveness_result.bounding_box.left`	**number** The left coordinate of the bounding box (percentage of image width) **Example:** `0.1`
`liveness_result.bounding_box.width`	**number** The width of the bounding box (percentage of image width) **Example:** `0.1`
`liveness_result.bounding_box.height`	**number** The height of the bounding box (percentage of image height) **Example:** `0.1`

Field	Description
`documents`	**one of 2 shapes** Full details of the documents which were validated. This field is only returned when `return_documents: true` is set on the details request. Returns an empty array if the documents object has no data.

Field	Description

DECRYPTED DOCUMENTS

Field	Description
`document_1`	**object** Details from the first document
`document_1.` `document_type`	**string** The document type **Enum:** `licence` , `passport` , `medicare` , `visa` , `centrelink` , `birth_certificate` , `nz_licence` **Example:** `licence`
`document_1.` `document_number`	**integer** The document number. (1 for the first document) **Example:** `1`
`document_1.` `ocr_status`	**string or null** The OCR status of this document. For manual-entry document types (centrelink, birth_certificate), this is null. **Enum:** `running` , `failed` , `complete` **Example:** `complete`
`document_1.` `ocr_attempts`	**integer** The number of OCR attempts **Example:** `1`
`document_1.` `validation_status`	**string or null** The validation status of the document **Enum:** `running` , `failed` , `complete` **Example:** `complete`
`document_1.` `validation_result`	**one of 3 shapes** The validation result for this document. The shape of this object depends on the `document_verification` strategy configured for the ID Pass (see config.document_verification).

DVS VALIDATION RESULT

Field	Description
`strategy`	**string** The verification strategy used **Enum:** `dvs` **Example:** `dvs`

Field	Description
`images`	**object** The images captured during the check including the liveness probe image and any document images.
`images.` `probe_image`	**one of 2 shapes** The probe image captured during the liveness check (if liveness was requested) When a cipher_key is provided this image is decrypted; otherwise the encrypted payload is returned for local decryption.

DECRYPTED PROBE IMAGE

The probe image in base64 encoded JPEG format (string)

ENCRYPTED PROBE IMAGE

Field	Description
`iv`	**string** The initialisation vector for decrypting the probe image
`mac`	**string** The message authentication code for the encrypted probe image payload
`data`	**string** The encrypted probe image

`images.` `document_1`	**object** Images from document 1

`images.`
`document_1.`
`front`
	one of 2 shapes
	
	The front image for document 1 When a cipher_key is provided this image is decrypted; otherwise the encrypted payload is returned for local decryption.

Field	Description
	DECRYPTED FRONT IMAGE The front image in base64 encoded JPEG format (string) ENCRYPTED FRONT IMAGE **Field** / **Description** `iv` — **string** — The initialisation vector for decrypting the front image `mac` — **string** — The message authentication code for the encrypted front image payload `data` — **string** — The encrypted front image

`images.`
`document_1.`
`back`
	one of 2 shapes
	
	The back image for documents with a back side, eg licence When a cipher_key is provided this image is decrypted; otherwise the encrypted payload is returned for local decryption.

	DECRYPTED BACK IMAGE The back image in base64 encoded JPEG format (string) ENCRYPTED BACK IMAGE **Field** / **Description** `iv` — **string** — The initialisation vector for decrypting the back image `mac` — **string** — The message authentication code for the encrypted back image payload `data` — **string** — The encrypted back image

Field	Description
images. document_1. photo	**one of 2 shapes** The photo image for document types with a photo, eg licence, passport, visa When a cipher_key is provided this image is decrypted; otherwise the encrypted payload is returned for local decryption.

DECRYPTED PHOTO IMAGE

The photo image in base64 encoded JPEG format (string)

ENCRYPTED PHOTO IMAGE

Field	Description
iv	**string** The initialisation vector for decrypting the photo image
mac	**string** The message authentication code for the encrypted photo image payload
data	**string** The encrypted photo image

images. document_2	**object** Images from document 2
images. document_2. front	**one of 2 shapes** The front image for document 2 When a cipher_key is provided this image is decrypted; otherwise the encrypted payload is returned for local decryption.

Field	Description
	DECRYPTED FRONT IMAGE The front image in base64 encoded JPEG format (string) ENCRYPTED FRONT IMAGE **Field** / **Description** `iv` — **string** — The initialisation vector for decrypting the front image `mac` — **string** — The message authentication code for the encrypted front image payload `data` — **string** — The encrypted front image

`images.`
`document_2.`
`back`
	one of 2 shapes
	
	The back image for documents with a back side, eg licence When a cipher_key is provided this image is decrypted; otherwise the encrypted payload is returned for local decryption.

	DECRYPTED BACK IMAGE The back image in base64 encoded JPEG format (string) ENCRYPTED BACK IMAGE **Field** / **Description** `iv` — **string** — The initialisation vector for decrypting the back image `mac` — **string** — The message authentication code for the encrypted back image payload `data` — **string** — The encrypted back image

Field	Description
images. document_2. photo	**one of 2 shapes** The photo image for document types with a photo, eg licence, passport, visa When a cipher_key is provided this image is decrypted; otherwise the encrypted payload is returned for local decryption.

DECRYPTED PHOTO IMAGE

The photo image in base64 encoded JPEG format (string)

ENCRYPTED PHOTO IMAGE

Field	Description
iv	**string** The initialisation vector for decrypting the photo image
mac	**string** The message authentication code for the encrypted photo image payload
data	**string** The encrypted photo image

images. document_3	**object** Images from document 2
images. document_3. front	**one of 2 shapes** The front image for document 3 When a cipher_key is provided this image is decrypted; otherwise the encrypted payload is returned for local decryption.

Field	Description
	DECRYPTED FRONT IMAGE The front image in base64 encoded JPEG format (string) ENCRYPTED FRONT IMAGE **Field** / **Description** `iv` — **string** — The initialisation vector for decrypting the front image `mac` — **string** — The message authentication code for the encrypted front image payload `data` — **string** — The encrypted front image

`images.`
`document_3.`
`back`
	one of 2 shapes
	
	The back image for documents with a back side, eg licence When a cipher_key is provided this image is decrypted; otherwise the encrypted payload is returned for local decryption.

	DECRYPTED BACK IMAGE The back image in base64 encoded JPEG format (string) ENCRYPTED BACK IMAGE **Field** / **Description** `iv` — **string** — The initialisation vector for decrypting the back image `mac` — **string** — The message authentication code for the encrypted back image payload `data` — **string** — The encrypted back image

Field	Description
images. document_3. photo	**one of 2 shapes** The photo image for document types with a photo, eg licence, passport, visa When a cipher_key is provided this image is decrypted; otherwise the encrypted payload is returned for local decryption.

DECRYPTED PHOTO IMAGE

The photo image in base64 encoded JPEG format (string)

ENCRYPTED PHOTO IMAGE

Field	Description
iv	**string** The initialisation vector for decrypting the photo image
mac	**string** The message authentication code for the encrypted photo image payload
data	**string** The encrypted photo image

images. id_photo	**one of 2 shapes** The ID photo image provided by the user (if requested) When a cipher_key is provided this image is decrypted; otherwise the encrypted payload is returned for local decryption.

Field	Description
	DECRYPTED ID PHOTO IMAGE The ID photo image in base64 encoded JPEG format (string) ENCRYPTED ID PHOTO IMAGE

Field	Description
iv	string The initialisation vector for decrypting the ID photo image
mac	string The message authentication code for the encrypted ID photo image payload
data	string The encrypted ID photo image

images. id_photo_cropped	one of 2 shapes The cropped ID photo image (if requested) When a cipher_key is provided this image is decrypted; otherwise the encrypted payload is returned for local decryption.
	DECRYPTED CROPPED ID PHOTO IMAGE The cropped ID photo image in base64 encoded JPEG format (string) ENCRYPTED CROPPED ID PHOTO IMAGE

Field	Description
iv	string The initialisation vector for decrypting the cropped ID photo image
mac	string The message authentication code for the encrypted cropped ID photo image payload
data	string The encrypted cropped ID photo image

Sample response


```
{
    "message": "Ok",
```

```
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "id": "00000000-0000-0000-0000-000000000000",
    "link": "https://idpass.globaldata.net.au/idpass?token=abcd123456",
    "status": "new",
    "verification_status": "passed",
    "verification_description": "Document 1 had biographic identity edits before verification\nThe identity was
    "config": {
        "link_validity_days": 3,
        "image_retention_days": 5,
        "check_liveness": true,
        "document_1_allowed_types": [
            "licence",
            "passport"
        ],
        "document_2_allowed_types": [],
        "document_3_allowed_types": [],
        "webhook_url": "https://example.com/webhook",
        "webhook_events": [
            "complete"
        ],
        "return_url": "https://example.com/return",
        "privacy_policy_url": "https://example.com/privacy",
        "requester_name": "Sample Company",
        "requested_identity": {
            "first_name": "John",
            "middle_name": "Andrew",
            "last_name": "Smith",
            "date_of_birth": "1990-03-21"
        }
    },
    "created_at": "2024-05-16T01:26:08+00:00",
    "consent_given_at": "2024-05-16T01:35:11+00:00",
    "consent_text": "I confirm that I am authorised to provide my personal details...",
    "completed_at": "2024-05-16T01:38:28+00:00",
    "ip_address": "203.22.251.3",
    "logs": [
        "2024-05-16T01:31:23+00:00 - 203.22.251.3 - Starting IdPass flow",
        "2024-05-16T01:35:11+00:00 - 203.22.251.3 - Consent given"
    ],
    "validation_summary": {
        "document_verification": "dvs",
        "liveness_result": {
            "validated": true,
            "confidence": "99.811",
            "liveness_attempts": 1,
            "bounding_box": {
                "top": 0.1,
                "left": 0.1,
                "width": 0.1,
                "height": 0.1
            }
        },
        "id_photo_result": {
            "status": "complete",
            "bounding_box": {
                "top": 0.1,
                "left": 0.1,
                "width": 0.1,
                "height": 0.1
            },
            "image_passed": true,
```

```
            "biometric_passed": true
        },
        "document_1_result": {
            "document_type": "licence",
            "validated_fields": {
                "first_name": "John",
                "middle_name": "Andrew",
                "last_name": "Smith",
                "date_of_birth": "1990-03-21",
                "partial_date_of_birth": "1980-03",
                "full_name": "John Smith"
            },
            "biographic_validated": true,
            "biometric_validated": true,
            "biometric_similarity": "99.804",
            "biometric_threshold": 80,
            "changes": {
                "passport_number": {
                    "from": "PB6015447",
                    "to": "M1000000"
                }
            }
        },
        "verified_identity": {
            "first_name": "John",
            "middle_name": "Andrew",
            "last_name": "Smith",
            "date_of_birth": "1990-03-21",
            "partial_date_of_birth": "1980-03"
        },
        "requested_identity_mismatch": false
    },
    "documents": {
        "document_1": {
            "document_type": "licence",
            "document_number": 1,
            "ocr_status": "complete",
            "ocr_attempts": 1,
            "validation_status": "complete",
            "validation_result": {
                "strategy": "dvs",
                "verification_result_code": "Y",
                "verification_request_number": "7a3ed6e1-b707-4e2d-bacb-e2dfe8f57406",
                "additional_information": [
                    {
                        "code": "DL-006",
                        "message": "Card number not matched."
                    }
                ],
                "originating_agency_code": "DVS123",
                "checked_at": "2026-04-16 04:28:53"
            },
            "validation_attempts": 1,
            "biometric_result": {
                "passed": true,
                "threshold": 80,
                "similarity": 98.8764,
                "face_occluded": false
            },
            "ocr_data": {
                "first_name": "example value",
```

```
                "last_name": "example value",
                "date_of_birth": "1990-03-21"
            },
            "validation_data": {
                "first_name": "example value",
                "last_name": "example value",
                "date_of_birth": "1990-03-21",
                "partial_date_of_birth": "1980-03",
                "registration_number": "0100234",
                "registration_number_as_entered": "1/234/56X",
                "district_number_confirmed": "01"
            }
        },
        "id_photo": {
            "biometric_result": {
                "similarity": 98.8764,
                "passed": true,
                "threshold": 80
            }
        }
    },
    "images": {
        "probe_image": "string",
        "document_1": {
            "front": "string",
            "back": "string",
            "photo": "string"
        },
        "document_2": {
            "front": "string",
            "back": "string",
            "photo": "string"
        },
        "document_3": {
            "front": "string",
            "back": "string",
            "photo": "string"
        },
        "id_photo": "string",
        "id_photo_cropped": "string"
    }
}
```


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

Cancel an ID Pass

POST `/idpass/cancel`

Cancel an ID Pass. Only ID Passes with status `new`, `opened`, or `in_progress` can be cancelled. Terminal statuses (`complete`, `expired`, `failed`, `cancelled`) will return an error.

Please review the full ID Pass documentation here:

ID Pass documentation

Refund behaviour

- **Refund:** A cancel request will **refund the fee** if the ID Pass status is still `new` or `opened` (no verification steps have been taken).
- **No refund:** If any steps have been taken (for example status is `in_progress` or the user has started the flow), **no refund** is issued.

Request body

The ID of the ID Pass to cancel

Field	Description
`id` REQUIRED	string (uuid) The id of the ID Pass to cancel **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Sample request

```
{
    "id": "fe4291ca-d831-4760-96df-c9cb03b3cd95"
}
```

Responses

200 **ID Pass cancelled successfully** (application/json)

Field	Description
`message`	string A message indicating the result of the request **Example:** `Ok`
`api_reference`	string (uuid) A unique identifier for this request
`id`	string (uuid) The identifier for this ID Pass
`status`	string The status of the ID Pass after cancellation **Enum:** `cancelled` **Example:** `cancelled`

Field	Description
`fee_refunded`	**boolean** Whether a fee refund was applied (true if no verification steps had been taken and a fee had been charged) **Example:** `false`

Sample response

```
{
    "message": "Ok",
    "api_reference": "00000000-0000-0000-0000-000000000000",
    "id": "00000000-0000-0000-0000-000000000000",
    "status": "cancelled",
    "fee_refunded": false
}
```

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

Perform a Likeness (Face Similarity) Check

POST `/likeness_check`

Compares two face images (an ID/document or reference image and a probe/selfie image) and returns a similarity score together with a threshold evaluation flag (`biometric_passed`). The service is intended for validating whether the two supplied images likely represent the same individual.

Inputs

Provide both images as Base64 **data URI** strings (recommended form: `...`) or raw base64 without a prefix. Supported formats are JPEG and PNG. Maximum file size per image is 5 MB.

An optional `similarity_threshold` (0-100) can be supplied to override the system default. If omitted, the configured default threshold is used when determining `biometric_passed` .

Similarity & Threshold

The underlying biometric engine returns a `similarity` value (floating point). The request (or default) threshold is applied: `biometric_passed = similarity >= threshold` .

Typical Use Case

1. Capture / obtain an authoritative reference photo (e.g. ID document portrait).
2. Capture a live probe (selfie) image from the user.
3. Submit both images to this endpoint.
4. Use the `biometric_passed` boolean (and raw similarity) to decide downstream workflow steps.

Error Handling

Validation and processing issues return HTTP 400 with a descriptive `message` . If multiple validation errors are present, the API returns the **first** validation message. Face comparison engine errors (for example `no_face_found`) are also returned as HTTP 400 messages.

Request body

Two face images (reference & probe) and an optional threshold override.

Field	Description
`photo` REQUIRED	string Base64 (optionally data URI) reference image (e.g. ID/document portrait). JPEG or PNG. Max 5 MB. **Example:** `...`

Field	Description
`probe_image` REQUIRED	**string** Base64 (optionally data URI) probe / selfie image to compare against the reference. JPEG or PNG. Max 5 MB. **Example:** `...`
`similarity_threshold`	**number (float) [0..100]** Optional override threshold (0-100). If omitted, system default is used. **Example:** `80`

Sample request

```
{
    "photo": "...",
    "probe_image": "...",
    "similarity_threshold": 80
}
```

Responses

200 **Likeness comparison completed successfully** (application/json)

Field	Description
`message`	**string** Always `Ok` on success. **Example:** `Ok`
`api_reference`	**string (uuid)** Unique audit reference for this API call. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`result`	**object** Likeness comparison result object.
`result.` `similarity`	**number (float)** Biometric engine similarity score (higher means more similar). **Example:** `95.5`
`result.` `threshold`	**number (float)** Threshold used to evaluate pass/fail (request value or system default). **Example:** `80`

Field	Description
result. biometric_passed	**boolean** True if `similarity >= threshold`. **Example:** `true`

Sample response

```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "result": {
        "similarity": 95.5,
        "threshold": 80,
        "biometric_passed": true
    }
}
```

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

Create a Liveness Check

POST `/liveness_check`

A liveness check evaluates whether the captured face belongs to a live human (as opposed to a spoof such as a photo, screen, or mask).

This endpoint creates a short-lived link for performing a face liveness verification. The end user will be guided through a liveness capture flow in their browser. Once the capture is completed (or the link expires), the result can be retrieved using the `GET /api/v2/liveness_check?token=...` endpoint.

The resulting data will include a numeric confidence value (`confidence`), the applied `threshold` , and a boolean `passed` indicator (see the result endpoint for details).

Link validity

The link is valid for a configurable number of minutes (`link_valid_mins` , min 5 / max 30, default 10). After expiry, if no result has been produced, the check will eventually be marked expired.

Threshold

Optionally supply a custom `threshold` (0-99) that will be stored alongside the check and later echoed in the result. If omitted, a system default (currently 70) is applied.

Audit images

The optional `audit_images` parameter (0-4) controls how many additional intermediate frame images (beyond the single probe image) are retained temporarily (up to ~30 minutes) for audit / review. These images (including bounding box metadata when available) are only returned by the result endpoint when requested via `audit_images` > 0 during creation.

Webhooks

If a `webhook_url` is supplied, lifecycle events (e.g. completion / expiry) may be posted to that URL. Webhook invocations are not retried indefinitely; implement idempotency on your side.

Webhook Signature Verification

When a `webhook_url` is provided and an event (e.g. completion / expiry) occurs, a POST request is sent with a JSON body and a `Signature` HTTP header.

The `Signature` header is an HMAC SHA-256 hex digest of the payload generated using the `secret` returned at creation time.

IP Whitelisting for Webhooks

Webhooks will originate from one of the IP addresses listed in:

IP whitelisting for webhooks

Return URL Redirect & lc_token

After the user completes the liveness flow (or the flow reaches a terminal state), the browser will be redirected to your supplied `return_url` with an added query parameter:

`?lc_token={token}`

Example: `https://example.com/return?lc_token=u2Qe8C9vYl1kP4a7bD6fT3mN9xR5sZ0q`

IMPORTANT: Do not trust the `lc_token` value from the client redirect. Always treat it as untrusted input and fetch the definitive result from the API using `GET /api/v2/liveness_check?token=...` under your authenticated server context. This ensures the token belongs to your account and has not been tampered with.

On terminal failure states, an additional `error` parameter may be appended:

`?lc_token={token}&error=failed` or `?lc_token={token}&error=expired`

If your `return_url` already contains query parameters, these values will be appended using `&` instead of `?`.

Request body

Parameters to create the liveness check link

Field	Description
`return_url` REQUIRED	**string (url)** URL to redirect / link back to after the user completes the liveness flow. **Example:** `https://example.com/return`
`webhook_url`	**string (url)** Optional webhook URL to receive event callbacks (e.g. completion / expiry). **Example:** `https://example.com/webhook`
`link_valid_mins`	**integer [5..30]** Number of minutes the liveness link remains valid (default 10, min 5, max 30). **Default:** `10` **Example:** `10`
`threshold`	**integer [0..99]** Custom threshold (0-99) applied when interpreting the liveness score. Defaults to system configured value. **Example:** `70`

Field	Description
`audit_images`	integer [0..4] Number of additional audit frame images (0-4) to retain for short-term retrieval in the result response. **Example:** `2`

Sample request

```
{
    "return_url": "https://example.com/return",
    "webhook_url": "https://example.com/webhook",
    "link_valid_mins": 10,
    "threshold": 70,
    "audit_images": 2
}
```

Responses

200 **Liveness check created successfully** (application/json)

Field	Description
`message`	string A message indicating the result (`Ok` on success). **Example:** `Ok`
`api_reference`	string (uuid) Unique identifier for this request (for support / audit tracing). **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`token`	string Token used to reference the liveness check (supply to the result endpoint). **Example:** `u2Qe8C9vYl1kP4a7bD6fT3mN9xR5sZ0q`
`url`	string (url) The user-facing liveness capture URL (sandbox or live domain depending on environment). **Example:** `https://sandbox-idcheck.globaldata.net.au/liveness/u2Qe8C9vYl1kP4a7bD6fT3mN9xR5sZ0q`

Field	Description
`secret`	**string** Secret associated with this check (may be required for some subsequent secured operations). **Example:** `7f6a1e88-b6d2-4d8a-9f3f-5c4d2b1a0e9f`
`webhook_url`	**string (url)** Echoed only when a non-null webhook_url was supplied. **Example:** `https://example.com/webhook`

Sample response

```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "token": "u2Qe8C9vYl1kP4a7bD6fT3mN9xR5sZ0q",
    "url": "https://sandbox-idcheck.globaldata.net.au/liveness/u2Qe8C9vYl1kP4a7bD6fT3mN9xR5sZ0q",
    "secret": "7f6a1e88-b6d2-4d8a-9f3f-5c4d2b1a0e9f",
    "webhook_url": "https://example.com/webhook"
}
```

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

Retrieve a Liveness Check Result

GET `/liveness_check`

Retrieves the current status or final result of a previously created liveness check using the `token` returned by the create (POST) endpoint. This endpoint is idempotent and can be safely polled until a terminal state is reached.

Status lifecycle

A liveness check progresses through these internal states:

- `pending` - Created but not yet claimed by the client (no session started).
- `in_progress` - Claimed and session active / capturing.
- `complete` - Result available (successful or failed evaluation) and this endpoint returns HTTP 200.
- `expired` - Link validity period elapsed without completion; no result available.

While the check is `pending` or `in_progress`, this endpoint returns HTTP 202 with a descriptive message (`Liveness check is pending` or `Liveness check is in progress`). Once processing is finished it returns HTTP 200.

If a check has expired without completion, it will return HTTP 410 with the message `Liveness check has expired`.

Polling guidance

Recommended polling interval: 2-3 seconds. Exponential backoff is encouraged to reduce load. Stop polling once HTTP 200 is received or if you determine from your own business logic the check should be abandoned.

Result object

On success (HTTP 200) the response includes a `result` object mirroring what was stored for the check:

- `passed` (boolean) - Indicates whether the captured face passed the liveness evaluation.
- `confidence` (number) - Confidence score returned by the liveness provider.
- `threshold` (integer 0-99) - Either the custom value provided at creation or the system default.
- `checked_at` (datetime) - Time the result was finalized.
- `error` (string, only when failed/expired) - Terminal error code (`failed` or `expired`).

Optional fields when available:

- `image` - Base64 data URI (JPEG) of the probe frame (only if retained and not yet expired from short-term cache).
- `audit_images` - Array of objects each containing:
  - `image` - Base64 data URI (JPEG) of an intermediate frame.
  - `bounding_box` - Optional bounding box metadata (structure may evolve, treat as informational).

Images are ephemeral (cached ~30 minutes). After expiry the same result will be returned without images.

Security considerations

Always perform this query from a trusted backend using your API key. Do not trust the client-provided `lc_token` query parameter from the redirect alone; validate it by fetching the result here (403/400 will protect you if the token does not belong to your account).

Error / edge cases

- Missing or unknown `token` => HTTP 400 `Invalid check token`.
- Access without product entitlement => HTTP 403.
- Rate limiting => HTTP 429 (standard throttling rules apply).

Query parameters

Field	Description
`token` REQUIRED	string The token identifying the liveness check (returned from the create endpoint) **Example:** `u2Qe8C9vYl1kP4a7bD6fT3mN9xR5sZ0q`

Responses

200 **Liveness check completed - final result returned** (application/json)

Field	Description

`message`	**string** Always `Ok` on success. **Example:** `Ok`
`api_reference`	**string (uuid)** Unique identifier for this request (audit reference). **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`result`	**object** Liveness evaluation result
`result.` `passed`	**boolean** Indicates whether the subject was deemed live (true) or not (false). **Example:** `true`
`result.` `confidence`	**number (float)** Confidence score produced by the liveness provider. **Example:** `87`
`result.` `threshold`	**integer** Threshold applied when interpreting the score. **Example:** `75`
`result.` `checked_at`	**string (date-time)** Timestamp when the liveness result was finalized. **Example:** `2026-02-17T10:15:30Z`
`result.` `error`	**string** Terminal error code for failed / expired checks. **Enum:** `failed`, `expired` **Example:** `failed`
`result.` `bounding_box`	**object** Optional bounding box metadata for the detected face.
`result.` `image`	**string** Probe image (data URI) if still available. **Example:** `...`

result. audit_images	**array of objects** Array of intermediate audit frame images if requested and available.
result. audit_images[]. image	**string** Audit frame image (data URI) **Example:** `...`
result. audit_images[]. bounding_box	**object** Optional bounding box metadata for the detected face.

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "result": {
        "passed": true,
        "confidence": 87,
        "threshold": 75,
        "checked_at": "2026-02-17T10:15:30Z",
        "error": "failed",
        "bounding_box": {
            "left": 0.12,
            "top": 0.33,
            "width": 0.27,
            "height": 0.27
        },
        "image": "...",
        "audit_images": [
            {
                "image": "...",
                "bounding_box": {
                    "Top": 0.12,
                    "Left": 0.33,
                    "Width": 0.27,
                    "Height": 0.27
                }
            }
        ]
    }
}
```


202 **Liveness check not yet complete (pending or in progress)** (application/json)

Field	Description
message	**string** Status message - either `Liveness check is pending` or `Liveness check is in progress`. **Example:** `Liveness check is in progress`

Sample response

```
{
    "message": "Liveness check is in progress"
}
```

410 **Liveness check has expired** (application/json)

Field	Description
`message`	**string** Expiry message **Example:** `Liveness check has expired`

Sample response

```
{
    "message": "Liveness check has expired"
}
```

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

NZ Driver Licence Check

POST `/nz_driver_licence`

Verifies an individual's identity against the New Zealand NZTA Driver Licence database.

Input

- `last_name` is required.
- `first_name` is optional - omit for single-name (mononym) licence holders.
- `middle_name` is optional.
- `birth_date` must be in `YYYY-MM-DD` format.
- `licence_number` must be a valid NZ driver licence number (2 uppercase letters followed by 6 digits with a valid check digit).
- `licence_version` is the 3-digit version number from the front of the licence.
- `consent` must be `true`.

Output

Returns:

- `reporting_reference` - unique transaction identifier
- `match_status` - `"Match"` or `"NoMatch"`
- `document_verified` - boolean indicating whether the supplied licence details were verified by NZTA
- `additional_information` - only present when the data source returned an explanatory message alongside a `NoMatch` result

Sandbox environment data

When simulating queries in the sandbox environment, the following record will return a match:

First Name	Last Name	Date of Birth	Licence Number	Licence Version
Dana	Mitchell	1965-09-13	DB512036	001

Name matching is case insensitive, and `licence_number` is converted to upper case before it is validated, so `db512036` is also accepted. Any other otherwise-valid identity returns a `match_status` of `NoMatch` with `document_verified` set to `false`.

The following sentinel values simulate the remaining outcomes:

Field	Value	Sandbox result
`licence_number`	`DB111112`	`200` - a `NoMatch` result carrying `additional_information`.

Field	Value	Sandbox result
`licence_number`	`DB000000`	`503` - the licence source is unavailable.
`licence_version`	`999`	`400` - the request was rejected as invalid.

Request body

NZ Driver Licence verification request.

Field	Description
`first_name`	**string [max 60 characters] or null** First name as it appears on the licence. Omit for single-name (mononym) licence holders. **Example:** `Dana`
`middle_name`	**string [max 60 characters] or null** Middle name if present on the licence. **Example:**
`last_name` **REQUIRED**	**string [max 60 characters]** Last name as it appears on the licence. **Example:** `Mitchell`
`birth_date` **REQUIRED**	**string (date)** Date of birth in YYYY-MM-DD format. **Example:** `1965-09-13`
`licence_number` **REQUIRED**	**string** NZ driver licence number. **Example:** `DB512036`
`licence_version` **REQUIRED**	**string [max 60 characters]** 3-digit version number from the front of the licence. **Example:** `001`
`consent` **REQUIRED**	**boolean** Must be true. Confirms consent has been obtained. **Example:** `true`

Sample request

```
{
    "first_name": "Dana",
    "middle_name": "",
    "last_name": "Mitchell",
    "birth_date": "1965-09-13",
    "licence_number": "DB512036",
    "licence_version": "001",
    "consent": true
}
```

Responses

200 **NZ Driver Licence verification result.** (application/json)

Field	Description
`message`	**string** **Example:** `Ok`
`function`	**string** **Example:** `nz_driver_licence`
`api_reference`	**string (uuid)** **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`data`	**object**
`data.` `reporting_reference`	**string** Unique transaction identifier. **Example:** `8a2439b6-b8a4-450c-9ef0-ca382ceb482e`
`data.` `match_status`	**string** Overall match result. `Match` indicates the supplied identity was fully verified; `NoMatch` indicates verification failed. **Enum:** `Match`, `NoMatch` **Example:** `Match`
`data.` `document_verified`	**boolean** Whether NZTA reports the supplied licence details as verified. Defaults to `false` when the source did not return a verification verdict. **Example:** `true`

Field	Description
data. additional_information	string An explanatory message returned by the data source about the result. Only present when the source supplied one - typically alongside a `NoMatch` result. **Example:** `driversLicenceVersion does not match driversLicenceNo`

Sample response


```
{
    "message": "Ok",
    "function": "nz_driver_licence",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "data": {
        "reporting_reference": "8a2439b6-b8a4-450c-9ef0-ca382ceb482e",
        "match_status": "Match",
        "document_verified": true,
        "additional_information": "driversLicenceVersion does not match driversLicenceNo"
    }
}
```


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

Payroll Check

POST `/payroll_check`

Validates a person's details against payroll records.

Minimum search requirements

In order to perform a valid lookup, a minimum of the following fields are required:

- first_name
- last_name
- birth_date
- consent

The response will indicate if the provided fields are either `MATCHED` or `UNMATCHED` against the payroll records.

Sandbox environment

When simulating queries in the sandbox environment, the following records will return a match:

First Name	Middle Name	Last Name	Birth Date	Address	Email	Telephone	Employer ABN
John	James	Doe	1990-01-01	4/123 Fake St, Fakeville Vic 3987	john@example.com	0400123456	12345678901
Jane	Karen	Smith	1995-02-18	456 Wrong Ave, Sampleville NSW 2785	jane@example.com	0412345678	98765432109

> Note: If the first name is `service` and the last name is `unavailable`, the match will return a 503 error - Third Party Unavailable.

Request body

The details of the person to validate

Field	Description
`first_name`	**string** The first name of the individual **Example:** `John`

Field	Description
`middle_name`	**string** The middle name of the individual **Example:** `James`
`last_name`	**string** The last name of the individual **Example:** `Doe`
`birth_date`	**string (date)** The date of birth of the individual **Example:** `1990-01-01`
`address`	**object** The address of the individual
`address.` `street_address`	**string** The street line of the address **Example:** `4/123 Fake Street`
`address.` `suburb`	**string** The suburb of the address **Example:** `Fakeville`
`address.` `state`	**string** The state of the address **Enum:** `NSW`, `VIC`, `QLD`, `SA`, `WA`, `TAS`, `NT`, `ACT` **Example:** `VIC`
`address.` `postcode`	**string** The postcode of the address **Example:** `3987`
`email`	**string** The email address of the individual **Example:** `john@example.com`

Field	Description
`phone`	**string** The phone number of the individual in 0NSN format **Example:** `0400123456`
`employer_abn`	**string** The ABN of the employer **Example:** `12345678901`
`consent`	**boolean** The individual's consent to search payroll records **Example:** `true`

Sample request

```
{
    "first_name": "John",
    "middle_name": "James",
    "last_name": "Doe",
    "birth_date": "1990-01-01",
    "address": {
        "street_address": "4/123 Fake Street",
        "suburb": "Fakeville",
        "state": "VIC",
        "postcode": "3987"
    },
    "email": "john@example.com",
    "phone": "0400123456",
    "employer_abn": "12345678901",
    "consent": true
}
```

Responses

200 **Result of the validation check** (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`

Field	Description
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`match_results`	**object** The result of the payroll check
`match_results.` `person`	**string** The result of matching the first name, last name and birth date. **Enum:** `MATCHED` , `UNMATCHED` **Example:** `MATCHED`
`match_results.` `middle_name`	**string** The result of matching the middle name **Enum:** `MATCHED` , `UNMATCHED` **Example:** `MATCHED`
`match_results.` `address`	**string** The result of matching the address **Enum:** `MATCHED` , `UNMATCHED` **Example:** `MATCHED`
`match_results.` `email`	**string** The result of matching the email **Enum:** `MATCHED` , `UNMATCHED` **Example:** `MATCHED`
`match_results.` `phone`	**string** The result of matching the phone number **Enum:** `MATCHED` , `UNMATCHED` **Example:** `MATCHED`

Field	Description
match_results. employer_abn	string The result of matching the employer ABN **Enum:** `MATCHED` , `UNMATCHED` **Example:** `MATCHED`

Sample response

```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "match_results": {
        "person": "MATCHED",
        "middle_name": "MATCHED",
        "address": "MATCHED",
        "email": "MATCHED",
        "phone": "MATCHED",
        "employer_abn": "MATCHED"
    }
}
```

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

Super Check

POST `/super_check`

Validates a person's details against superannuation records.

Minimum search requirements

In order to perform a valid lookup, a minimum of the following fields are required:

- first_name
- last_name
- birth_date
- consent

The response will indicate if the provided fields are either `MATCHED` or `UNMATCHED` against the payroll records.

Sandbox environment

When simulating queries in the sandbox environment, the following records will return a match:

First Name	Middle Name	Last Name	Birth Date	Address	Email	Telephone	Employer ABN

John	James	Doe	1990-01-01	4/123 Fake St, Fakeville Vic 3987	john@example.com	0400123456	12345678901
Fiona		Cheng	1982-06-28	3 Right Ave, Hiddenville WA 6824	fiona@example.com	0400987654	91237856482

> Note: If the first name is `service` and the last name is `unavailable`, the match will return a 503 error - Third Party Unavailable.

Request body

The details of the person to validate

Field	Description
`first_name`	**string** The first name of the individual **Example:** `John`
`middle_name`	**string** The middle name of the individual **Example:** `James`
`last_name`	**string** The last name of the individual **Example:** `Doe`
`birth_date`	**string (date)** The date of birth of the individual **Example:** `1990-01-01`
`address`	**object** The address of the individual
`address.street_address`	**string** The street line of the address **Example:** `4/123 Fake Street`
`address.suburb`	**string** The suburb of the address **Example:** `Fakeville`

Field	Description
address. state	**string** The state of the address **Enum:** `NSW` , `VIC` , `QLD` , `SA` , `WA` , `TAS` , `NT` , `ACT` **Example:** `VIC`
address. postcode	**string** The postcode of the address **Example:** `3987`
email	**string** The email address of the individual **Example:** `john@example.com`
phone	**string** The phone number of the individual in 0NSN format **Example:** `0400123456`
employer_abn	**string** The ABN of the employer **Example:** `12345678901`
consent	**boolean** The individual's consent to search superannuation records **Example:** `true`

Sample request

```
{
    "first_name": "John",
    "middle_name": "James",
    "last_name": "Doe",
    "birth_date": "1990-01-01",
    "address": {
        "street_address": "4/123 Fake Street",
        "suburb": "Fakeville",
        "state": "VIC",
        "postcode": "3987"
    },
    "email": "john@example.com",
    "phone": "0400123456",
    "employer_abn": "12345678901",
    "consent": true
```

```
}
```

Responses

200 **Result of the validation check** (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. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`match_results`	**object** The result of the payroll check
`match_results.person`	**string** The result of matching the first name, last name and birth date. **Enum:** `MATCHED`, `UNMATCHED` **Example:** `MATCHED`
`match_results.middle_name`	**string** The result of matching the middle name **Enum:** `MATCHED`, `UNMATCHED` **Example:** `MATCHED`
`match_results.address`	**string** The result of matching the address **Enum:** `MATCHED`, `UNMATCHED` **Example:** `MATCHED`
`match_results.email`	**string** The result of matching the email **Enum:** `MATCHED`, `UNMATCHED` **Example:** `MATCHED`

Field	Description
match_results. phone	string The result of matching the phone number **Enum:** `MATCHED` , `UNMATCHED` **Example:** `MATCHED`
match_results. employer_abn	string The result of matching the employer ABN **Enum:** `MATCHED` , `UNMATCHED` **Example:** `MATCHED`

Sample response

```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "match_results": {
        "person": "MATCHED",
        "middle_name": "MATCHED",
        "address": "MATCHED",
        "email": "MATCHED",
        "phone": "MATCHED",
        "employer_abn": "MATCHED"
    }
}
```

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

Payroll and Super Check

POST `/payroll_super_check`

Validates a person's details against payroll and/or superannuation records.

Minimum search requirements

In order to perform a valid lookup, a minimum of the following fields are required:

- first_name
- last_name
- birth_date
- either super_consent or payroll_consent

The response will indicate if the provided fields are either `MATCHED` or `UNMATCHED` against the records.

Sandbox environment

When simulating queries in the sandbox environment, the following records will return a match:

First Name	Middle Name	Last Name	Birth Date	Address	Email	Telephone	Employer ABN	Match Source
John	James	Doe	1990-01-01	4/123 Fake St, Fakeville Vic 3987	john@example.com	0400123456	12345678901	Super and Payroll
Jane	Karen	Smith	1995-02-18	456 Wrong Ave, Sampleville NSW 2785	jane@example.com	0412345678	98765432109	Payroll
Fiona		Cheng	1982-06-28	3 Right Ave, Hiddenville WA 6824	fiona@example.com	0400987654	91237856482	Super

> Note: If the first name is `service` and the last name is `unavailable`, the match will return a 503 error - Third Party Unavailable.

Request body

The details of the person to validate

Field	Description
`first_name`	**string** The first name of the individual **Example:** `John`
`middle_name`	**string** The middle name of the individual **Example:** `James`
`last_name`	**string** The last name of the individual **Example:** `Doe`
`birth_date`	**string (date)** The date of birth of the individual **Example:** `1990-01-01`
`address`	**object** The address of the individual

Field	Description
address. street_address	**string** The street line of the address **Example:** `4/123 Fake Street`
address. suburb	**string** The suburb of the address **Example:** `Fakeville`
address. state	**string** The state of the address **Enum:** `NSW`, `VIC`, `QLD`, `SA`, `WA`, `TAS`, `NT`, `ACT` **Example:** `VIC`
address. postcode	**string** The postcode of the address **Example:** `3987`
email	**string** The email address of the individual **Example:** `john@example.com`
phone	**string** The phone number of the individual in 0NSN format **Example:** `0400123456`
employer_abn	**string** The ABN of the employer **Example:** `12345678901`
super_consent	**boolean** The individual's consent to search superannuation records **Example:** `true`
payroll_consent	**boolean** The individual's consent to search payroll records **Example:** `true`

Sample request

```
{
    "first_name": "John",
    "middle_name": "James",
    "last_name": "Doe",
    "birth_date": "1990-01-01",
    "address": {
        "street_address": "4/123 Fake Street",
        "suburb": "Fakeville",
        "state": "VIC",
        "postcode": "3987"
    },
    "email": "john@example.com",
    "phone": "0400123456",
    "employer_abn": "12345678901",
    "super_consent": true,
    "payroll_consent": true
}
```

Responses

200 **Result of the validation check** (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. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`super`	**object** The result of the superannuation check
`super.person`	**string** The result of matching the first name, last name and birth date. **Enum:** `MATCHED`, `UNMATCHED` **Example:** `MATCHED`
`super.middle_name`	**string** The result of matching the middle name **Enum:** `MATCHED`, `UNMATCHED` **Example:** `MATCHED`

Field	Description
super. address	**string** The result of matching the address **Enum:** `MATCHED` , `UNMATCHED` **Example:** `MATCHED`
super. email	**string** The result of matching the email **Enum:** `MATCHED` , `UNMATCHED` **Example:** `MATCHED`
super. phone	**string** The result of matching the phone number **Enum:** `MATCHED` , `UNMATCHED` **Example:** `MATCHED`
super. employer_abn	**string** The result of matching the employer ABN **Enum:** `MATCHED` , `UNMATCHED` **Example:** `MATCHED`
payroll	**object** The result of the payroll check
payroll. person	**string** The result of matching the first name, last name and dob. **Example:** `MATCHED`
payroll. middle_name	**string** The result of matching the middle name **Example:** `MATCHED`
payroll. address	**string** The result of matching the address **Example:** `MATCHED`

Field	Description
payroll. email	**string** The result of matching the email **Example:** `MATCHED`
payroll. telephone	**string** The result of matching the telephone number **Example:** `MATCHED`
payroll. employer_abn	**string** The result of matching the employer ABN **Example:** `MATCHED`

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "super": {
        "person": "MATCHED",
        "middle_name": "MATCHED",
        "address": "MATCHED",
        "email": "MATCHED",
        "phone": "MATCHED",
        "employer_abn": "MATCHED"
    },
    "payroll": {
        "person": "MATCHED",
        "middle_name": "MATCHED",
        "address": "MATCHED",
        "email": "MATCHED",
        "telephone": "MATCHED",
        "employer_abn": "MATCHED"
    }
}
```


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

Person History

POST `/person_history`

The Person History API allows you to query and retrieve additional historical data for a specific individual. By providing certain identifying information, you can obtain a comprehensive set of records that match the given criteria.

ADC Check

The returned records are checked against the ADC (Australian Death Check) service and the result is returned in the 'deceased' field of the match response. A 'Y' indicates that the individual is recorded as deceased, while an 'N' indicates that the individual is not recorded as deceased. The ADC lookup is based on the matched individual's name, date of birth and state.

The ADC check is performed when the following fields are available:

- first_name
- last_name
- birth_date
- state

Minimum search requirements

In order to perform a valid lookup, a minimum of the following fields are required:

- first_name and last_name, and at least one of:
- birth_date
- phone
- email

If less than the minimum requirements are supplied, the API will return a 400 error with the message "One of email or phone or dob is required."

Sandbox environment data

When simulating queries in the sandbox environment, the following records will return a match:

First Name	Last Name	Birth Date	Email	Telephone	Address
John	Smith	1998-03-21	jsmith1998@example.com	0399999999, 0491222111	20 HARDY ST, LILYDALE 3140 VIC
John	Smith			0491222111	8 WALTHAM ST, RICHMOND 3121 VIC

First Name	Last Name	Birth Date	Email	Telephone	Address
Mary	Jones	1989-08-12	mary_jones89@example.com	0399995000	35 YARALLA ST, CONCORD WEST 2138 NSW

Request body

The details of the record to search

Field	Description
`first_name`	**string** The first name of the individual **Example:** `John`
`last_name`	**string** The last name of the individual **Example:** `Smith`
`birth_date`	**string (date)** The date of birth of the individual **Example:** `1998-03-21`
`phone`	**string** The phone number in 0NSN format **Example:** `0491222111`
`email`	**string** The email address **Example:** `jsmith1998@example.com`

Sample request

```
{
    "first_name": "John",
    "last_name": "Smith",
    "birth_date": "1998-03-21",
    "phone": "0491222111",
    "email": "jsmith1998@example.com"
}
```

Responses

200 **Result of the check** (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. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`matches`	**array of objects** Array of matched individuals.
`matches[].first_name`	**string** The first name of the matched individual. **Example:** `John`
`matches[].middle_name`	**string** The middle name of the matched individual. **Example:** `Andrew`
`matches[].last_name`	**string** The last name of the matched individual. **Example:** `Smith`
`matches[].birth_date`	**string or null** The dob of the matched individual. **Example:** `1998-03-21`
`matches[].deceased`	**string** The result of the ADC check for the matched individual. - Y: The person is recorded as deceased - N: The person is not recorded as deceased **Enum:** `Y`, `N` **Example:** `N`

Field	Description
matches[]. address	**string** The address of the matched individual. **Example:** `20 HARDY ST`
matches[]. suburb	**string** The suburb of the matched individual. **Example:** `LILYDALE`
matches[]. state	**string** The state of the matched individual. **Enum:** `ACT`, `NSW`, `NT`, `QLD`, `SA`, `TAS`, `VIC`, `WA` **Example:** `VIC`
matches[]. postcode	**string** The postcode of the matched individual. **Example:** `3140`
matches[]. date_start	**string (date)** The date the address record was first seen. **Example:** `2010-01-01`
matches[]. date_end	**string (date)** The date the address record was last seen. **Example:** `2022-03-15`
matches[]. phones	**array of objects** Array of phone numbers associated with the matched individual.
matches[]. phones[]. phone	**string** The phone number in 0NSN format. **Example:** `0491222111`
matches[]. phones[]. date_start	**string (date)** The date the phone record was first seen. **Example:** `2010-01-01`

Field	Description
matches[]. phones[]. date_end	**string (date)** The date the phone record was last seen. **Example:** `2022-03-15`
matches[]. emails	**array of objects** Array of email addresses associated with the matched individual.
matches[]. emails[]. email	**string** The email address. **Example:** `jsmith1998@example.com`
matches[]. emails[]. date_start	**string (date)** The date the email record was first seen. **Example:** `2010-01-01`
matches[]. emails[]. date_end	**string (date)** The date the email record was last seen. **Example:** `2022-03-15`

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "matches": [
        {
            "first_name": "John",
            "middle_name": "Andrew",
            "last_name": "Smith",
            "birth_date": "1998-03-21",
            "deceased": "N",
            "address": "20 HARDY ST",
            "suburb": "LILYDALE",
            "state": "VIC",
            "postcode": "3140",
            "date_start": "2010-01-01",
            "date_end": "2022-03-15",
            "phones": [
                {
                    "phone": "0491222111",
                    "date_start": "2010-01-01",
                    "date_end": "2022-03-15"
                }
            ],
            "emails": [
                {
                    "email": "jsmith1998@example.com",
```

```
                    "date_start": "2010-01-01",
                    "date_end": "2022-03-15"
                }
            ]
        }
    ]
}
```

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

Marketing Contact Lookup

POST `/marketing_contact_lookup`

A focused contact lookup intended for marketing and outreach customers (for example real estate agents and marketing service providers). The caller supplies exactly one piece of information about a person and receives back up to `max_records` matching person records from our universe (default 1, max 25), each including name, contact details, the associated address and per-contact marketing opt-in status. Records are returned most-recent first.

Inputs

Exactly one of the following lookup keys must be supplied:

Key	Description
`phone`	Australian landline or mobile, in either `0NSN` (e.g. `0299995000`) or E.164 (`+61299995000`) format
`email`	Email address
`gnaf_id`	A GNAF address identifier (e.g. `GANSW716615055`)
`full_address`	A single free-text Australian address; this will be parsed and resolved to a GNAF
Address parts	`street_address` plus `postcode` (with optional `suburb` and `state` to improve match accuracy)

Plus the following optional parameters:

Key	Description
`max_records`	Integer, `1` - `25`, default `1`. The maximum number of person records to return in `records[]`. Persons are deduplicated by underlying identity before slicing, so a person who lived at the address across multiple tenancies counts as one record.
`require_marketing_optin`	Boolean, default `false`. When `true`, each returned record's phones and emails are filtered down to those explicitly opted in for marketing; persons left with no opted-in contacts are skipped entirely and we keep scanning further down the candidate list to fill the `max_records` quota.
`min_date_end`	ISO 8601 date (`YYYY-MM-DD`), optional. When supplied, candidate persons whose most-recent association ended before this date are excluded before the `max_records` slice. Inclusive at the boundary (a `date_end` equal to `min_date_end` passes). Persons with a current / open-ended association (`date_end: null` in the response) are always kept.

What we return

Up to `max_records` matching person records, ordered most-recent first, each containing name, date of birth (and DOB range, where the exact DOB is unknown), gender, all known phone numbers and email addresses (each annotated with a per-contact `marketing_opt_in` flag), and the associated address record.

Persons flagged as deceased are excluded. Records are filtered against our suppressions system. If no match is found (or all matches are excluded for the reasons above) the response returns an empty `records: []` array.

What this query does NOT do

- It does not perform a social check (use the Social Check API).
- It does not perform a DNC check (use the DNC API).
- It does not check phone or email connectivity / deliverability (use the Phone Ping and Email Ping APIs).
- It does not return court data.
- It does not return business associations.

Sandbox environment data

When simulating queries in the sandbox environment, the following records will return a match. The `Marketing opt-in` column shows which contacts are flagged as opted-in for marketing - use these to exercise both `marketing_opt_in: true` and `marketing_opt_in: false` in the response, and to test the `require_marketing_optin: true` filter behaviour.

First Name	Last Name	DOB	Address	Contact	Marketing opt-in
John	Smith	1998-03-21	20 HARDY ST, LILYDALE 3140 VIC	phone `0399999999`	no
John	Smith	1998-03-21		phone `0491222111`	no
John	Smith	1998-03-21		email `jsmith1998@example.com`	no
Robert	Brown	1971-03-18	8 WALTHAM ST, RICHMOND 3121 VIC	phone `0399998888`	yes
Mary	Jones	1989-08-12	35 YARALLA ST, CONCORD WEST 2138 NSW	phone `0299995000`	yes
Mary	Jones	1989-08-12		phone `0491999888`	no
Mary	Jones	1989-08-12		email `mary_jones89@example.com`	yes
Peter	Jones	1985-06-15	35 YARALLA ST, CONCORD WEST 2138 NSW	phone `0399994100`	yes
Peter	Jones	1985-06-15		phone `0491333222`	no

First Name	Last Name	DOB	Address	Contact	Marketing opt-in
Peter	Jones	1985-06-15		email `peter_jones85@example.com`	yes
Emily	Tanner	1992-04-30	35 YARALLA ST, CONCORD WEST 2138 NSW (past)	phone `0399994201`	no
Emily	Tanner	1992-04-30		email `emily_tanner@example.com`	no
R	Brown	(unknown)	22 BRABYN ST, WINDSOR 2756 NSW	phone `0880885999`	no
R	Brown	(unknown)		email `rp_brown71@example.com`	no

Note that `35 YARALLA ST, CONCORD WEST 2138 NSW` (`GANSW716615055`) is shared by three persons: Mary Jones and Peter Jones are current residents and Emily Tanner is a past resident. This makes it a useful sandbox fixture for exercising `max_records` and the `require_marketing_optin` scan-to-fill behaviour.

Worked examples:

- `{ "phone": "0299995000" }` returns Mary Jones with two phones (one opt-in, one not) and one opt-in email.
- `{ "phone": "0299995000", "require_marketing_optin": true }` returns Mary Jones with only her opt-in phone and her opt-in email.
- `{ "phone": "0399999999" }` returns John Smith with all contacts annotated `marketing_opt_in: false` .
- `{ "phone": "0399999999", "require_marketing_optin": true }` returns `records: []` because John Smith has no opted-in contacts.
- `{ "gnaf_id": "GANSW716615055" }` returns Mary Jones (the most-recent current resident), one record.
- `{ "gnaf_id": "GANSW716615055", "max_records": 5 }` returns three records ordered most-recent first: Mary Jones, Peter Jones, Emily Tanner.
- `{ "gnaf_id": "GANSW716615055", "max_records": 5, "require_marketing_optin": true }` returns Mary Jones and Peter Jones (each filtered down to their opt-in contacts); Emily Tanner is skipped because none of her contacts are opted in.
- `{ "gnaf_id": "GANSW716615055", "max_records": 5, "min_date_end": "2020-01-01" }` returns Mary Jones and Peter Jones; Emily Tanner is excluded because her tenancy ended in `2019-07-19` , before the supplied minimum.

Opt-in flag notes

Note that the optin flag is associated with the person and their email address or phone number, not the address. The address is returned for reference only and should not be assumed to be opted in for marketing.

Request body

The lookup key and optional opt-in filter.

Field	Description

`phone`	**string** Australian phone number in `0NSN` (e.g. `0299995000`) or E.164 (e.g. `+61299995000`) format. **Example:** `0299995000`
`email`	**string (email)** Email address. Not case sensitive. **Example:** `mary_jones89@example.com`
`gnaf_id`	**string** GNAF address identifier. **Example:** `GANSW716615055`
`full_address`	**string** A single free-text Australian address. This will be parsed and resolved to a GNAF. **Example:** `35 Yaralla St, Concord West NSW 2138`
`street_address`	**string** Street address. Must be supplied together with `postcode`. `suburb` and `state` are optional but improve match accuracy. **Example:** `35 Yaralla St`
`suburb`	**string** Suburb. Optional - improves match accuracy when address parts are supplied. **Example:** `Concord West`
`state`	**string** State. Optional - improves match accuracy when address parts are supplied. **Enum:** `ACT`, `NSW`, `NT`, `QLD`, `SA`, `TAS`, `VIC`, `WA` **Example:** `NSW`
`postcode`	**string** Australian postcode. Required when `street_address` is supplied. **Example:** `2138`

`require_marketing_optin`	**boolean** When `true`, each returned record's phones and emails are filtered down to those explicitly opted in for marketing; persons left with no opted-in contacts are skipped entirely and we keep scanning further down the candidate list to fill the `max_records` quota. Defaults to `false`. **Example:** `false`
`max_records`	**integer [1..25]** The maximum number of person records to return in `records[]`. Persons are deduplicated by underlying identity before slicing, so a person who lived at the address across multiple tenancies counts as one record. Defaults to `1`. Range: `1` to `25`. **Default:** `1` **Example:** `5`
`min_date_end`	**string (date)** Optional ISO 8601 date (`YYYY-MM-DD`). Excludes candidate persons whose most-recent association ended before this date, before the `max_records` slice. Inclusive at the boundary (a `date_end` equal to `min_date_end` passes). Persons with a current / open-ended association (`date_end: null` in the response) are always kept. **Example:** `2020-01-01`

Sample request

```
{
    "phone": "0299995000",
    "email": "mary_jones89@example.com",
    "gnaf_id": "GANSW716615055",
    "full_address": "35 Yaralla St, Concord West NSW 2138",
    "street_address": "35 Yaralla St",
    "suburb": "Concord West",
    "state": "NSW",
    "postcode": "2138",
    "require_marketing_optin": false,
    "max_records": 5,
    "min_date_end": "2020-01-01"
}
```

Responses

200 **Result of the lookup.** (application/json)

Field	Description

`message`	**string** A message indicating the result of the request. This will be `Ok` for a successful response (including no-match results). **Example:** `Ok`
`function`	**string** The name of the API function that was called. **Example:** `marketing_contact_lookup`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`records`	**array of objects** Up to `max_records` matching person records, ordered most-recent first. Empty array when no match is found (or when `require_marketing_optin = true` and no matching person has any opted-in contacts).
`records[].` `title`	**string** The title (e.g. MR, MS) of the matched person. **Example:** `MS`
`records[].` `first_name`	**string** First name. **Example:** `MARY`
`records[].` `middle_name`	**string** Middle name. **Example:** `SALLY`
`records[].` `last_name`	**string** Last name. **Example:** `JONES`
`records[].` `dob`	**string or null** Date of birth (where known). **Example:** `1989-08-12`

records[]. dob_range_from	**string or null** Earliest possible date of birth (used when the exact DOB is unknown).
records[]. dob_range_to	**string or null** Latest possible date of birth (used when the exact DOB is unknown).
records[]. gender	**string** Gender. **Example:** `FEMALE`
records[]. phones	**array of objects** Phone numbers associated with this person, each annotated with a marketing opt-in flag.
records[]. phones[]. phone	**string** Phone number in 0NSN format. **Example:** `0299995000`
records[]. phones[]. marketing_opt_in	**boolean** Whether this phone number is explicitly opted in for marketing. **Example:** `true`
records[]. emails	**array of objects** Email addresses associated with this person, each annotated with a marketing opt-in flag.
records[]. emails[]. email	**string** Email address. **Example:** `mary_jones89@example.com`
records[]. emails[]. marketing_opt_in	**boolean** Whether this email address is explicitly opted in for marketing. **Example:** `true`
records[]. address_id	**string** GNAF address identifier. **Example:** `GANSW716615055`

records[]. street_address	**string** Street address. **Example:** `35 YARALLA ST`
records[]. suburb	**string** Suburb. **Example:** `CONCORD WEST`
records[]. state	**string** State. **Enum:** `ACT`, `NSW`, `NT`, `QLD`, `SA`, `TAS`, `VIC`, `WA` **Example:** `NSW`
records[]. postcode	**string** Postcode. **Example:** `2138`
records[]. date_start	**string or null** First date the person was associated with this address. **Example:** `2016-06-09`
records[]. date_end	**string or null** Last date the person was associated with this address. `null` for current / open-ended associations.
records[]. latitude	**string or null** Latitude of the address. **Example:** `-33.84970097`
records[]. longitude	**string or null** Longitude of the address. **Example:** `151.09426095`

Sample response


```
{
    "message": "Ok",
    "function": "marketing_contact_lookup",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "records": [
```

```
        {
            "title": "MS",
            "first_name": "MARY",
            "middle_name": "SALLY",
            "last_name": "JONES",
            "dob": "1989-08-12",
            "dob_range_from": null,
            "dob_range_to": null,
            "gender": "FEMALE",
            "phones": [
                {
                    "phone": "0299995000",
                    "marketing_opt_in": true
                }
            ],
            "emails": [
                {
                    "email": "mary_jones89@example.com",
                    "marketing_opt_in": true
                }
            ],
            "address_id": "GANSW716615055",
            "street_address": "35 YARALLA ST",
            "suburb": "CONCORD WEST",
            "state": "NSW",
            "postcode": "2138",
            "date_start": "2016-06-09",
            "date_end": null,
            "latitude": "-33.84970097",
            "longitude": "151.09426095"
        }
    ]
}
```


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

Deceased Check

POST `/deceased_check`

The Deceased Check API allows you to check if an individual is recorded as deceased against the Global Data death check service.

Deceased Check

The check is performed based on the name and date of birth / date of death of the individual. If the individual is found in the deceased records, then the deceased indicator will be set to 'Y'.

Search modes

Note that the search treats the first name field differently depending on whether a date of birth or date of death is provided.

Date of birth based search

When providing a name and date of birth only, the check will be performed based on the name and date of birth. In this mode the name search for first name allows the the omission of the middle name(s). So a search for "John Smith" will match "John

Andrew Smith" and "John Smith" but not "Andrew Smith".

When providing a first and middle name (provided together in the first_name field), the search will be performed based on the first and middle name. So a search for "John Andrew Smith" will match "John Andrew Smith" but not "John Smith" or "John Peter Smith".

Date of death based search

When providing a name and date of death only, the check will be performed based on the name and date of death. In this mode the name search for first name must be an exact match. So a search for "John Smith" will match "John Smith" but not "John Andrew Smith".

Date of birth and date of death based search

When providing a name, date of birth and date of death, the check will be performed based on the name and date of birth. Once results are returned, the date of death will be checked against the returned results. This search mode operates identically to the date of birth based search and allows for the omission of the middle name(s).

Sandbox environment data

When simulating queries in the sandbox environment, the following records will return a match:

First Name	Last Name	Date of Birth	Date of Death
John	Smith	1998-03-21	2023-07-24
John	Doe	1971-01-23	2003-08-22
John Andrew	Doe	1971-01-23	2011-11-03

These samples can be queried with or without the date of death.

Request body

The details of the record to search

Field	Description
`first_name`	**string** The first and optional middle name of the individual **Example:** `John Andrew`
`last_name`	**string** The last name of the individual **Example:** `Doe`
`birth_date`	**string (date)** The date of birth of the individual **Example:** `1998-03-21`

Field	Description
`death_date`	**string (date) or null** The date of death of the individual (optional) **Example:** `2011-11-03`

Sample request

```
{
    "first_name": "John Andrew",
    "last_name": "Doe",
    "birth_date": "1998-03-21",
    "death_date": "2011-11-03"
}
```

Responses

200 **Result of the check** (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. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`deceased_indicator`	**string** The deceased indicator. This will be 'Y' if the individual is found in the deceased records, otherwise it will be 'N'. **Enum:** `Y`, `N` **Example:** `N`

Sample response

```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "deceased_indicator": "N"
}
```

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

Enhance Record Plus

POST `/enhance_record_plus`

The Enhance Record call will perform a record enhancement on the supplied person and associated contact details. This version of the endpoint will also conduct a court check and social check on the results.

Minimum search requirements

In order to perform a valid lookup, a minimum of the following fields are required:

- first_name, last_name and at least one of dob or address
- phone
- email

If less than the minimum requirements are supplied, the API will return a 400 error with the message "Insufficient information to perform the request"

Opt-in

By default, the results will include all available records. If you would like to limit the results to only those that have opted in to marketing or public records (eg whitepages), you can set the `opt-in` parameter to `true`. Note that any social records returned are NOT checked for opt-in status and must not be used for marketing purposes.

Social Records Warning

The social records are included based on the emails or phone numbers appearing in social media records, and must not be taken to directly represent the person from this search record or be used for marketing purposes. Organisations must make their own enquiries as to the accuracy and appropriate use of this information.

Court Records Warning

The records are included based on the name and state appearing in public court records, and must not be taken to directly represent the person from this search record. Organisations must make their own enquiries as to the accuracy and appropriate use of this information.

Sandbox environment data

When simulating queries in the sandbox environment, the following records will return a match:

First Name	Last Name	Street Address	Suburb	Postcode	State	Phone	Email	Result Type
Michael	Raymond	12 Conestoga Way	Upper Coomera	4209	QLD		michael.raymond@example.com	Match, Social, Court
Emma	Russell	4 Westmill Dr	Hoppers Crossing	3029	VIC	0427519643		Match

First Name	Last Name	Street Address	Suburb	Postcode	State	Phone	Email	Result Type
Owen	Wyllie	4 Butcherbird Cl	Eli Waters	4655	QLD	0741241873	bc68@pardswit.org.au	Match
Deanna	Price	45/3 Spurway Dr	Baulkham Hills	2153	NSW	0298389934		Match
Peter	Jenkin	2 Tanami Cl	Belrose	2085	NSW	0294511557, 0294511557	peter508@moraitive.net.au	Match
Christian	Wessels	PO Box 280	Camden	2570	NSW	0246571285	vzchristian52@bittomizess.net	Match
Julie	Spivey	34/12 Melville Rd	Salisbury East	5109	SA	0410093241	spivey93@namithusly.net	Match

Request body

The details of the record to enhance

Field	Description
`first_name`	**string** The first name of the individual **Example:** `Michael`
`middle_name`	**string** The middle name of the individual **Example:** `John`
`last_name`	**string** The last name of the individual **Example:** `Raymond`
`dob`	**string (date)** The date of birth of the individual **Example:** `1969-06-21`
`phone`	**string** The phone number in 0NSN format **Example:** `0412345678`

Field	Description
`email`	**string** The email address **Example:** `michael.raymond@example.com`
`street_address`	**string** The street/postal address component (Address line 1) **Example:** `12 Conestoga Way`
`suburb`	**string** The suburb component of the address **Example:** `Upper Coomera`
`state`	**string** The Australian state of the individual's address **Enum:** `ACT`, `NSW`, `NT`, `QLD`, `SA`, `TAS`, `VIC`, `WA` **Example:** `QLD`
`postcode`	**string** The postcode component of the address **Example:** `4209`
`opt-in`	**boolean** Limit the results to only those that have opted in to marketing or public records (eg whitepages). Note that any social records returned are NOT checked for opt-in status and must not be used for marketing purposes. - `true` - Limit the results to only those that have opted in to marketing or public records (eg whitepages). - `false` - Include all records. **Example:** `false`

Sample request

```
{
    "first_name": "Michael",
    "middle_name": "John",
    "last_name": "Raymond",
    "dob": "1969-06-21",
    "phone": "0412345678",
    "email": "michael.raymond@example.com",
    "street_address": "12 Conestoga Way",
```

```
    "suburb": "Upper Coomera",
    "state": "QLD",
    "postcode": "4209",
    "opt-in": false
}
```

Responses

200 **Result of the check** (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. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`supplied_address`	**array of objects** The supplied address, parsed and cleaned.
`supplied_address[].` `match_result`	**string** The address match type: • address_not_found: No matching address found • address_empty: No address supplied • address_corrected: Address was corrected from supplied address • address_match: Address was matched as supplied **Enum:** `address_not_found`, `address_empty`, `address_corrected`, `address_match` **Example:** `address_match`
`supplied_address[].` `address_id`	**string** The unique identifier for the address **Example:** `ABCDE123456789`
`supplied_address[].` `street_address`	**string** The street/postal address component (Address line 1) **Example:** `UNIT 1/12A SAMPLE STREET`

Field	Description
supplied_address[]. suburb	**string** The suburb component of the address **Example:** `SAMPLEVILLE`
supplied_address[]. state	**string** The state from the address (2 or 3 letter abbreviation, upper case) **Enum:** `ACT`, `NSW`, `NT`, `QLD`, `SA`, `TAS`, `VIC`, `WA` **Example:** `VIC`
supplied_address[]. postcode	**string** The 4 digit postcode of the address **Example:** `3999`
supplied_address[]. last_sale_date	**string (date|blank)** Date property was last for sale YYYY-MM-DD (if available) **Example:** `2019-01-01`
supplied_address[]. last_rental_date	**string (date|blank)** Date property was last for rent YYYY-MM-DD (if available) **Example:** `2022-03-15`
supplied_address[]. demographic_irsad	**string** The IRSD (Index of Relative Socio-economic Advantage and Disadvantage) for the address **Example:** `6`
supplied_address[]. demographic_ier	**string** The IER (Index of Education and Occupation) for the address **Example:** `5`
supplied_address[]. demographic_ieo	**string** The IEO (Index of Economic Resources) for the address **Example:** `4`
match_person	**object** The matched person details

Field	Description
match_person. match_result	**string** The person match type: • no_match: No matching person found • name_dob: Full name and date of birth match • full_name_exact: Full name match • first_name_initial: First name initial and last name match • last_name: Last name match • similar_last_name: Similar last name match • name_last_phone: Last name and phone number match • name_phone: Full name and phone number match • name_initial_phone: First name initial, last name and phone number match • name_email: Full name and email match • name_initial_email: First name initial, last name and email match • name_last_email: Last name and email match • phone_only: Phone number match • email_only: Email match, • phone_email: Phone number and email match • phone_hash: Phone number hash match • email_hash: Email hash match • address: Person at address **Enum:** `no_match`, `name_dob`, `full_name_exact`, `first_name_initial`, `last_name`, `similar_last_name`, `name_last_phone`, `name_phone`, `name_initial_phone`, `name_email`, `name_initial_email`, `name_last_email`, `phone_only`, `email_only`, `phone_email`, `phone_hash`, `email_hash`, `address` **Example:** `full_name_exact`
match_person. first_name	**string** The first name of the individual **Example:** `Jane`
match_person. middle_name	**string** The middle name of the individual **Example:** `Sarah`
match_person. last_name	**string** The last name of the individual **Example:** `Smith`

Field	Description
match_person. dob	**string (date)** The date of birth of the individual **Example:** `1990-08-12`
match_person. gender	**string** The gender of the person **Enum:** `MALE`, `FEMALE`, `X` **Example:** `FEMALE`
match_person. judgements	**string** Count of any judgements associated with the person (if available) **Example:** `1`
match_person. business	**string** List of any ABN records possibly associated with the person **Example:** `ABN 12345678901,ABN 9876432109`
match_person. deceased	**string or null** The deceased status of the person: - Y: The person is recorded as deceased **Enum:** `Y`
additional_address	**array of objects** Additional address associated with the record.
additional_address[]. address_id	**string** The unique identifier for the address **Example:** `ABCDE123456789`
additional_address[]. street_address	**string** The street/postal address component (Address line 1) **Example:** `UNIT 1/12A SAMPLE STREET`
additional_address[]. suburb	**string** The suburb component of the address **Example:** `SAMPLEVILLE`

Field	Description
additional_address[]. state	**string** The state from the address (2 or 3 letter abbreviation, upper case) **Enum:** `ACT` , `NSW` , `NT` , `QLD` , `SA` , `TAS` , `VIC` , `WA` **Example:** `VIC`
additional_address[]. postcode	**string** The 4 digit postcode of the address **Example:** `3999`
additional_address[]. last_sale_date	**string (date|blank)** Date property was last for sale YYYY-MM-DD (if available) **Example:** `2019-01-01`
additional_address[]. last_rental_date	**string (date|blank)** Date property was last for rent YYYY-MM-DD (if available) **Example:** `2022-03-15`
additional_address[]. demographic_irsad	**string** The IRSD (Index of Relative Socio-economic Advantage and Disadvantage) for the address **Example:** `6`
additional_address[]. demographic_ier	**string** The IER (Index of Education and Occupation) for the address **Example:** `5`
additional_address[]. demographic_ieo	**string** The IEO (Index of Economic Resources) for the address **Example:** `4`
phones	**array** Additional phone numbers for this record.
emails	**array** Additional email addresses for this record.

Field	Description
court_records	**array of objects** Court records associated with the person. The records are included based on the name and state appearing in public court records, and must not be taken to directly represent the person from this search record. Organisations must make their own enquiries as to the accuracy and appropriate use of this information.
court_records[]. fullname	**string** The full name of the person **Example:** `SMITH, JOHN ANDREW`
court_records[]. date	**string (date)** The date of the court record **Example:** `2022-03-15`
court_records[]. court	**string** The court where the record was found **Example:** `MAGISTRATES COURT`
court_records[]. location	**string** The location of the court **Example:** `SYDNEY`
court_records[]. state	**string** The state of the court **Enum:** `ACT`, `NSW`, `NT`, `QLD`, `SA`, `TAS`, `VIC`, `WA` **Example:** `NSW`
court_records[]. case_no	**string** The case number of the court record **Example:** `123456789`
court_records[]. listing	**string** The listing of the court record **Enum:** `civil`, `criminal` **Example:** `civil`

Field	Description
court_records[]. case_title	**string** The title of the court case **Example:** `SMITH V JONES`
court_records[]. listing_type	**string** The type of listing **Enum:** `civil`, `criminal` **Example:** `civil`
court_records[]. additional_info	**string** Additional information about the court record **Example:** `Defence To Claim, Vehicle Property Damage (Other)`
social_records	**array of objects** Social records associated with the emails or phone records.
social_records[]. social_reference	**string** The email address or phone number to which this social profile is linked **Example:** `person@example.com`
social_records[]. social_detail	**object**
social_records[]. social_detail. person	**array of objects** The social person record
social_records[]. social_detail. person[]. name_combined	**string** The combined name of the person **Example:** `John Andrew Smith`
social_records[]. social_detail. person[]. name_first	**string** The first name of the person **Example:** `John`
social_records[]. social_detail. person[]. name_middle	**string** The middle name of the person **Example:** `Andrew`

Field	Description
social_records[]. social_detail. person[]. name_last	**string** The last name of the person **Example:** `Smith`
social_records[]. social_detail. person[]. gender	**string** The gender of the person **Example:** `male`
social_records[]. social_detail. person[]. dob	**string** The date (or year) of birth of the person **Example:** `1990`
social_records[]. social_detail. location	**string** The location of the social profile **Example:** `Queensland, Australia`
social_records[]. social_detail. employment	**array of objects**
social_records[]. social_detail. employment[]. company	**string** The company name **Example:** `Example Company`
social_records[]. social_detail. employment[]. title	**string** The title at the company **Example:** `CEO`
social_records[]. social_detail. employment_history	**array of objects**
social_records[]. social_detail. employment_history[]. company	**string** The company name **Example:** `Example Company`
social_records[]. social_detail. employment_history[]. title	**string** The title at the company **Example:** `CEO`

Field	Description
social_records[]. social_detail. employment_history[]. date_start	**string (date)** The start date of the employment **Example:** `2010-01-01`
social_records[]. social_detail. employment_history[]. date_end	**string (date)** The end date of the employment **Example:** `2022-03-15`
social_records[]. social_detail. emails	**array of strings** The email addresses associated with the person
social_records[]. social_detail. phones	**array of strings** The phone numbers associated with the person
social_records[]. social_detail. skills	**array of strings** The skills associated with the person
social_records[]. social_detail. interests	**array of strings** The interests associated with the person
social_records[]. social_detail. education	**array of objects** The education associated with the person
social_records[]. social_detail. education[]. school	**string** The institution name **Example:** `Example University`
social_records[]. social_detail. education[]. type	**string** The type of institution **Example:** `post-secondary institution`
social_records[]. social_detail. education[]. url	**string** The URL of the institution **Example:** `sample.edu.au`
social_records[]. social_detail. education[]. studies	**array** The studies undertaken

Field	Description
social_records[]. social_detail. education[]. date_start	**string (date)** The start date of the education **Example:** `2010-01-01`
social_records[]. social_detail. education[]. date_end	**string (date)** The end date of the education **Example:** `2014-12-31`
social_records[]. social_detail. socials	**array of objects** Other social profiles associated with the person
social_records[]. social_detail. socials[]. url	**string** The URL of the social profile **Example:** `https://www.facebook.com/person`
social_records[]. social_detail. socials[]. network	**string** The social network of the profile **Example:** `Facebook`
social_records[]. social_detail. socials[]. username	**string** The username of the social profile **Example:** `person.12345`

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "supplied_address": [
        {
            "match_result": "address_match",
            "address_id": "ABCDE123456789",
            "street_address": "UNIT 1/12A SAMPLE STREET",
            "suburb": "SAMPLEVILLE",
            "state": "VIC",
            "postcode": "3999",
            "last_sale_date": "2019-01-01",
            "last_rental_date": "2022-03-15",
            "demographic_irsad": "6",
            "demographic_ier": "5",
            "demographic_ieo": "4"
        }
    ],
    "match_person": {
```

```
        "match_result": "full_name_exact",
        "first_name": "Jane",
        "middle_name": "Sarah",
        "last_name": "Smith",
        "dob": "1990-08-12",
        "gender": "FEMALE",
        "judgements": "1",
        "business": "ABN 12345678901,ABN 9876432109",
        "deceased": "Y"
    },
    "additional_address": [
        {
            "address_id": "ABCDE123456789",
            "street_address": "UNIT 1/12A SAMPLE STREET",
            "suburb": "SAMPLEVILLE",
            "state": "VIC",
            "postcode": "3999",
            "last_sale_date": "2019-01-01",
            "last_rental_date": "2022-03-15",
            "demographic_irsad": "6",
            "demographic_ier": "5",
            "demographic_ieo": "4"
        }
    ],
    "phones": [
        "0299995000",
        "0299995001"
    ],
    "emails": [
        "person@example.com",
        "person@example.edu"
    ],
    "court_records": [
        {
            "fullname": "SMITH, JOHN ANDREW",
            "date": "2022-03-15",
            "court": "MAGISTRATES COURT",
            "location": "SYDNEY",
            "state": "NSW",
            "case_no": "123456789",
            "listing": "civil",
            "case_title": "SMITH V JONES",
            "listing_type": "civil",
            "additional_info": "Defence To Claim, Vehicle Property Damage (Other)"
        }
    ],
    "social_records": [
        {
            "social_reference": "person@example.com",
            "social_detail": {
                "person": [
                    {
                        "name_combined": "John Andrew Smith",
                        "name_first": "John",
                        "name_middle": "Andrew",
                        "name_last": "Smith",
                        "gender": "male",
                        "dob": "1990"
                    }
                ],
                "location": "Queensland, Australia",
```

```
                "employment": [
                    {
                        "company": "Example Company",
                        "title": "CEO"
                    }
                ],
                "employment_history": [
                    {
                        "company": "Example Company",
                        "title": "CEO",
                        "date_start": "2010-01-01",
                        "date_end": "2022-03-15"
                    }
                ],
                "emails": [
                    "person@example.com",
                    "person@example.edu"
                ],
                "phones": [
                    "0299995000",
                    "0299995001"
                ],
                "skills": [
                    "Python",
                    "Java"
                ],
                "interests": [
                    "Gardening",
                    "Cooking"
                ],
                "education": [
                    {
                        "school": "Example University",
                        "type": "post-secondary institution",
                        "url": "sample.edu.au",
                        "studies": [
                            "mathematics"
                        ],
                        "date_start": "2010-01-01",
                        "date_end": "2014-12-31"
                    }
                ],
                "socials": [
                    {
                        "url": "https://www.facebook.com/person",
                        "network": "Facebook",
                        "username": "person.12345"
                    }
                ]
            }
        }
    ]
}
```


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

Banned and Disqualified Person Search

POST `/banned_disqualified_persons`

Banned and Disqualified Person Search searches multiple public authorities for persons who have been banned or disqualified from performing a role.

The sources used in this search include:

- ASIC Banned Persons Register
- ATO Disqualified SMSF Trustees (as published by the ATO)

This endpoint allows multiple requests to be simultaneously made in a single query.

Search Types

The search types affect how the names are matched. One of these three search types must be provided.

Search Type	Description
narrow_search	A narrow search will match exact first name, exact last name. The middle name is matched based on the similarity_threshold. This is the most restrictive search and should be used for low risk searches.
medium_search	A medium search will match exact last name, and will allow for a partial match on other names based on the similarity_threshold. This is a good balance between accuracy and flexibility.
broad_search	A broad search will match partial first name, partial middle name, and partial last name. This is the most flexible search and should be used for high risk searches.

Similarity Threshold

The `similarity_threshold` parameter controls how closely a returned name must match the search input to be included in the results. For each record, the API calculates a similarity score by comparing the search values (first, middle, and last names) against the result's names by measuring their likeness. This likeness is calculated by comparing the characters in the search name and the result name, measuring how much of the text overlaps, and expressing that as a percentage similarity.

These individual scores are averaged into an overall similarity percentage, which is stored with each result. The `similarity_threshold` then acts as a filter to only results with an average similarity equal to or greater than the threshold are returned. The returned result includes the overall similarity percentage in the `name_similarity` property.

Banned Types

The `banned_types` parameter controls which types of banned person to search for. **If omitted, the API will search for all types.**

The available types are:

Banned Type	Description
afs_banned_disqualified	A financial service provider banned or disqualified under section 922A(2) of Corporations Act 2001
banned_futures	Banned futures representatives register (pre-AFS licences)

Banned Type	Description
banned_securities	Banned securities representatives register (pre-AFS licences)
credit_banned_disqualified	Persons against whom a banning order or disqualification order is made under Part 2-4 of the NCCP Act
disqualified_director	The company directors and other office holders disqualified under 1274AA of Corporations Act 2001
disqualified_smsf	Disqualified SMSF auditors for whom an order disqualifying the person from being an approved SMSF auditor is in force under 130F of the SIS Act
ato_disqualified_trustee	Persons disqualified from acting as a trustee of an SMSF

Sandbox environment data

When simulating queries in the sandbox environment, the following records will return a match. Any other name returns an empty `matches` array. The same names are seeded into both registers, so an unfiltered search for one of them will typically return a record from each.

ASIC banned persons register

First name	Middle name	Last name	Banned type	Suburb	State
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	-	-
Adam	Darren	Halliday	disqualified_director	Illawong	NSW

ATO disqualified trustees register (returned as `banned_type: ato_disqualified_trustee`)

First name	Middle name	Last name	Suburb	State	Disqualified
John	Michael	Smith	Shepparton	VIC	2021-02-01
John	Robert	Smith	Sydney	NSW	2021-03-01
Jane	Penny Sally	Roberts	Camberwell	VIC	2021-04-01
Jane	-	Roberts	Camberwell	VIC	2021-05-01
J	S	Roberts	Melbourne	VIC	2021-06-01
P	-	Zao	Woolongong	NSW	2021-07-01
Simone	N	Zao	-	-	2021-08-01
Adam	Darren	Halliday	Illawong	NSW	2021-09-01

Request body

The details of the individuals to search for.

Field	Description
`search_type`	**string** The type of search to perform. Narrow searches for exact matches, medium searches for similar matches and broad searches for partial matches. **Enum:** `narrow_search`, `medium_search`, `broad_search` **Example:** `medium_search`
`banned_types`	**array of strings** The type of banned person to search for. Leave empty to search for all types. **Enum:** `afs_banned_disqualified`, `banned_futures`, `banned_securities`, `credit_banned_disqualified`, `disqualified_director`, `disqualified_smsf`, `ato_disqualified_trustee`
`similarity_threshold`	**number [0..100]** The minimum similarity percentage to return results for. **Default:** `80` **Example:** `80`
`requests`	**array of objects** The list of requests to make.
`requests[].` `check_id`	**string** The unique identifier for this request. This can be used match the request in the results. **Example:** `abcd123456789`
`requests[].` `first_name`	**string [1..50 characters]** The first name of the individual **Example:** `Phil`
`requests[].` `middle_name`	**string [1..50 characters] or null** The middle name of the individual
`requests[].` `last_name`	**string [1..50 characters]** The last name of the individual **Example:** `Smith`

Sample request

```
{
    "search_type": "medium_search",
    "banned_types": [
        "afs_banned_disqualified"
    ],
    "similarity_threshold": 80,
    "requests": [
        {
            "check_id": "abcd123456789",
            "first_name": "Phil",
            "middle_name": null,
            "last_name": "Smith"
        }
    ]
}
```

Responses

200 **Successful 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`
`function`	**string** The function that was called. **Example:** `banned_disqualified_persons`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`results`	**array of objects** A list of results for each of the requests made.
`results[].` `check_id`	**string** The unique identifier for this request. This can be used match the request in the results. **Example:** `abcd123456789`
`results[].` `matches`	**array of objects** Array of matched individuals.

Field	Description
results[]. matches[]. first_name	**string** The first name of the individual. **Example:** `Phil`
results[]. matches[]. middle_name	**string or null** The middle name of the individual. **Example:** `Michael`
results[]. matches[]. last_name	**string** The last name of the individual. **Example:** `Smith`
results[]. matches[]. banned_type	**string** The type of banned person the individual is. **Example:** `afs_banned_disqualified`
results[]. matches[]. document_number	**string** Document or reference number used to identify the document containing the ban or disqualification notice/order. **Example:** `1234567890`
results[]. matches[]. start_date	**string** The start date of the ban or disqualification. **Example:** `2021-01-01`
results[]. matches[]. end_date	**string or null** The end date of the ban or disqualification. **Example:** `2021-01-01`
results[]. matches[]. address_suburb	**string or null** The suburb of the individual. **Example:** `Sydney`
results[]. matches[]. address_state	**string or null** The state of the individual. **Example:** `NSW`

Field	Description
results[]. matches[]. address_postcode	**string or null** The postcode of the individual. **Example:** `2000`
results[]. matches[]. address_country	**string or null** The 2 letter ISO 3166-2 country code **Example:** `AU`
results[]. matches[]. comments	**string or null** Additional information associated with the banned or disqualified person. **Example:** `This is a comment`
results[]. matches[]. source_url	**string or null** The URL of the document containing the ban or disqualification notice/order (if available). **Example:** `https://www.example.com/media/61424.pdf`
results[]. matches[]. name_similarity	**number** The similarity percentage of the name of the individual to the search name. **Example:** `100`

Sample response


```
{
    "message": "Ok",
    "function": "banned_disqualified_persons",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "results": [
        {
            "check_id": "abcd123456789",
            "matches": [
                {
                    "first_name": "Phil",
                    "middle_name": "Michael",
                    "last_name": "Smith",
                    "banned_type": "afs_banned_disqualified",
                    "document_number": "1234567890",
                    "start_date": "2021-01-01",
                    "end_date": "2021-01-01",
                    "address_suburb": "Sydney",
                    "address_state": "NSW",
                    "address_postcode": "2000",
                    "address_country": "AU",
                    "comments": "This is a comment",
                    "source_url": "https://www.example.com/media/61424.pdf",
                    "name_similarity": 100
```

```
                }
            ]
        }
    ]
}
```

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

Check phone connectivity

GET `/phone_ping`

Tests the connectivity of one or more supplied phone numbers. This connectivity check will return a code indicating if the phone number is connected or disconnected.

Sandbox environment

When simulating queries in the sandbox environment, the connectivity response will be determined by the phone number ending:

Number Type	Response	Description
Any number ending in 9999		Simulate a timeout, will never return a result
Any number ending in 999	C	Simulate a slow result, takes 20 seconds but returns as connected
Any number ending in 998	D	Simulate a slow result, takes 20 seconds but returns as disconnected
Any number ending in 997	U	Simulate a slow result, takes 20 seconds but returns as undetermined
Any number ending in 99	I	Phone number Not Supported / Invalid
Any number ending in 9	D	Phone number is not connected
Any number ending in 8	U	Phone number is undetermined
Any other number	C	Phone Number is connected

Request body

The details of the record to enhance

Field	Description
`phones`	**string** One or more (comma separated) 10 digit Australian phone number to be checked (0NSN format) **Example:** `0417034325`

Field	Description
`timeout`	**integer [10..300] or null** Maximum number of seconds to wait for connectivity results before returning. If a result is not available within this time, the phone will be returned with a ping_result of `U` (Undetermined). Must be between 10 and 300 seconds. When supplied, this value is used exactly as given and overrides the default. If not specified, a default timeout is used that automatically scales with the number of phones submitted (30 seconds for fewer than 10 phones, increasing with larger batches). When checking many numbers it is recommended to either omit this parameter so the scaling default applies, or to supply a value large enough for all results to be returned. **Example:** `60`

Sample request

```
{
    "phones": "0417034325",
    "timeout": 60
}
```

Responses

200 **Result of phone ping check** (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)** **Example:** `738d9a89-b6fe-4fc1-96be-1389b2a506a2`
`results`	**array of objects** The checked phone numbers and their connectivity status
`results[].` `phone`	**string** The checked phone number in 0NSN format **Example:** `0417034325`

Field	Description
results[]. ping_result	**string** The result of the connectivity check for the phone - `C` - The phone is connected - `D` - The phone is disconnected - `U` - The state of the phone number cannot be detected - `I` - The phone number is invalid **Enum:** `C`, `D`, `U`, `I` **Example:** `C`
results[]. info	**string** Additional information about the result **Example:** `Number Live`
results[]. carrier	**string** The carrier of the phone number (if available) **Example:** `Telstra`
results[]. geo_location	**string** Approximate geographic location for the phone number (if available) **Example:** `Sydney`

Sample response

```
{
    "message": "Ok",
    "api_reference": "738d9a89-b6fe-4fc1-96be-1389b2a506a2"
}
```

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

Check phone DNC status

GET `/phone_dnc`

Tests the DNC status of one or more supplied phone numbers against the Australian DNC register. This DNC check will return a Y/N result indicating if the phone number is on the DNC register or not.

Sandbox environment

When simulating queries in the sandbox environment, the DNC response will be determined by the phone number ending:

Number Type	Response	Description
Any number ending in 9999	U	Simulate a long timeout
Any number ending in 999	Y	Simulate a slow result, takes 20 seconds but returns phone number on the DNC register
Any number ending in 998	N	Simulate a slow result, takes 20 seconds but returns phone number not on the DNC register
Any number ending in 997	U	Simulate a slow result, takes 20 seconds but returns as undetermined
Any number ending in 99	I	Phone number Not Supported / Invalid
Any number ending in 9, 8, 7, 6, 5, 4	N	Phone number not on the DNC register
Any number ending in 3, 2, 1, 0	Y	Phone number on the DNC register

> Note: Phone numbers must start with a valid Australian prefix, ie 02, 03, 04, 07, 08. Any other number will return as Invalid.

Request body

The details of the record to enhance

Field	Description
`phones`	**string** One or more (comma separated) 10 digit Australian phone number to be checked (0NSN format) **Example:** `0417034325`

Sample request

```
{
    "phones": "0417034325"
}
```

Responses

200 **Result of phone DNC check** (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`

Field	Description
`api_reference`	**string (uuid)** **Example:** `738d9a89-b6fe-4fc1-96be-1389b2a506a2`
`records`	**array of objects** The checked phone numbers and their DNC status
`records[].` `phone`	**string** The checked phone number in 0NSN format **Example:** `0417034325`
`records[].` `dnc_result`	**string** The result of the DNC check for the phone number - `Y` - The phone number is on the DNC register - `N` - The phone number is not on the DNC register - `I` - The phone number is invalid - `U` - The state of the phone number cannot be checked **Enum:** `Y`, `N`, `I`, `U` **Example:** `Y`
`records[].` `dnc_reference`	**string** The reference number for the DNC check **Example:** `1234567890`
`records[].` `info`	**string** Human readable information about the result **Example:** `Phone is on the DNC register`

Sample response

```
{
    "message": "Ok",
    "api_reference": "738d9a89-b6fe-4fc1-96be-1389b2a506a2"
}
```

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

Phone Contact Lookup

POST `/phone_contact_lookup`

The Phone Contact Lookup API allows you to retrieve the person or persons linked to a specific Australian phone number in the Global Data Universe.

The API requires a phone number to be provided and will return the following data if the phone number is found to be associated with a person in the Global Data Universe:

- First Name
- Middle Name
- Last Name
- Address
- Suburb
- State
- Postcode
- Date the record was last seen

If multiple people are linked to the phone number, the API will return multiple records (up to a maximum of 5).

Batch processing

The API can accept up to 100 phone numbers per request. Each result will include the phone number in the response and will be returned in the same order as the phone numbers in the request.

Sandbox environment data

When processing queries in the sandbox environment, the following records will return a match:

Phone Number	First Name	Middle Name	Last Name	Address	Suburb	State	Postcode	Date Last Seen
+61399999999	John	Andrew	Smith	20 Hardy St	Lilydale	VIC	3140	2019-02-20
+61491222111	John	Andrew	Smith	20 Hardy St	Lilydale	VIC	3140	2019-02-20
+61491222111	John	Andrew	Smith	8 Waltham St	Richmond	VIC	3121	2027-04-01
+61427151494	Mark		Wood	PO Box 8144	Australian National University	ACT	0200	2019-07-17
+61491222444	Mary	Sally	Jones	35 Yaralla St	Concord West	NSW	2138	2099-01-01
+61491222333	Robert		Brown	6 Waterview Cl	Port Macquarie	NSW	2444	2099-07-19
+61880885999	Marion		Filewood	375 Argent St	Broken Hill	NSW	2880	2002-07-16
+61880885999	Aaron	Albert	Filewood	375 Argent St	Broken Hill	NSW	2880	2002-07-16

Phone Number	First Name	Middle Name	Last Name	Address	Suburb	State	Postcode	Date Last Seen
+61412024869	Marion		Filewood	375 Argent St	Broken Hill	NSW	2880	2002-07-16

> **Note:** This API returns PII and as such requires an appropriate privacy policy and PII handling processes to be in place.

Request body

The phone numbers to search for

Field	Description
`phones`	**array of strings [items 1..32 characters]** The phone numbers to search for in E.164 format.

Sample request

```
{
    "phones": [
        "+61491222111",
        "+61491222112",
        "+6100000"
    ]
}
```

Responses

200 **Result of the check** (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. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`results`	**array of objects** Array of results for each phone number.

Field	Description
results[]. phone	**string** The phone number in E.164 format.
results[]. message	**string** The result of the search for the phone number. This will be `Ok` if the search was successful (whether it returned a match or not) or an error message if the search could not be completed. For example 'Error: Invalid phone number'.
results[]. matches	**array of objects** Array of matched records for the phone number.
results[]. matches[]. first_name	**string** The first name of the matched individual.
results[]. matches[]. middle_name	**string or null** The middle name of the matched individual.
results[]. matches[]. last_name	**string** The last name of the matched individual.
results[]. matches[]. address	**string** The address of the matched individual.
results[]. matches[]. suburb	**string** The suburb of the matched individual.
results[]. matches[]. state	**string** The state of the matched individual.
results[]. matches[]. postcode	**string** The postcode of the matched individual.
results[]. matches[]. date_last_seen	**string (date)** The date the address record was last seen in the Global Data Universe

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "results": [
        {
            "phone": "+61491222111",
```

```
            "message": "Ok",
            "matches": [
                {
                    "first_name": "John",
                    "middle_name": "Andrew",
                    "last_name": "Smith",
                    "address": "20 HARDY ST",
                    "suburb": "LILYDALE",
                    "state": "VIC",
                    "postcode": "3140",
                    "date_last_seen": "2022-03-15"
                }
            ]
        },
        {
            "phone": "+61491222112",
            "message": "Ok",
            "matches": []
        },
        {
            "phone": "+6100000",
            "message": "Error: Invalid phone number",
            "matches": []
        }
    ]
}
```


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

Phone Contact Validate

POST `/phone_contact_validate`

The Phone Contact Validate API allows you to validate the link between a specific Australian phone number and a person associated with that phone number in the Global Data Universe.

The API requires a phone number and persons details to be provided and will return a match if the phone number is found to be associated with that person in the Global Data Universe. If no match is found, the API will return a `no_match` result. If the phone number is not a valid Australian phone number or could not be searched, the API will return an `error` result.

Request Parameters:

- Phone Number
- First Name
- Middle Name
- Last Name

Batch processing

The API can accept up to 100 requests per call. The response will include the results for each request in the same order as the requests were made.

Sandbox environment data

When processing queries in the sandbox environment, the following records are available:

Phone Number	First Name	Middle Name	Last Name	Address	Suburb	State	Postcode	Date Last Seen
+61399999999	John	Andrew	Smith	20 Hardy St	Lilydale	VIC	3140	2019-02-20
+61491222111	John	Andrew	Smith	20 Hardy St	Lilydale	VIC	3140	2019-02-20
+61491222111	John	Andrew	Smith	8 Waltham St	Richmond	VIC	3121	2027-04-01
+61427151494	Mark		Wood	PO Box 8144	Australian National University	ACT	0200	2019-07-17
+61491222444	Mary	Sally	Jones	35 Yaralla St	Concord West	NSW	2138	2099-01-01
+61491222333	Robert		Brown	6 Waterview Cl	Port Macquarie	NSW	2444	2099-07-19
+61880885999	Marion		Filewood	375 Argent St	Broken Hill	NSW	2880	2002-07-16
+61880885999	Aaron	Albert	Filewood	375 Argent St	Broken Hill	NSW	2880	2002-07-16
+61412024869	Marion		Filewood	375 Argent St	Broken Hill	NSW	2880	2002-07-16

Name match results

Before the matching process, names are standardised for comparison and any aliases are resolved.

The `first_name_match` and `last_name_match` properties will return with one of the `exact`, `partial`, `alias` or `no_match` values. The `last_name_match` property will return with one of the `exact` or `no_match` values.

Name Match Result	Description
`exact`	An exact match was found for the name. This is the best match result and will be returned if the name is an exact match for the name in the record.
`partial`	A partial substring match was found for the name. This will be returned if the name is a substring match starting with the name in the record. For example, if the name in the record is "Johnathan" and the name in the request is "John" or "J", the match result will be `partial`.
`alias`	An alias match was found for the name. This will be returned if the name is an alias match for the name in the record. For example, if the name in the record is "William" and the name in the request is "Bill" or "Willy", the match result will be `alias`.
`no_match`	No match was found for the name.

Note: a last name must match exactly for a match to be returned.

The returned `result` parameter will be determined by the following rules:

- If all of the name match results are `exact`, then the result will be `match`.
- If all of the name match results are `no_match`, then the result will be `no_match`.
- If any of the name match results are `partial` or `alias`, then the result will be `possible_match`.

Request body

The phone number and persons details to validate

Field	Description
`requests`	**array of objects** The list of requests to make.
`requests[].` `phone`	**string** The phone number to validate in E.164 format.
`requests[].` `first_name`	**string** The first name of the person to validate.
`requests[].` `middle_name`	**string or null** The optional middle name of the person to validate.
`requests[].` `last_name`	**string** The last name of the person to validate.

Sample request

```
{
    "requests": [
        {
            "phone": "+61491222111",
            "first_name": "Johnny",
            "middle_name": "A",
            "last_name": "Smith"
        },
        {
            "phone": "+61491222444",
            "first_name": "Paul",
            "middle_name": null,
            "last_name": "Smitherton"
        }
    ]
}
```

Responses

200 **Successful 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. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`results`	**array of objects** A list of results for each of the requests made.
`results[].` `phone`	**string** The phone number from the request in E.164 format.
`results[].` `result`	**string** The result of the match check for the phone number in the request. • `match` : An exact match was found for the named person. • `no_match` : No match was found for the named person. • `possible_match` : A possible match is returned when the `first_name` , or `middle_name` properties had a partial or alias match. • `error` : The request was skipped because the phone number is not a valid Australian phone number or could not be searched. **Enum:** `match` , `no_match` , `possible_match` , `error`
`results[].` `message`	**string** A message indicating the result of the request. This will be `Ok` if the request was successful or an error message if the request could not be completed. For example `Error: Invalid phone number` .
`results[].` `first_name_match`	**string or null** The result of the match check for the first name. This will be null if the match_result is `error` . • `exact` : An exact match was found for the first name. • `partial` : A partial substring match was found for the first name. • `alias` : An alias match was found for the first name. • `no_match` : No match was found for the first name. **Enum:** `exact` , `partial` , `alias` , `no_match`

results[]. middle_name_match	**string or null** The result of the match check for the middle name. This will be null if the match_result is `error` or if the middle name was not provided in the request. - `exact` : An exact match was found for the middle name. - `partial` : A partial substring match was found for the middle name. - `alias` : An alias match was found for the middle name. - `no_match` : No match was found for the middle name. **Enum:** `exact` , `partial` , `alias` , `no_match`
results[]. last_name_match	**string or null** The result of the match check for the last name. This will be null if the match_result is `error` . **Enum:** `exact` , `no_match`

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "results": [
        {
            "phone": "+61491222111",
            "result": "possible_match",
            "message": "Ok",
            "first_name_match": "alias",
            "middle_name_match": "partial",
            "last_name_match": "exact"
        },
        {
            "phone": "+61491222444",
            "result": "no_match",
            "message": "Ok",
            "first_name_match": "no_match",
            "middle_name_match": "no_match",
            "last_name_match": "no_match"
        }
    ]
}
```


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

PEP / Sanction Search

POST `/pep_sanction_check`

PEP / Sanction Search is a powerful API query that can be used to find individuals or companies who are politically exposed persons or have an international sanction applied to them. The results provide a list of pep or sanction sources that the individuals or companies are listed on and there is the option to tailor the search to narrow, medium or broad.

Search Types

Narrow Search

A narrow search will match exact first name, exact last name and exact birth date. This is the most restrictive search and should be used for low risk searches. For companies, the search will match companies with names which contain the exact names provided.

Medium Search

A medium search will match exact last name and birth date year, and will allow for a partial match on other names. This is a good balance between accuracy and flexibility. For companies, this search will match companies names which contain the partial names provided. However the results will filter out any results where the matched name only consists of words from a stop word list. The stop word list is a list of common words that are not useful for matching, such as "group", "company", "LTD", etc.

Broad Search

A broad search will match partial first name, partial last name and partial birth date. This is the most flexible search and should be used for high risk searches. For companies, this search type will match companies with names which contain the partial names provided.

Note on Matching

When matching names, the API ignores the order of the names as some names may be reported in different orders. Consequently, it is possible to search for John Michael and find a match for an individual with a reported name of Michael John.

When matching company names, the API will ignore birth dates and pep countries.

Sandbox environment data

When simulating queries in the sandbox environment, the following records will return a match:

Individuals

First name	Middle name	Last name	Birth date	PEP?	Sanctioned?	Country

Ed	Virgil	Leuschke	1993-03-23	No	Yes	AU
Justus	Tanner	Beatty	1950-06-04	Yes	No	AU
Julien	-	Kovacek	2002-08-14	Yes	No	AU
Alford	-	McGlynn	1994-03-13	No	Yes	AU

Companies

Company name	Sanction country
Southern Cross Commodities Pty Ltd	AU, US
Baltic Petrochem Logistics OU	EU, GB
Jade Meridian Holdings Pte Ltd	US
Andes Orbital Freight SAC	GB
Sahel Mineral Ventures SARL	UN, AU

Request body

The details of the individual to search for.

Field	Description
`company_name`	**string [1..255 characters] or null** The name of the company to search for
`vessel_name`	**string [1..255 characters] or null** The name of the vessel to search for
`first_name`	**string [1..255 characters]** The first name of the individual **Example:** `Phil`
`middle_name`	**string [1..255 characters] or null** The middle name of the individual
`last_name`	**string [2..255 characters]** The last name of the individual **Example:** `Smith`
`full_name`	**string [2..255 characters] or null** The full name of the individual. This can be provided instead of first, middle and last names.

Field	Description
`birth_date`	**string (date)** The birth date of the individual **Example:** `1938-10-24`
`search_type`	**string** The type of search to perform. Narrow searches for exact matches, medium searches for similar matches and broad searches for partial matches. **Enum:** `narrow_search`, `medium_search`, `broad_search` **Example:** `broad_search`
`pep_countries`	**array of strings [items 2..7 characters]** A list of countries of the politically exposed person as a country code to include in the search
`sanction_countries`	**array of strings [items 2..7 characters]** A list of countries of the sanction as a country code to include in the search
`max_results`	**integer [1..100]** The maximum number of results to return **Default:** `20` **Example:** `20`
`similarity_threshold`	**number [0..100]** The minimum similarity percentage to return results for. **Default:** `80` **Example:** `80`
`pep_sanction_extended_result`	**boolean** Whether to return include the extended results for the individual matches **Default:** `false` **Example:** `true`

Sample request

```
{
    "company_name": null,
    "vessel_name": null,
    "first_name": "Phil",
```

```
    "middle_name": null,
    "last_name": "Smith",
    "full_name": null,
    "birth_date": "1938-10-24",
    "search_type": "broad_search",
    "pep_countries": [
        "au"
    ],
    "sanction_countries": [
        "au"
    ],
    "max_results": 20,
    "similarity_threshold": 80,
    "pep_sanction_extended_result": true
}
```

Responses

200 **Successful 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. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`function`	**string** The function that was called. **Example:** `pep_sanction_check`
`matches`	**array of objects** Array of matched individuals.
`matches[].uuid`	**string (uuid)** The unique identifier of the individual. **Example:** `389d8f0f-b2c4-465a-b2e9-8a14722031f7`
`matches[].caption`	**string** **Example:** `Phil Smith`

Field	Description
matches[]. is_pep	**boolean** Whether the individual is a politically exposed person. **Example:** `true`
matches[]. is_sanctioned	**boolean** Whether the individual is sanctioned. **Example:** `false`
matches[]. distance	**number** The levenshtein distance of the names of the individual to the search name. **Example:** `0`
matches[]. similarity	**number** The percentage similarity of the matched name to the search name. **Example:** `100`
matches[]. matched_name	**string** The name of the individual that was matched to the search name and used for the similarity score. **Example:** `PHIL SMITH`
matches[]. sanctions	**array of strings** Array of countries where the individual is sanctioned.
matches[]. pep_matches	**array of strings** Array of sources where the individual has been identified as a pep.
matches[]. sanction_matches	**array of strings** Array of sources where the individual has been identified as being sanctioned.
matches[]. data	**object** Additional data about the individual which is included if pep_sanction_extended_result is true.
more_results	**boolean** Whether there are more results available. **Example:** `false`

Sample response

```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "function": "pep_sanction_check",
    "matches": [
        {
            "uuid": "389d8f0f-b2c4-465a-b2e9-8a14722031f7",
            "caption": "Phil Smith",
            "is_pep": true,
            "is_sanctioned": false,
            "distance": 0,
            "similarity": 100,
            "matched_name": "PHIL SMITH",
            "sanctions": [],
            "pep_matches": [
                "wd_peps"
            ],
            "sanction_matches": [],
            "data": {
                "id": "Q21598350",
                "schema": "Person",
                "target": true,
                "caption": "Phil Smith",
                "datasets": [
                    "wd_peps",
                    "wikidata"
                ],
                "last_seen": "2024-06-14T12:48:02",
                "referents": [],
                "first_seen": "2023-04-20T10:30:17",
                "properties": {
                    "name": [
                        "??? ????",
                        "?????? ????",
                        "Phil Smith"
                    ],
                    "alias": [
                        "Philip John Smith"
                    ],
                    "notes": [
                        "Australian politician and teacher"
                    ],
                    "gender": [
                        "male"
                    ],
                    "topics": [
                        "role.pep"
                    ],
                    "country": [
                        "au"
                    ],
                    "keywords": [
                        "State government"
                    ],
                    "lastName": [
                        "Smith"
                    ],
                    "position": [
                        "Member of the Western Australian Legislative Assembly"
                    ],
```

```
                "birthDate": [
                    "1938-10-24"
                ],
                "firstName": [
                    "Philip",
                    "John"
                ],
                "modifiedAt": [
                    "2023-03-07"
                ],
                "wikidataId": [
                    "Q21598350"
                ],
                "nationality": [
                    "au"
                ],
                "positionOccupancies": [
                    {
                        "id": "wd-18fdca25ec2964d511fe7a039144bd178a3039f9",
                        "schema": "Occupancy",
                        "target": false,
                        "caption": "Occupancy",
                        "datasets": [
                            "wd_peps"
                        ],
                        "last_seen": "2024-06-14T12:48:02",
                        "referents": [],
                        "first_seen": "2023-09-08T07:00:40",
                        "properties": {
                            "post": [
                                {
                                    "id": "Q20165902",
                                    "schema": "Position",
                                    "target": false,
                                    "caption": "Member of the Western Australian Legislative Assembly",
                                    "datasets": [
                                        "ann_pep_positions",
                                        "wd_peps"
                                    ],
                                    "last_seen": "2024-06-14T14:38:06",
                                    "referents": [],
                                    "first_seen": "2023-09-08T07:00:40",
                                    "properties": {
                                        "name": [
                                            "Member of the Western Australian Legislative Assembly"
                                        ],
                                        "topics": [
                                            "gov.state"
                                        ],
                                        "country": [
                                            "au"
                                        ],
                                        "wikidataId": [
                                            "Q20165902"
                                        ]
                                    },
                                    "last_change": "2023-12-22T14:40:54"
                                }
                            ],
                            "holder": [
                                "Q21598350"
```

```
                                ],
                                "status": [
                                    "unknown"
                                ]
                            },
                            "last_change": "2023-09-08T07:00:40"
                        }
                    ]
                },
                "last_change": "2024-01-31T00:47:01"
            }
        }
    ],
    "more_results": false
}
```


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

PEP / Sanction Bulk Search

POST `/pep_sanction_checks`

PEP / Sanction Bulk Search is a powerful API query that can be used to find individuals or companies who are politically exposed persons or have an international sanction applied to them. The results provide a list of pep or sanction sources that the individuals or companies are listed on and there is the option to tailor the search to narrow, medium or broad.

This endpoint allows multiple requests to be simultaneously made in a single query.

The same configuration parameters are used for all requests in the requests array:

- search_type
- pep_countries
- sanction_countries
- max_results
- similarity_threshold
- pep_sanction_extended_result

Search Types

Narrow Search

A narrow search will match exact first name, exact last name and exact birth date. This is the most restrictive search and should be used for low risk searches. For companies, the search will match companies with names which contain the exact names provided.

Medium Search

A medium search will match exact last name and birth date year, and will allow for a partial match on other names. This is a good balance between accuracy and flexibility. For companies, this search will match companies names which contain the partial names provided. However the results will filter out any results where the matched name only consists of words from a stop word list. The stop word list is a list of common words that are not useful for matching, such as "group", "company", "LTD", etc.

Broad Search

A broad search will match partial first name, partial last name and partial birth date. This is the most flexible search and should be used for high risk searches. For companies, this search type will match companies with names which contain the partial names provided.

Note on Matching

When matching names, the API ignores the order of the names as some names may be reported in different orders. Consequently, it is possible to search for John Michael and find a match for an individual with a reported name of Michael John.

When matching company names, the API will ignore birth dates and pep countries.

Sandbox environment data

When simulating queries in the sandbox environment, the following records will return a match:

Individuals

First name	Middle name	Last name	Birth date	PEP?	Sanctioned?	Country
Ed	Virgil	Leuschke	1993-03-23	No	Yes	AU
Justus	Tanner	Beatty	1950-06-04	Yes	No	AU
Julien	-	Kovacek	2002-08-14	Yes	No	AU
Alford	-	McGlynn	1994-03-13	No	Yes	AU

Companies

Company name	Sanction country
Southern Cross Commodities Pty Ltd	AU, US
Baltic Petrochem Logistics OU	EU, GB
Jade Meridian Holdings Pte Ltd	US
Andes Orbital Freight SAC	GB
Sahel Mineral Ventures SARL	UN, AU

Request body

The details of the individual to search for.

Field	Description

`search_type`	**string** The type of search to perform. Narrow searches for exact matches, medium searches for similar matches and broad searches for partial matches. **Enum:** `narrow_search`, `medium_search`, `broad_search` **Example:** `broad_search`
`pep_countries`	**array of strings [items 2..7 characters]** A list of countries of the politically exposed person as a country code to include in the search
`sanction_countries`	**array of strings [items 2..7 characters]** A list of countries of the sanction as a country code to include in the search
`max_results`	**integer [1..100]** The maximum number of results to return **Default:** `20` **Example:** `20`
`similarity_threshold`	**number [0..100]** The minimum similarity percentage to return results for. **Default:** `80` **Example:** `80`
`pep_sanction_extended_result`	**boolean** Whether to return include the extended results for the individual matches **Default:** `false` **Example:** `true`
`requests`	**array of objects** The list of requests to make.
`requests[].` `check_id`	**string** The unique identifier for this request. This can be used match the request in the results. **Example:** `abcd123456789`
`requests[].` `company_name`	**string [1..255 characters] or null** The name of the company to search for

requests[]. vessel_name	**string [1..255 characters] or null** The name of the vessel to search for
requests[]. first_name	**string [1..255 characters]** The first name of the individual **Example:** `Phil`
requests[]. middle_name	**string [1..255 characters] or null** The middle name of the individual
requests[]. last_name	**string [2..255 characters]** The last name of the individual **Example:** `Smith`
requests[]. full_name	**string [2..255 characters] or null** The full name of the individual. This can be provided instead of first, middle and last names.
requests[]. birth_date	**string (date)** The birth date of the individual **Example:** `1938-10-24`

Sample request

```
{
    "search_type": "broad_search",
    "pep_countries": [
        "au"
    ],
    "sanction_countries": [
        "au"
    ],
    "max_results": 20,
    "similarity_threshold": 80,
    "pep_sanction_extended_result": true,
    "requests": [
        {
            "check_id": "abcd123456789",
            "company_name": null,
            "vessel_name": null,
            "first_name": "Phil",
            "middle_name": null,
            "last_name": "Smith",
            "full_name": null,
            "birth_date": "1938-10-24"
        }
    ]
}
```

Responses

200 **Successful 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`
`function`	**string** The function that was called. **Example:** `pep_sanction_check_bulk`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`results`	**array of objects** A list of results for each of the requests made.
`results[].` `check_id`	**string** The unique identifier for this request. This can be used match the request in the results. **Example:** `abcd123456789`
`results[].` `matches`	**array of objects** Array of matched individuals.

`results[].`
`matches[].`
`uuid`
	string (uuid)
	
	The unique identifier of the individual.
	
	Example: `389d8f0f-b2c4-465a-b2e9-8a14722031f7`

`results[].`
`matches[].`
`caption`
	string
	
	Example: `Phil Smith`

`results[].`
`matches[].`
`is_pep`
	boolean
	
	Whether the individual is a politically exposed person.
	
	Example: `true`

Field	Description
results[]. matches[]. is_sanctioned	**boolean** Whether the individual is sanctioned. **Example:** `false`
results[]. matches[]. distance	**number** The levenshtein distance of the names of the individual to the search name. **Example:** `0`
results[]. matches[]. similarity	**number** The percentage similarity of the matched name to the search name. **Example:** `100`
results[]. matches[]. matched_name	**string** The name of the individual that was matched to the search name and used for the similarity score. **Example:** `PHIL SMITH`
results[]. matches[]. sanctions	**array of strings** Array of countries where the individual is sanctioned.
results[]. matches[]. pep_matches	**array of strings** Array of sources where the individual has been identified as a pep.
results[]. matches[]. sanction_matches	**array of strings** Array of sources where the individual has been identified as being sanctioned.
results[]. matches[]. data	**object** Additional data about the individual which is included if pep_sanction_extended_result is true.
results[]. more_results	**boolean** Whether there are more results available. **Example:** `false`

Sample response


```
{
    "message": "Ok",
    "function": "pep_sanction_check_bulk",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "results": [
```

```
{
    "check_id": "abcd123456789",
    "matches": [
        {
            "uuid": "389d8f0f-b2c4-465a-b2e9-8a14722031f7",
            "caption": "Phil Smith",
            "is_pep": true,
            "is_sanctioned": false,
            "distance": 0,
            "similarity": 100,
            "matched_name": "PHIL SMITH",
            "sanctions": [],
            "pep_matches": [
                "wd_peps"
            ],
            "sanction_matches": [],
            "data": {
                "id": "Q21598350",
                "schema": "Person",
                "target": true,
                "caption": "Phil Smith",
                "datasets": [
                    "wd_peps",
                    "wikidata"
                ],
                "last_seen": "2024-06-14T12:48:02",
                "referents": [],
                "first_seen": "2023-04-20T10:30:17",
                "properties": {
                    "name": [
                        "??? ????",
                        "?????? ????",
                        "Phil Smith"
                    ],
                    "alias": [
                        "Philip John Smith"
                    ],
                    "notes": [
                        "Australian politician and teacher"
                    ],
                    "gender": [
                        "male"
                    ],
                    "topics": [
                        "role.pep"
                    ],
                    "country": [
                        "au"
                    ],
                    "keywords": [
                        "State government"
                    ],
                    "lastName": [
                        "Smith"
                    ],
                    "position": [
                        "Member of the Western Australian Legislative Assembly"
                    ],
                    "birthDate": [
                        "1938-10-24"
                    ],
                    "firstName": [
```

```
                "Philip",
                "John"
            ],
            "modifiedAt": [
                "2023-03-07"
            ],
            "wikidataId": [
                "Q21598350"
            ],
            "nationality": [
                "au"
            ],
            "positionOccupancies": [
                {
                    "id": "wd-18fdca25ec2964d511fe7a039144bd178a3039f9",
                    "schema": "Occupancy",
                    "target": false,
                    "caption": "Occupancy",
                    "datasets": [
                        "wd_peps"
                    ],
                    "last_seen": "2024-06-14T12:48:02",
                    "referents": [],
                    "first_seen": "2023-09-08T07:00:40",
                    "properties": {
                        "post": [
                            {
                                "id": "Q20165902",
                                "schema": "Position",
                                "target": false,
                                "caption": "Member of the Western Australian Legislative Assemb
                                "datasets": [
                                    "ann_pep_positions",
                                    "wd_peps"
                                ],
                                "last_seen": "2024-06-14T14:38:06",
                                "referents": [],
                                "first_seen": "2023-09-08T07:00:40",
                                "properties": {
                                    "name": [
                                        "Member of the Western Australian Legislative Assembly"
                                    ],
                                    "topics": [
                                        "gov.state"
                                    ],
                                    "country": [
                                        "au"
                                    ],
                                    "wikidataId": [
                                        "Q20165902"
                                    ]
                                },
                                "last_change": "2023-12-22T14:40:54"
                            }
                        ],
                        "holder": [
                            "Q21598350"
                        ],
                        "status": [
                            "unknown"
                        ]
                    },
```

```
                                    "last_change": "2023-09-08T07:00:40"
                                }
                            ]
                        },
                        "last_change": "2024-01-31T00:47:01"
                    }
                }
            ],
            "more_results": false
        }
    ]
}
```


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

Social Check

GET `/social_check`

The social_check method will perform a social media check using a phone number, email address, or social media profile URL. The social media check will return a number of data structures (where available) including:

- Social media profile links
- Associated email addresses and phone numbers
- Employment history
- Education history
- Interests and skills

Supported social media profile URLs

Social media profile URLs must be supplied as full URLs and must include a path identifying the specific profile. Bare handles such as `@johnsmith` are not accepted, and a domain on its own (e.g. `linkedin.com`) is also not accepted.

The following URL formats are all accepted (using `linkedin.com/in/johnsmith` as an example path):

- `https://www.linkedin.com/in/johnsmith`
- `http://www.linkedin.com/in/johnsmith`
- `https://linkedin.com/in/johnsmith`
- `http://linkedin.com/in/johnsmith`
- `www.linkedin.com/in/johnsmith`
- `linkedin.com/in/johnsmith`

In other words, the scheme (`http://` or `https://`) and the `www.` prefix are both optional, but the domain and a non-empty path are required. A trailing slash is allowed and will be stripped.

Profile URLs from the following networks are supported:

`facebook.com`, `fb.com`, `linkedin.com`, `twitter.com`, `x.com`, `xing.com`, `indeed.com`, `github.com`, `meetup.com`, `instagram.com`, `quora.com`, `gravatar.com`, `klout.com`, `stackoverflow.com`, `angellist.com`, `youtube.com`, `foursquare.com`, `pinterest.com`, `vimeo.com`, `aboutme.com`, `flickr.com`, `ello.com`, `crunchbase.com`, `dribbble.com`, `google.com`, `wordpress.com`, `myspace.com`, `behance.com`, `soundcloud.com`, `reddit.com`.

Sandbox environment data

When simulating queries in the sandbox environment, the following inputs will return a match. Any other input that passes validation will return `No match`. Every input listed against a single person resolves to the same record and so will share the same `person_id`, which can be used to demonstrate cross-input deduplication.

Person	Phone	Email	Profile URL

Mary Bird	+61449335480	mary.bird@example.com birdlady@example.com	linkedin.com/in/mary-bird-9rk5reef facebook.com/mary88-sample-user
Robert Jones		robert@example.com	facebook.com/robert-jones-sample-user
Michael Raymond		michael.raymond@example.com	linkedin.com/in/michael-raymond-sample

Profile URL inputs may include the `https://` (or `http://`) scheme and a `www.` prefix - they are normalised before lookup, so for example `https://www.linkedin.com/in/mary-bird-9rk5reef` will also return the Mary Bird record.

Request body

The details of the record to enhance

Field	Description
`records`	**string** Comma separated list of phone numbers, email addresses and/or social media profile URLs. **Note:** Phone numbers should be in E.164 format, or Australian 0NSN format. Social media profile URLs must include a path (e.g. `https://www.linkedin.com/in/johnsmith`) and must be from one of the supported networks listed above. **Example:** `test@example.com,0412345678,https://www.linkedin.com/in/johnsmith`

Sample request

```
{
    "records": "test@example.com,0412345678,https://www.linkedin.com/in/johnsmith"
}
```

Responses

200 **Result of social media check** (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)** **Example:** `738d9a89-b6fe-4fc1-96be-1389b2a506a2`
`records`	**array of objects** A set of matched records

Field	Description
records[]. record	**string** The phone number, email address or social media profile URL being checked (as supplied) **Example:** `test@example.com`
records[]. result	**string** Was a match found for this record? - `Match` - A match was found - `No Match` - No match was found - `Error` - An error occurred **Enum:** `Match`, `No Match`, `Error` **Example:** `Match`
records[]. likelihood	**integer [1..10]** A confidence score indicating the strength of the match, from 1 (lowest) to 10 (highest). This value reflects how closely the input data matched the returned record. **Example:** `8`
records[]. person_id	**string or null** A unique identifier for the matched person. This value is consistent across lookups, so if two different inputs (e.g. two different email addresses) resolve to the same person, they will share the same `person_id`. This can be used to deduplicate results. **Example:** `a3f8b2c1d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1`
records[]. location	**string** String representing general geographic location. The format will depend on the social media information available but will generally consist of suburb, state and country. **Example:** `sydney, new south wales, australia`
records[]. location_locality	**string** The locality (city/suburb) component of the person's location. **Example:** `sydney`
records[]. location_region	**string** The region (state/province) component of the person's location. **Example:** `new south wales`

Field	Description
records[]. location_country	**string** The country component of the person's location. **Example:** `australia`
records[]. person	**object** Details of the person associated with the record.
records[]. person. name_first	**string** The first name of the person **Example:** `John`
records[]. person. name_middle	**string** The middle name of the person **Example:** `Michael`
records[]. person. name_last	**string** The last name of the person **Example:** `Smith`
records[]. person. name_combined	**string** The full name of the person **Example:** `John Michael Smith`
records[]. person. gender	**string** The gender of the person **Enum:** `male`, `female` **Example:** `male`
records[]. person. dob	**string (date)** The date of birth of the person This can be either a full date in the format YYYY-MM-DD or a year only in the format YYYY **Example:** `1990-05-23`
records[]. person. industry	**string** The industry the person is associated with. **Example:** `accounting`

Field	Description
records[]. emails	**array of strings** A list of email addresses associated with the person
records[]. emails_detailed	**array of objects** A list of email addresses associated with the person, including type classification. This provides the same email addresses as the `emails` field but with additional metadata.
records[]. emails_detailed[]. address	**string** The email address **Example:** `test@example.com`
records[]. emails_detailed[]. type	**string or null** The classification of the email address, if known. Common values include: - `current_professional` - A current work email - `personal` - A personal email May be null if the type is unknown. **Example:** `current_professional`
records[]. phones	**array of strings** A list of phone numbers associated with the person (in E.164 format)
records[]. employment	**object** Current employment details of the person
records[]. employment. company	**string** The name of the company the person works for **Example:** `Sample Corp`
records[]. employment. title	**string** The job title of the person **Example:** `Software Engineer`
records[]. employment. company_website	**string** The website of the company the person works for **Example:** `samplecorp.com`

Field	Description
records[]. employment. company_size	**string** The approximate size of the company the person works for **Example:** `51-200`
records[]. employment. company_industry	**string** The industry of the company the person works for **Example:** `information technology and services`
records[]. employment. start_date	**string** The start date of the person's current employment. This can be a year/month in the format YYYY-MM or a year only in the format YYYY. **Example:** `2021-04`
records[]. employment_history	**array of objects** A list of employment history items for the person, ordered by start date (most recent first).
records[]. employment_history[]. company	**string** The name of the company the person worked for **Example:** `Other Corp`
records[]. employment_history[]. title	**string** The job title of the person **Example:** `Systems Engineer`
records[]. employment_history[]. date_start	**string (date)** The start date of the person's employment. This can be either a full date in the format YYYY-MM-DD or a year/month in the format YYYY-MM or a year only in the format YYYY **Example:** `2015-01-01`
records[]. employment_history[]. date_end	**string (date)** The end date of the person's employment. This can be either a full date in the format YYYY-MM-DD or a year/month in the format YYYY-MM or a year only in the format YYYY **Example:** `2020-01-01`

Field	Description
records[]. employment_history[]. industry	**string** The industry of the company the person worked for **Example:** `retail`
records[]. employment_history[]. company_size	**string** The approximate size of the company **Example:** `201-500`
records[]. employment_history[]. company_website	**string** The website of the company **Example:** `othercorp.com.au`
records[]. employment_history[]. is_primary	**boolean** Whether this is the person's current primary role **Example:** `false`
records[]. education	**array of objects** Education details of the person, ordered by start date (most recent first).
records[]. education[]. school	**string** The name of the institution **Example:** `University of Sydney`
records[]. education[]. type	**string** The type of institution **Example:** `University`
records[]. education[]. url	**string** The URL of the institution **Example:** `https://www.sydney.edu.au/`
records[]. education[]. studies	**array of strings** Studies undertaken at the institution

Field	Description
records[]. education[]. date_start	**string (date)** The start date of the person's studies. This can be either a full date in the format YYYY-MM-DD or a year/month in the format YYYY-MM or a year only in the format YYYY **Example:** `2015-01-01`
records[]. education[]. date_end	**string (date)** The end date of the person's studies. This can be either a full date in the format YYYY-MM-DD or a year/month in the format YYYY-MM or a year only in the format YYYY **Example:** `2020-01-01`
records[]. education[]. location	**string** The geographic location of the institution **Example:** `sydney, new south wales, australia`
records[]. education[]. domain	**string** The web domain of the institution **Example:** `sydney.edu.au`
records[]. interests	**array of strings** A list of interests of the person
records[]. skills	**array of strings** A list of skills of the person
records[]. socials	**array of objects** A list of social media profiles of the person
records[]. socials[]. network	**string** The type of social media profile **Example:** `LinkedIn`
records[]. socials[]. url	**string** The URL of the social media profile **Example:** `https://www.linkedin.com/in/johnsmith/`

Field	Description
records[]. socials[]. username	**string** The username of the social media profile **Example:** `johnsmith`
records[]. socials[]. id	**string** The ID of the social media profile **Example:** `1234567890`
records[]. record_date	**string (date)** The date this record was last updated, in the format YYYY-MM-DD. **Example:** `2025-03-19`

Sample response

```
{
    "message": "Ok",
    "api_reference": "738d9a89-b6fe-4fc1-96be-1389b2a506a2"
}
```

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

Social Contact Expand

POST `/social_contact_expand`

The `social_contact_expand` method takes a batch of contact identifiers (phone numbers, email addresses, or social media profile URLs) and, for each one, surfaces additional contact phones and emails associated with that person.

For each input the endpoint runs two stages:

1. **`social_search`** - a social media check is performed against the supplied identifier. Any phones or emails returned by the social check are included in this block.
2. **`universe_search`** - one universe lookup is performed for each phone or email known for the contact: the original input itself (when it is a phone or email), plus any phones or emails returned by the social check. Each lookup that finds related persons in our universe database produces its own `universe_search` block listing the additional phones and emails discovered for those persons.

Result blocks - "no match" means the block is omitted

Each lookup is treated independently and **only blocks that actually returned phones or emails appear in the response**. There is no fixed number or order guarantee for the blocks - the caller should iterate over `records` and inspect each block's `type`. Specifically:

- If the social check found nothing but a universe lookup did, only the matching `universe_search` block(s) are returned (no `social_search` block).

- If the social check found phones or emails but none of the universe lookups (including the one for the original input, if applicable) found anything, only the `social_search` block is returned (no `universe_search` blocks).
- If both stages produced data, the response contains the `social_search` block and one `universe_search` block per universe lookup that found data.
- If nothing matched anywhere, `records` is an empty array and `message` is `No match`.

The per-item `message` field summarises this overall outcome (`Ok` if any block has data, `No match` otherwise). Suppressed phones and emails are excluded from results.

Supported inputs

Each request item's `search` value is auto-detected as one of the following:

- **Email address** - any RFC-compliant email address.
- **Phone number** - in either E.164 format (e.g. `+61412345678`) or Australian local 0NSN format (e.g. `0412345678`).
- **Social media profile URL** - same supported networks and URL formats as the Social Check endpoint. Profile URLs cannot be searched in the universe database, so a profile-only input will only yield universe matches when the social check returns associated phones or emails.

Invalid inputs (e.g. malformed values) return a per-item `Error: Invalid search value` message in the response and are not billed.

Sandbox environment data

The sandbox uses the same simulated dataset as the Social Check endpoint. Universe searches in the sandbox run against the sample Pango database.

Request body

Batch of search records to expand.

Field	Description
`requests` REQUIRED	**array of objects [1..50 items]** An array of search records to expand. Maximum 50 records per call.
`requests[].` `search_id`	**string [max 50 characters]** An optional caller-provided identifier for this search. When provided, it is echoed back in the corresponding result item so the caller can match results to inputs. Maximum 50 characters. **Example:** `abc-1`

`requests[].`
`search`
REQUIRED
	string [max 200 characters]
	The contact identifier to expand. May be an email address, a phone number (E.164 or 0NSN format), or a social media profile URL. Maximum 200 characters.
	Example: `john@example.com`

Sample request

```
{
    "requests": [
        {
            "search_id": "abc-1",
            "search": "john@example.com"
        }
    ]
}
```

Responses

200 **Result of the contact expansion.** (application/json)

Field	Description
`message`	**string** A message indicating the result of the request. Will be `Ok` for any successful call. **Example:** `Ok`
`api_reference`	**string (uuid)** **Example:** `738d9a89-b6fe-4fc1-96be-1389b2a506a2`
`results`	**array of objects** Per-input result items, returned in the same order as the request.
`results[].search_id`	**string or null** The caller-provided `search_id` (if any) for this input. **Example:** `abc-1`
`results[].search`	**string** The original `search` value as supplied in the request. **Example:** `john@example.com`
`results[].message`	**string** The per-input result. - `Ok` - The social search or at least one universe search returned phones/emails. - `No match` - No phones or emails were discovered for this input. - `Error: Invalid search value` - The supplied `search` value could not be recognised as an email, phone or social profile URL. **Example:** `Ok`

Field	Description
results[]. records	**array of objects** The discovered records for this input. Each item is either a `social_search` block (the social media check on the original input) or a `universe_search` block (a universe lookup on a phone or email). Only blocks that actually returned phones or emails are included - blocks for searches that produced no match are omitted entirely (so a response may contain only `universe_search` blocks if the social check produced no match, or only a `social_search` block if no universe lookup found anything). When `message` is `No match`, this array is empty.
results[]. records[]. type	**string** The type of search this block represents. - `social_search` - results from the social media check. - `universe_search` - results from a universe lookup on a phone or email. **Enum:** `social_search`, `universe_search` **Example:** `social_search`
results[]. records[]. search_for	**string** The phone, email or profile URL this block was searched for. For `social_search` this is the original input. For `universe_search` this is the phone or email that was looked up in the universe. **Example:** `john@example.com`
results[]. records[]. emails	**array of strings** Email addresses discovered for this search. Lowercased. The original input email (if any) and the universe key being searched are excluded.
results[]. records[]. phones	**array of strings** Phone numbers discovered for this search, in E.164 format. The original input phone (if any) and the universe key being searched are excluded.

Sample response


```
{
    "message": "Ok",
    "api_reference": "738d9a89-b6fe-4fc1-96be-1389b2a506a2",
    "results": [
        {
            "search_id": "abc-1",
            "search": "john@example.com",
            "message": "Ok",
            "records": [
                {
                    "type": "social_search",
                    "search_for": "john@example.com",
```

```
                    "emails": [
                        "john.doe@example.com"
                    ],
                    "phones": [
                        "+61412345001"
                    ]
                }
            ]
        }
    ]
}
```

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

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).

Company House Search

POST `/company_house_search`

Searches the UK Companies House register for companies matching the provided search term and returns a list of results.

Search Behaviour

The search is intelligent and will detect if the search term matches a company number format:

- **Company number search**: If the search term matches the 8-character company number format (e.g., `12345678` or `SC123456`), a direct lookup is performed.
- **Name search**: Otherwise, a fuzzy name search is performed against registered company names.

Company Number Formats

Prefix	Jurisdiction	Example
(none)	England & Wales	12345678
SC	Scotland	SC123456
NI	Northern Ireland	NI012345
FC	Foreign Company	FC012345
OC	LLP (England & Wales)	OC123456

Sandbox Environment

The sandbox dataset is deterministic and does not use live data. Use one of the following sample companies to receive predictable matches:

Company Name	Company Number	Status	Type
TEST COMPANY LTD	12345678	Active	ltd
SAMPLE HOLDINGS PLC	87654321	Active	plc
DORMANT SERVICES LTD	11223344	Dormant	ltd
DISSOLVED EXAMPLE LTD	99887766	Dissolved	ltd
SCOTTISH ENTERPRISE SC	SC654321	Active	ltd
DELAYED REPORT LTD	55667788	Active	ltd

You can search by company name (e.g., `TEST COMPANY`) or by company number (e.g., `12345678`).

Request body

The search criteria for finding companies.

Field	Description
`company` REQUIRED	**string [max 255 characters]** The company name or company number to search for. Maximum 255 characters. **Example:** `Test Company`
`is_operational`	**boolean** When `true`, filters results to only include companies with an active/operational status. **Default:** `false` **Example:** `false`
`max_results`	**integer [1..200]** Maximum number of results to return. Value must be between 1 and 200. **Default:** `200` **Example:** `50`

Sample request

```
{
    "company": "Test Company",
    "is_operational": false,
    "max_results": 50
}
```

Responses

200 **Successful response with search results.** (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`
`function`	**string** The API function that handled the request. **Example:** `company_house_search`

Field	Description
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`records`	**array of objects** Array of matching company records.
`records[].` `company_number`	**string** The unique 8-character company number. **Example:** `12345678`
`records[].` `company_name`	**string** The registered name of the company. **Example:** `TEST COMPANY LTD`
`records[].` `company_status`	**string** The current status of the company (e.g., active, dissolved, liquidation). **Example:** `active`
`records[].` `company_type`	**string** The type of company (e.g., ltd, plc, llp). **Example:** `ltd`
`records[].` `date_of_creation`	**string (date)** The date the company was incorporated. **Example:** `2020-01-15`
`records[].` `registered_office_address`	**object** The registered office address of the company.

`records[].`
`registered_office_address.`
`address_line_1`
	string
	
	Example: `123 High Street`

`records[].`
`registered_office_address.`
`address_line_2`
	string
	
	Example: `Floor 2`

`records[].`
`registered_office_address.`
`locality`
	string
	
	Example: `London`

Field	Description
records[].registered_office_address.region	**string** **Example:** `Greater London`
records[].registered_office_address.postal_code	**string** **Example:** `SW1A 1AA`
records[].registered_office_address.country	**string** **Example:** `United Kingdom`
records_returned	**integer** The number of records returned in this response. **Example:** `5`
total_records_available	**integer** The total number of matching records available. **Example:** `150`

Sample response


```
{
    "message": "Ok",
    "function": "company_house_search",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "records": [
        {
            "company_number": "12345678",
            "company_name": "TEST COMPANY LTD",
            "company_status": "active",
            "company_type": "ltd",
            "date_of_creation": "2020-01-15",
            "registered_office_address": {
                "address_line_1": "123 High Street",
                "address_line_2": "Floor 2",
                "locality": "London",
                "region": "Greater London",
                "postal_code": "SW1A 1AA",
                "country": "United Kingdom"
            }
        }
    ],
    "records_returned": 5,
    "total_records_available": 150
}
```


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

Company House Report

POST `/company_house_report`

Requests a comprehensive UK Companies House report for the specified company. The response confirms the job was queued and returns an `api_reference`. Poll `GET /company_house_report/{api_reference}` until the report payload is ready.

Report Contents

The company report includes:

- Company registration details
- Registered office address
- Current and previous company names
- Officers (directors, secretaries)
- Persons with significant control (PSCs)
- Filing history summary
- Accounts information
- Charges (if any)

Sandbox Environment

The sandbox dataset is deterministic and does not use live data. Use one of the following sample company numbers to receive predictable reports:

Company Name	Company Number	Status	Notes
TEST COMPANY LTD	12345678	Active	Standard company with 2 officers
SAMPLE HOLDINGS PLC	87654321	Active	PLC with multiple officers including corporate secretary
DORMANT SERVICES LTD	11223344	Dormant	Dormant company
DISSOLVED EXAMPLE LTD	99887766	Dissolved	Dissolved company with previous names
SCOTTISH ENTERPRISE SC	SC654321	Active	Scottish company
DELAYED REPORT LTD	55667788	Active	Returns 202 on first poll (simulates async processing)

Request body

The company number to generate a report for.

Field	Description
`company_number` REQUIRED	**string [8..8 characters]** The unique 8-character UK Companies House company number. **Example:** `12345678`

Sample request

```
{
    "company_number": "12345678"
}
```

Responses

200 **Report request accepted.** (application/json)

Field	Description
`message`	**string** Always `Ok` on success. **Example:** `Ok`
`function`	**string** The API function that handled the request. **Example:** `company_house_report`
`api_reference`	**string (uuid)** Audit reference for this report request. Poll `/company_house_report/{api_reference}` to retrieve results. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Sample response

```
{
    "message": "Ok",
    "function": "company_house_report",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95"
}
```

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

Retrieve Company House Report

GET `/company_house_report/{api_reference}`

Polls the status of a UK Companies House report that was previously queued via `POST /company_house_report`. Provide the `api_reference` returned from the original request to determine whether the report is ready, download the JSON payload, or stream a PDF copy. This retrieval call is not billable and can be repeated until the report is ready.

Response Behaviour

- While the report is still compiling, the API responds with HTTP `202`. Keep polling with an exponential backoff.
- Once ready, the endpoint returns HTTP `200` with the full JSON report.

- Append `?pdf=true` to stream a formatted PDF copy of the same report. When `pdf=true`, the response body is a PDF (`application/pdf`) instead of JSON.
- The service polls upstream data for up to 30 seconds; if no report is available at that point you will receive a `202` response. Continue polling with the same `api_reference`.

Sandbox Usage

The sandbox dataset is deterministic and does not use live data. First request a report for one of the sandbox company numbers via `POST /company_house_report`, then poll this endpoint with the returned `api_reference`.

Note: The company `55667788` (DELAYED REPORT LTD) will return HTTP `202` on the first poll to simulate asynchronous processing. Subsequent polls will return the report.

Path parameters

Field	Description
`api_reference` REQUIRED	string (uuid) The API reference (UUID) returned when the report was queued via `POST /company_house_report`. Example: `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Query parameters

Field	Description
`pdf`	boolean When set to `true`, returns the report as a PDF document instead of JSON. Omit (or send `false`) to receive the JSON payload. Default: `false`

Responses

200 **Report is ready. Returns JSON by default or a PDF when `pdf=true`.** (application/json, application/pdf)

Field	Description
`message`	string Always `Ok` when the report payload is returned. Example: `Ok`
`function`	string Name of the API function that handled the request. Example: `company_house_report_show`

Field	Description
`api_reference`	**string (uuid)** Audit reference for **this** retrieval call (not the original queue reference). **Example:** `7cbb094c-0f27-42f4-9f36-9cf0b2f6d5ff`
`report`	**object** Full UK Companies House company report. The top-level fields come from the Companies House company profile endpoint. Additional sections (`officers` , `filing_history` , `persons_with_significant_control`) are fetched from their respective Companies House endpoints and included when available.
`report.` `company_name`	**string** The registered name of the company. **Example:** `TEST COMPANY LTD`
`report.` `company_number`	**string** The Companies House company number. **Example:** `12345678`
`report.` `company_status`	**string** The current status of the company (e.g. `active` , `dissolved` , `liquidation` , `receivership` , `administration` , `converted-closed`). **Example:** `active`
`report.` `company_status_detail`	**string** Additional detail about the company status, if any.
`report.` `type`	**string** The type of company (e.g. `ltd` , `plc` , `llp` , `private-unlimited` , etc.). **Example:** `ltd`
`report.` `subtype`	**string** The company subtype, if applicable.
`report.` `date_of_creation`	**string (date)** The date the company was created/incorporated. **Example:** `2020-01-15`

Field	Description
report. date_of_cessation	**string (date)** The date the company was dissolved or ceased. Present only for dissolved/ceased companies.
report. jurisdiction	**string** The jurisdiction the company is registered in (e.g. `england-wales`, `scotland`, `northern-ireland`). **Example:** `england-wales`
report. registered_office_address	**object** The company's registered office address.
report. registered_office_address. premises	**string**
report. registered_office_address. address_line_1	**string** **Example:** `123 Test Street`
report. registered_office_address. address_line_2	**string** **Example:** `Test District`
report. registered_office_address. care_of	**string**
report. registered_office_address. locality	**string** **Example:** `London`
report. registered_office_address. region	**string**
report. registered_office_address. country	**string** **Example:** `United Kingdom`
report. registered_office_address. po_box	**string**
report. registered_office_address. postal_code	**string** **Example:** `EC1A 1BB`
report. registered_office_is_in_dispute	**boolean** Whether the registered office address is in dispute.

Field	Description
report. accounts	**object (dynamic)** Accounting information including reference dates and the last/next filing periods.
report. annual_return	**object (dynamic)** Annual return information (pre-2016 companies).
report. can_file	**boolean** Whether the company can currently file with Companies House. **Example:** `true`
report. confirmation_statement	**object (dynamic)** Confirmation statement due dates and status.
report. corporate_annotation	**string** Corporate annotation text, if any.
report. external_registration_number	**string** The registration number of an external company.
report. foreign_company_details	**object (dynamic)** Details of a foreign company, if applicable.
report. branch_company_details	**object (dynamic)** Details of a branch company, if applicable.
report. partial_data_available	**boolean** Indicates partial data availability.
report. previous_company_names	**array of objects** List of previous names the company has used.
report. previous_company_names[]. name	**string**
report. previous_company_names[]. ceased_on	**string (date)**
report. previous_company_names[]. effective_from	**string (date)**

Field	Description
report. service_address	**object (dynamic)** The service address, if different from the registered office.
report. sic_codes	**array of objects** Standard Industrial Classification codes expanded with descriptions.
report. sic_codes[]. code	**string** The SIC code. **Example:** `62020`
report. sic_codes[]. description	**string** Human-readable description of the SIC code. **Example:** `Information technology consultancy activities`
report. super_secure_managing_officer_count	**integer** Count of super-secure managing officers, if any.
report. undeliverable_registered_office_address	**boolean** Whether the registered office address has been flagged as undeliverable.
report. report_date	**string (date-time)** ISO 8601 UTC timestamp indicating when the report was generated. **Example:** `2026-02-03T10:00:00Z`
report. officers	**array of objects** Array of officers (directors, secretaries, etc.) associated with the company. Active officers (no `resigned_on`) are listed first, sorted by appointment date descending. Resigned officers follow, sorted by resignation date ascending. Only present when the company has an officers record at Companies House.
report. officers[]. officer_id	**string** The unique officer identifier within Companies House. **Example:** `ABC123DEF456`
report. officers[]. name	**string** The officer's name in `SURNAME, Forenames` format. **Example:** `SMITH, John David`

Field	Description
report. officers[]. officer_role	**string** The role held (e.g. `director`, `secretary`, `corporate-director`, `corporate-secretary`, `llp-member`, `llp-designated-member`). **Example:** `director`
report. officers[]. appointed_on	**string (date)** The date the officer was appointed. **Example:** `2020-01-15`
report. officers[]. resigned_on	**string (date)** The date the officer resigned. Absent for active officers.
report. officers[]. appointed_before	**string (date)** Present when `is_pre_1992_appointment` is `true`.
report. officers[]. is_pre_1992_appointment	**boolean** Whether the appointment pre-dates 1992 records.
report. officers[]. date_of_birth	**string** Date of birth truncated to year and month (`YYYY-MM`). **Example:** `1975-06`
report. officers[]. nationality	**string** The officer's nationality. **Example:** `British`
report. officers[]. country_of_residence	**string** The officer's country of residence. **Example:** `England`
report. officers[]. occupation	**string** The officer's occupation.
report. officers[]. person_number	**string** Internal Companies House person number.
report. officers[]. address	**object (dynamic)** The correspondence address of the officer.

Field	Description
report. officers[]. former_names	**array of objects** Previous names for the officer.
report. officers[]. former_names[]. forenames	**string**
report. officers[]. former_names[]. surname	**string**
report. officers[]. identification	**object** Identification details for corporate officers.
report. officers[]. identification. identification_type	**string** **Example:** `uk-limited-company`
report. officers[]. identification. legal_authority	**string**
report. officers[]. identification. legal_form	**string**
report. officers[]. identification. place_registered	**string**
report. officers[]. identification. registration_number	**string** **Example:** `12345678`
report. officers[]. identity_verification_details	**object (dynamic)** Identity verification information, if available.
report. officers[]. contact_details	**object (dynamic)** Contact details for corporate managing officers.
report. officers[]. principal_office_address	**object (dynamic)** Principal office address for corporate managing officers.
report. officers[]. responsibilities	**string** Responsibilities of a managing officer.

Field	Description
report. filing_history	**object** The company's filing history. Only present when the company has filings at Companies House. Up to 200 most recent filings are returned.
report. filing_history. total_files	**integer** The total number of filings for this company. **Example:** `25`
report. filing_history. returned_files	**integer** The number of filings included in this response (capped at 200). **Example:** `25`
report. filing_history. files	**array of objects** Array of individual filing records.
report. filing_history. files[]. transaction_id	**string** Unique transaction identifier for this filing.
report. filing_history. files[]. document_id	**string** Document identifier that can be used with the Company House Document endpoint to download the filing as a PDF.
report. filing_history. files[]. date	**string (date)** The date of the filing.
report. filing_history. files[]. type	**string** The type code of the filing (e.g. `AA`, `CS01`, `AP01`, `TM01`).
report. filing_history. files[]. category	**string** The category of filing (e.g. `accounts`, `confirmation-statement`, `officers`, `capital`, `mortgage`, `address`, `resolution`).
report. filing_history. files[]. subcategory	**string** Sub-category of the filing (e.g. `appointments`, `termination`, `change`, `create`, `satisfy`).
report. filing_history. files[]. description	**string** Human-readable description of the filing.

Field	Description
report. filing_history. files[]. pages	**integer** Number of pages in the document.
report. filing_history. files[]. barcode	**string** Barcode identifier for the filing.
report. filing_history. files[]. paper_filed	**boolean** Whether the filing was submitted on paper.
report. filing_history. files[]. annotations	**array of objects** Annotations associated with the filing.
report. filing_history. files[]. associated_filings	**array of objects** Related filings associated with this filing.
report. filing_history. files[]. resolutions	**array of objects** Resolution details, if applicable.
report. persons_with_significant_control	**array of objects** Array of persons with significant control (PSCs) over the company. Only present when the company has PSC records at Companies House.
report. persons_with_significant_control[]. kind	**string** The type of PSC (e.g. `individual-person-with-significant-control`, `corporate-entity-person-with-significant-control`, `legal-person-person-with-significant-control`). **Example:** `individual-person-with-significant-control`
report. persons_with_significant_control[]. name	**string** The name of the individual or entity. **Example:** `Mr John David Smith`
report. persons_with_significant_control[]. address	**object (dynamic)** The correspondence address of the PSC.
report. persons_with_significant_control[]. date_of_birth	**string** Date of birth truncated to year and month (`YYYY-MM`). Only for individuals.

Field	Description
report. persons_with_significant_control[]. country_of_residence	**string** Country of residence. Only for individuals.
report. persons_with_significant_control[]. nationality	**string** Nationality of the PSC.
report. persons_with_significant_control[]. natures_of_control	**array of strings** List of natures of control held (e.g. `ownership-of-shares-75-to-100-percent`, `voting-rights-75-to-100-percent`, `right-to-appoint-and-remove-directors`).
report. persons_with_significant_control[]. identification	**object (dynamic)** Identification details for corporate entity PSCs.
report. persons_with_significant_control[]. name_elements	**object (dynamic)** Broken-down name components for individual PSCs.
report. persons_with_significant_control[]. notified_on	**string (date)** The date the PSC was notified.
report. persons_with_significant_control[]. ceased	**boolean** Whether the PSC has ceased.
report. persons_with_significant_control[]. ceased_on	**string (date)** The date the PSC ceased, if applicable.
report. persons_with_significant_control[]. is_sanctioned	**boolean** Whether the PSC is sanctioned.
report. persons_with_significant_control[]. description	**string** Description of the PSC entry.
report. persons_with_significant_control[]. principal_office_address	**object (dynamic)** Principal office address for legal person/corporate entity PSCs.

Sample response


```
{
    "message": "Ok",
    "function": "company_house_report_show",
    "api_reference": "7cbb094c-0f27-42f4-9f36-9cf0b2f6d5ff",
    "report": {
        "company_name": "TEST COMPANY LTD",
```

```
        "company_number": "12345678",
        "company_status": "active",
        "type": "ltd",
        "date_of_creation": "2020-01-15",
        "jurisdiction": "england-wales",
        "can_file": true,
        "report_date": "2026-02-03T10:00:00Z",
        "registered_office_address": {
            "address_line_1": "123 Test Street",
            "address_line_2": "Test District",
            "locality": "London",
            "postal_code": "EC1A 1BB",
            "country": "United Kingdom"
        },
        "accounts": {
            "overdue": false,
            "next_due": "2026-10-15",
            "last_accounts": {
                "type": "full",
                "made_up_to": "2025-01-15"
            },
            "next_accounts": {
                "due_on": "2026-10-15",
                "overdue": false
            },
            "accounting_reference_date": {
                "day": "15",
                "month": "01"
            }
        },
        "confirmation_statement": {
            "overdue": false,
            "next_due": "2026-07-15",
            "last_made_up_to": "2025-06-15",
            "next_made_up_to": "2026-06-15"
        },
        "sic_codes": [
            {
                "code": "62020",
                "description": "Information technology consultancy activities"
            },
            {
                "code": "62090",
                "description": "Other information technology and computer service activities"
            }
        ],
        "officers": [
            {
                "name": "SMITH, John David",
                "officer_id": "ABC123DEF456",
                "officer_role": "director",
                "appointed_on": "2020-01-15",
                "date_of_birth": "1975-06",
                "nationality": "British",
                "country_of_residence": "England",
                "address": {
                    "premises": "123",
                    "address_line_1": "Test Street",
                    "address_line_2": "Test District",
                    "locality": "London",
                    "postal_code": "EC1A 1BB",
```

```
                    "country": "United Kingdom"
                }
            },
            {
                "name": "DOE, Jane Elizabeth",
                "officer_id": "XYZ789GHI012",
                "officer_role": "secretary",
                "appointed_on": "2020-01-15",
                "date_of_birth": "1980-03",
                "nationality": "British",
                "country_of_residence": "England",
                "address": {
                    "premises": "456",
                    "address_line_1": "Sample Road",
                    "locality": "London",
                    "postal_code": "EC1A 2CC",
                    "country": "United Kingdom"
                }
            }
        ],
        "filing_history": {
            "total_files": 5,
            "returned_files": 5,
            "files": [
                {
                    "date": "2025-10-01",
                    "type": "AA",
                    "pages": 15,
                    "barcode": "SANDBOX001",
                    "category": "accounts",
                    "description": "accounts-with-accounts-type-full",
                    "document_id": "sandbox-document-001",
                    "transaction_id": "SANDBOX00001"
                },
                {
                    "date": "2025-06-15",
                    "type": "CS01",
                    "pages": 3,
                    "barcode": "SANDBOX002",
                    "category": "confirmation-statement",
                    "description": "confirmation-statement-with-updates",
                    "document_id": "sandbox-document-002",
                    "transaction_id": "SANDBOX00002"
                }
            ]
        },
        "persons_with_significant_control": [
            {
                "kind": "individual-person-with-significant-control",
                "name": "Mr John David Smith",
                "natures_of_control": [
                    "ownership-of-shares-75-to-100-percent",
                    "voting-rights-75-to-100-percent"
                ],
                "notified_on": "2020-01-15",
                "country_of_residence": "England",
                "date_of_birth": "1975-06",
                "nationality": "British"
            }
        ]
    }
```

```
}
```

Also available as a PDF (application/pdf)

202 **Report is still being prepared (returned when no data is available after the 30-second polling window).**
(application/json)

Field	Description
`message`	**string** Status message indicating the report is not ready yet. **Example:** `Extract is still being processed. Please try again later.`

Sample response

```
{
    "message": "Extract is still being processed. Please try again later."
}
```

404 **No report was found for the provided `api_reference`, or it belongs to another account/environment.**
(application/json)

Field	Description
`message`	**string** Error message describing that the record could not be located. **Example:** `Not Found.`

Sample response

```
{
    "message": "Not Found."
}
```

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

Company House Officer Search

POST `/company_house_officer_search`

Searches the UK Companies House register for officers (directors, secretaries, and other officials) matching the provided name and returns a list of results.

Search Behaviour

The search performs a fuzzy match against officer names registered with Companies House. Results include current and former officers across all registered companies.

An optional `date_of_birth` filter can be provided to narrow results. When supplied, only officers whose date of birth matches the given year and month are returned. The day component is accepted for convenience but is not used for matching, as Companies House only provides month and year of birth for most officers.

Sandbox Environment

The sandbox dataset is deterministic and does not use live data. Use one of the following sample officer names to receive predictable matches:

Name	Officer ID	Type	DOB (Y-M)	Appointments
John David SMITH	ABC123DEF456	Natural	1975-06	3
Jane Elizabeth DOE	XYZ789GHI012	Natural	1980-03	1
Robert James JOHNSON	JKL345MNO678	Natural	1968-11	5
Christopher Mark TAYLOR	DEL789AYE012	Natural	1990-05	1 (delayed report)
CORPORATE SECRETARIES LIMITED	VWX567YZA890	Corporate	N/A	25
David DISQUALIFIED	DIS123QUA456	Natural (Disqualified)	1970-01	0

Search using names like `John Smith`, `JOHNSON`, or `CORPORATE SECRETARIES`.

Request body

The search criteria for finding officers.

Field	Description
`name` REQUIRED	**string [max 255 characters]** The name of the officer to search for. Maximum 255 characters. **Example:** `John Smith`
`date_of_birth`	**string (date) or null** Optional date of birth filter in YYYY-MM-DD format. When provided, only officers whose date of birth matches the year and month will be returned. The day component is accepted for convenience but is not used for matching, as Companies House only provides month and year of birth for most officers. **Example:** `1975-06-15`
`max_results`	**integer [1..100]** Maximum number of results to return. Value must be between 1 and 100. **Default:** `50` **Example:** `50`

Sample request

```
{
    "name": "John Smith",
    "date_of_birth": "1975-06-15",
    "max_results": 50
}
```

Responses

200 **Successful response with search results.** (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`
`function`	**string** The API function that handled the request. **Example:** `company_house_officer_search`
`api_reference`	**string (uuid)** A unique identifier for this request. This can be used to track the request in the logs. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`records`	**array of objects** Array of matching officer records.
`records[].officer_id`	**string** The unique identifier for the officer. Use this ID when requesting an officer report. **Example:** `ABC123DEF456`
`records[].title`	**string** The full name of the officer as it appears in the register. **Example:** `JOHN SMITH`
`records[].date_of_birth`	**string** The officer's date of birth (month and year only for privacy), formatted as YYYY-MM. **Example:** `1975-06`
`records[].address`	**object** The correspondence address of the officer.

Field	Description
records[]. address. address_line_1	**string** **Example:** `456 Oak Avenue`
records[]. address. locality	**string** **Example:** `Manchester`
records[]. address. postal_code	**string** **Example:** `M1 1AA`
records[]. address. country	**string** **Example:** `United Kingdom`
records[]. address_snippet	**string** A single-line summary of the officer's address. **Example:** `456 Oak Avenue, Manchester, M1 1AA`
records[]. appointment_count	**integer** Number of current company appointments. **Example:** `3`
records[]. description	**string** A description of the officer's roles. **Example:** `Total number of appointments 3`
records_returned	**integer** The number of records returned in this response. **Example:** `5`
more_results	**boolean** Indicates whether more results are available beyond the current page. **Example:** `true`

Sample response


```
{
    "message": "Ok",
    "function": "company_house_officer_search",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "records": [
        {
            "officer_id": "ABC123DEF456",
```

```
            "title": "JOHN SMITH",
            "date_of_birth": "1975-06",
            "address": {
                "address_line_1": "456 Oak Avenue",
                "locality": "Manchester",
                "postal_code": "M1 1AA",
                "country": "United Kingdom"
            },
            "address_snippet": "456 Oak Avenue, Manchester, M1 1AA",
            "appointment_count": 3,
            "description": "Total number of appointments 3"
        }
    ],
    "records_returned": 5,
    "more_results": true
}
```


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

Company House Officer Report

POST `/company_house_officer_report`

Requests a comprehensive UK Companies House officer report for the specified officer. The response confirms the job was queued and returns an `api_reference`. Poll `GET /company_house_officer_report/{api_reference}` until the report payload is ready.

Report Contents

The officer report includes:

- Personal details (name, date of birth - month/year only)
- Current appointments (directorships, secretaryships)
- Former appointments
- Disqualification records (if `disqualified=true`)
- Correspondence address

Officer Types

The `type` parameter filters results by officer type:

- `natural` - Individual persons (default)
- `corporate` - Corporate officers (companies acting as directors/secretaries)

Disqualified Officers

Set `disqualified=true` to retrieve disqualification information. This searches the UK register of disqualified directors maintained by Companies House.

Sandbox Environment

The sandbox dataset is deterministic and does not use live data. Use one of the following sample officer IDs to receive predictable reports:

Name	Officer ID	Type	Notes
John David SMITH	ABC123DEF456	Natural	Director with 3 appointments
Jane Elizabeth DOE	XYZ789GHI012	Natural	Secretary with 1 appointment
Robert James JOHNSON	JKL345MNO678	Natural	Director with 5 appointments
Christopher Mark TAYLOR	DEL789AYE012	Natural	Returns 202 on first poll
CORPORATE SECRETARIES LIMITED	VWX567YZA890	Corporate	Corporate secretary
David DISQUALIFIED	DIS123QUA456	Natural	Disqualified director (use `disqualified=true`)

Request body

The officer ID and optional parameters for the report.

Field	Description
`officer_id` REQUIRED	**string** The unique officer ID obtained from an officer search. **Example:** `ABC123DEF456`
`disqualified`	**boolean or null** When `true`, searches the disqualified officers register instead of the standard officers register. **Example:** `false`
`type`	**string or null** The type of officer to search for. Only relevant when `disqualified` is `true`. **Enum:** `natural`, `corporate` **Example:** `natural`

Sample request

```
{
    "officer_id": "ABC123DEF456",
    "disqualified": false,
    "type": "natural"
}
```

Responses

200 **Report request accepted.** (application/json)

Field	Description

`message`	**string** Always `Ok` on success. **Example:** `Ok`
`function`	**string** The API function that handled the request. **Example:** `company_house_officer_report`
`api_reference`	**string (uuid)** Audit reference for this report request. Poll `/company_house_officer_report/{api_reference}` to retrieve results. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Sample response

```
{
    "message": "Ok",
    "function": "company_house_officer_report",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95"
}
```

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

Retrieve Company House Officer Report

GET `/company_house_officer_report/{api_reference}`

Polls the status of a UK Companies House officer report that was previously queued via `POST /company_house_officer_report`. Provide the `api_reference` returned from the original request to determine whether the report is ready, download the JSON payload, or stream a PDF copy. This retrieval call is not billable and can be repeated until the report is ready.

Response Behaviour

- While the report is still compiling, the API responds with HTTP `202`. Keep polling with an exponential backoff.
- Once ready, the endpoint returns HTTP `200` with the full JSON report.
- Append `?pdf=true` to stream a formatted PDF copy of the same report. When `pdf=true`, the response body is a PDF (`application/pdf`) instead of JSON.
- The service polls upstream data for up to 30 seconds; if no report is available at that point you will receive a `202` response. Continue polling with the same `api_reference`.

Sandbox Environment

The sandbox dataset is deterministic and does not use live data. First request a report for one of the sandbox officer IDs via `POST /company_house_officer_report`, then poll this endpoint with the returned `api_reference`.

Name	Officer ID	Type	Notes
John David SMITH	ABC123DEF456	Natural	Director with 3 appointments
Jane Elizabeth DOE	XYZ789GHI012	Natural	Secretary with 1 appointment
Robert James JOHNSON	JKL345MNO678	Natural	Director with 5 appointments
Christopher Mark TAYLOR	DEL789AYE012	Natural	Returns 202 on first poll
CORPORATE SECRETARIES LIMITED	VWX567YZA890	Corporate	Corporate secretary
David DISQUALIFIED	DIS123QUA456	Natural	Disqualified director (use `disqualified=true`)

Note: The officer `DEL789AYE012` (Christopher Mark TAYLOR) will return HTTP `202` on the first poll to simulate asynchronous processing. Subsequent polls will return the report.

Path parameters

Field	Description
`api_reference` REQUIRED	string (uuid) The API reference (UUID) returned when the report was queued via `POST /company_house_officer_report`. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`

Query parameters

Field	Description
`pdf`	boolean When set to `true`, returns the report as a PDF document instead of JSON. Omit (or send `false`) to receive the JSON payload. **Default:** `false`

Responses

200 **Report is ready. Returns JSON by default or a PDF when `pdf=true`.** (application/json, application/pdf)

Field	Description
`message`	string Always `Ok` when the report payload is returned. **Example:** `Ok`

Field	Description
`function`	**string** Name of the API function that handled the request. **Example:** `company_house_officer_show`
`api_reference`	**string (uuid)** Audit reference for **this** retrieval call (not the original queue reference). **Example:** `7cbb094c-0f27-42f4-9f36-9cf0b2f6d5ff`
`report`	**object** Full UK Companies House officer report payload sourced from the Companies House `GET /officers/{officer_id}/appointments` endpoint. For non-disqualified officers the top-level object is an `appointmentList` containing the officer's personal details and an `items` array of every appointment (role) they hold or have held across different companies. Disqualified officer reports include a `disqualifications` array instead.
`report.` `officer_id`	**string** The unique officer identifier within Companies House. **Example:** `XYZ789GHI012`
`report.` `name`	**string** The full name of the officer as registered with Companies House. **Example:** `Jane Elizabeth DOE`
`report.` `date_of_birth`	**string** The officer's date of birth truncated to year and month (`YYYY-MM`). Omitted when Companies House does not provide it. **Example:** `1980-03`
`report.` `etag`	**string** The ETag of the upstream Companies House resource. **Example:** `sandbox-etag-002`
`report.` `kind`	**string** The kind of record. Possible values include: - `personal-appointment` - standard officer appointments - `personal-disqualification` - disqualified officer record **Example:** `personal-appointment`

Field	Description
report. type	**string** The type of officer. Possible values: - `natural` - a human individual - `corporate` - a corporate body acting as an officer **Enum:** `natural`, `corporate` **Example:** `natural`
report. is_corporate_officer	**boolean** Indicates whether the officer is a corporate body rather than a natural person. **Example:** `false`
report. disqualified	**boolean** Indicates whether the report was requested with the `disqualified` flag. When `true`, the `disqualifications` array is populated instead of `items`. **Example:** `false`
report. report_date	**string (date-time)** ISO 8601 UTC timestamp indicating when the report was generated. **Example:** `2026-02-07T23:48:28Z`
report. total_results	**integer** The total number of appointments (or disqualifications) held by this officer. **Example:** `1`
report. items_per_page	**integer** The pagination page size used when collecting data from Companies House. **Example:** `50`
report. start_index	**integer** The starting index of the items (always `0` in the report as all pages are collected). **Example:** `0`
report. items	**array of objects** Array of the officer's appointments across companies. Each entry represents a single role (e.g. director, secretary) at a specific company. Active appointments have no `resigned_on` field; former appointments include `resigned_on`.

Field	Description
report. items[]. name	**string** The officer's name as recorded for this appointment. **Example:** `Jane Elizabeth DOE`
report. items[]. officer_role	**string** The role held at the company. Common values include `director`, `secretary`, `corporate-secretary`, `corporate-director`, `llp-member`, `llp-designated-member`, `managing-officer`, among others. **Example:** `secretary`
report. items[]. appointed_on	**string (date)** The date the officer was appointed to this role. **Example:** `2020-01-15`
report. items[]. resigned_on	**string (date)** The date the officer resigned from this role. Absent for active appointments. **Example:** `2019-12-31`
report. items[]. appointed_to	**object** The company this appointment relates to.
report. items[]. appointed_to. company_name	**string** The registered name of the company. **Example:** `TEST COMPANY LTD`
report. items[]. appointed_to. company_number	**string** The Companies House company number. **Example:** `12345678`
report. items[]. appointed_to. company_status	**string** The current status of the company (e.g. `active`, `dissolved`). **Example:** `active`
report. items[]. address	**object** The correspondence address recorded for this appointment.
report. items[]. address. premises	**string** **Example:** `456`

Field	Description
report. items[]. address. address_line_1	**string** **Example:** `Sample Road`
report. items[]. address. address_line_2	**string**
report. items[]. address. care_of	**string**
report. items[]. address. locality	**string** **Example:** `London`
report. items[]. address. region	**string**
report. items[]. address. country	**string** **Example:** `United Kingdom`
report. items[]. address. postal_code	**string** **Example:** `EC1A 2CC`
report. items[]. address. po_box	**string**
report. items[]. name_elements	**object** Broken-down components of the officer's name.
report. items[]. name_elements. title	**string** **Example:** `Ms`
report. items[]. name_elements. forename	**string** **Example:** `Jane`
report. items[]. name_elements. other_forenames	**string** **Example:** `Elizabeth`
report. items[]. name_elements. surname	**string** **Example:** `DOE`

Field	Description
report. items[]. name_elements. honours	**string**
report. items[]. former_names	**array of objects** Previous names for the officer, if any.
report. items[]. former_names[]. forenames	**string**
report. items[]. former_names[]. surname	**string**
report. items[]. nationality	**string** The officer's nationality. May be absent for some appointments. **Example:** `British`
report. items[]. country_of_residence	**string** The officer's country of residence. May be absent. **Example:** `England`
report. items[]. occupation	**string** The officer's stated occupation. May be absent.
report. items[]. links	**object** Links to related Companies House resources.
report. items[]. links. company	**string** Relative link to the company profile. **Example:** `/company/12345678`
report. items[]. identification	**object** Identification details for corporate officers. Absent for natural persons.
report. items[]. identification. identification_type	**string** **Example:** `uk-limited-company`
report. items[]. identification. legal_authority	**string**

Field	Description
report. items[]. identification. legal_form	**string**
report. items[]. identification. place_registered	**string**
report. items[]. identification. registration_number	**string**
report. items[]. is_pre_1992_appointment	**boolean** Whether the officer was appointed before 1992. **Example:** `false`
report. items[]. appointed_before	**string (date)** Present only when `is_pre_1992_appointment` is `true`.
report. items[]. contact_details	**object** Contact details for corporate managing officers.
report. items[]. contact_details. contact_name	**string**
report. items[]. responsibilities	**string** Responsibilities of a managing officer.
report. disqualifications	**array of objects** Array of disqualification records. Only present when the report was requested with `disqualified=true`. When present, the `items` array will typically be empty.
report. disqualifications[]. case_identifier	**string** **Example:** `DIS/2020/001`
report. disqualifications[]. disqualification_type	**string** **Example:** `court-order`
report. disqualifications[]. disqualified_from	**string (date)** **Example:** `2020-06-01`
report. disqualifications[]. disqualified_until	**string (date)** **Example:** `2030-05-31`

Field	Description
report. disqualifications[]. reason	**object**
report. disqualifications[]. reason. section	**string** **Example:** `6`
report. disqualifications[]. reason. act	**string** **Example:** `Company Directors Disqualification Act 1986`
report. disqualifications[]. reason. description_identifier	**string** **Example:** `conviction-of-indictable-offence`
report. disqualifications[]. company_names	**array of strings**
report. disqualifications[]. heard_on	**string (date)** **Example:** `2020-05-15`
report. disqualifications[]. court_name	**string** **Example:** `High Court of Justice`

Sample response


```
{
    "message": "Ok",
    "function": "company_house_officer_show",
    "api_reference": "7cbb094c-0f27-42f4-9f36-9cf0b2f6d5ff",
    "report": {
        "officer_id": "XYZ789GHI012",
        "name": "Jane Elizabeth DOE",
        "date_of_birth": "1980-03",
        "etag": "sandbox-etag-002",
        "kind": "personal-appointment",
        "type": "natural",
        "is_corporate_officer": false,
        "disqualified": false,
        "report_date": "2026-02-03T10:00:00Z",
        "total_results": 1,
        "items_per_page": 50,
        "start_index": 0,
        "items": [
            {
                "name": "Jane Elizabeth DOE",
                "officer_role": "secretary",
                "appointed_on": "2020-01-15",
                "appointed_to": {
                    "company_name": "TEST COMPANY LTD",
                    "company_number": "12345678",
                    "company_status": "active"
```

```
                },
                "address": {
                    "premises": "456",
                    "address_line_1": "Sample Road",
                    "locality": "London",
                    "country": "United Kingdom",
                    "postal_code": "EC1A 2CC"
                },
                "nationality": "British",
                "name_elements": {
                    "title": "Ms",
                    "forename": "Jane",
                    "other_forenames": "Elizabeth",
                    "surname": "DOE"
                },
                "country_of_residence": "England",
                "links": {
                    "company": "/company/12345678"
                },
                "is_pre_1992_appointment": false
            }
        ]
    }
}
```


Also available as a PDF (application/pdf)

202 **Report is still being prepared (returned when no data is available after the 30-second polling window).**
(application/json)

Field	Description
`message`	**string** Status message indicating the report is not ready yet. **Example:** `Extract is still being processed. Please try again later.`

Sample response


```
{
    "message": "Extract is still being processed. Please try again later."
}
```


404 **No report was found for the provided `api_reference`, or it belongs to another account/environment.**
(application/json)

Field	Description
`message`	**string** Error message describing that the record could not be located. **Example:** `Not Found.`

Sample response

```
{
    "message": "Not Found."
}
```

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

Company House Document

GET `/company_house_document/{documentId}`

Retrieves a document from UK Companies House by its document ID. Documents are typically annual accounts, confirmation statements, or other filings associated with a company.

Document Types

Common document types include:

Document Type	Description
Annual Accounts	Financial statements filed annually
Confirmation Statement	Annual statement confirming company details
Articles of Association	Company constitution document
Certificate of Incorporation	Original registration certificate
Memorandum of Association	Founding document for older companies
Change of Director	Notification of officer changes

Response Format

This endpoint returns the document as a PDF file (`application/pdf`). If the requested document is not available in PDF format, a `415 Unsupported Media Type` error is returned.

Sandbox Environment

The sandbox dataset is deterministic and does not use live data. Use document IDs with the `sandbox-document-` prefix to receive a placeholder PDF document.

Document ID	Description
sandbox-document-accounts-2023	Sample annual accounts document
sandbox-document-confirmation	Sample confirmation statement
sandbox-document-incorporation	Sample certificate of incorporation

Any document ID starting with `sandbox-document-` will return a valid placeholder PDF. Document IDs not matching this format will return a `400` error in the sandbox environment.

Path parameters

Field	Description
`documentId` REQUIRED	string The unique document ID from Companies House filing history. **Example:** `DOC123456789`

Responses

200 **Document retrieved successfully.** (application/pdf)

Binary PDF file

415 **The requested document is not available in PDF format.** (application/json)

Field	Description
`message`	string Error message indicating the document format is not supported. **Example:** `Document media type is not supported`

Sample response

```
{
    "message": "Document media type is not supported"
}
```

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

UK Business Check

POST `/uk_business_check`

Performs a synchronous check of a single UK company by its Companies House company number. This is the UK equivalent of the Australian `/business_check` endpoint: one request, one JSON reply, no polling.

The response combines the Companies House company profile, officers and filing history into a single flat `match` object.

12 month window

Officers and filings are limited to the last 12 months so the response reflects recent activity rather than the full company record:

- `officers` contains officers who were **appointed or resigned** within the last 12 months. Long-standing officers with no change in that period are omitted, so this is a list of officer changes, not the current board. Use `/company_house_report` for the full officer list.
- `filing_history` contains filings dated within the last 12 months, newest first.

Note that the sandbox fixture companies carry fixed appointment and filing dates, so their `officers` and `filing_history` may be empty once those dates fall outside the window.

Company numbers

UK company numbers are always 8 characters. Most are numeric with leading zeros (e.g. `01234567`); Scottish, Northern Irish, LLP and other registrations use a two letter prefix (e.g. `SC123456`, `NI012345`, `OC301234`). The number is case-insensitive; it is uppercased before lookup.

Not found

If the company number is well-formed but Companies House has no record of it, the call succeeds with `"match": []`. This is still a billable attempt.

Sandbox Environment

The sandbox dataset is deterministic and does not use live data. Any other company number returns an empty `match`.

Company Name	Company Number	Status
TEST COMPANY LTD	12345678	Active
SAMPLE HOLDINGS PLC	87654321	Active
DORMANT SERVICES LTD	11223344	Dormant
DISSOLVED EXAMPLE LTD	99887766	Dissolved
SCOTTISH ENTERPRISE SC	SC654321	Active
DELAYED REPORT LTD	55667788	Active

Request body

The company number to check.

Field	Description
`number` REQUIRED	string The 8-character UK Companies House company number. **Pattern:** `^[A-Za-z0-9]{8}$` **Example:** `01234567`

Sample request

```
{
    "number": "01234567"
}
```

Responses

200 **Check complete.** (application/json)

Field	Description
`message`	**string** Always `Ok` on success. **Example:** `Ok`
`api_reference`	**string (uuid)** Audit reference for this call. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`match`	**one of 2 shapes** The matched company. An empty array `[]` when the company number is not known to Companies House.

Field	Description

UK BUSINESS CHECK MATCH

Field	Description
`company_number`	**string** **Example:** `01234567`
`company_name`	**string** **Example:** `ACME TRADING LIMITED`
`company_status`	**string** Companies House status, e.g. `active`, `dissolved`, `liquidation`, `dormant`. **Example:** `active`
`company_status_detail`	**string or null**
`type`	**string** Companies House company type, e.g. `ltd`, `plc`, `llp`. **Example:** `ltd`
`jurisdiction`	**string or null** **Example:** `england-wales`
`date_of_creation`	**string (date) or null** **Example:** `2001-03-14`
`date_of_cessation`	**string (date) or null**
`registered_office_address`	**object**
`registered_office_address.address_line_1`	**string or null** **Example:** `1 High St`
`registered_office_address.address_line_2`	**string or null**
`registered_office_address.locality`	**string or null** **Example:** `London`
`registered_office_address.region`	**string or null**
`registered_office_address.postal_code`	**string or null** **Example:** `SW1A 1AA`
`registered_office_address.country`	**string or null** **Example:** `England`
`previous_company_names`	**array of objects**
	string

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "match": {
        "company_number": "01234567",
        "company_name": "ACME TRADING LIMITED",
        "company_status": "active",
        "company_status_detail": null,
        "type": "ltd",
        "jurisdiction": "england-wales",
        "date_of_creation": "2001-03-14",
        "date_of_cessation": null,
        "registered_office_address": {
            "address_line_1": "1 High St",
            "address_line_2": null,
            "locality": "London",
            "region": null,
            "postal_code": "SW1A 1AA",
            "country": "England"
        },
        "previous_company_names": [
            {
                "name": "ACME LIMITED",
                "effective_from": "1998-01-01",
                "ceased_on": "2001-03-14"
            }
        ],
        "sic_codes": [
            "62012",
            "62020"
        ],
        "officers": [
            {
                "officer_id": "abc123",
                "name": "SMITH, John",
                "officer_role": "director",
                "appointed_on": "2020-05-01",
                "resigned_on": null
            }
        ],
        "filing_history": [
            {
                "transaction_id": "MzQ1NjAwMQ",
                "document_id": "8j9Kx2",
                "date": "2026-06-30",
                "type": "AA",
                "category": "accounts",
                "description": "accounts-with-accounts-type-micro-entity"
            }
        ]
    }
}
```


Standard error responses: 400, 401, 402, 403, 429, 503, 5XX (see Common error responses).

UK Business Check (Bulk)

POST `/uk_business_checks`

Checks up to 50 UK companies in a single synchronous request. Each item in `requests` is looked up exactly as `/uk_business_check` would, and the results are returned in the same order with the caller's `check_id` echoed back for correlation.

Per-item errors

A problem with one item does not fail the batch. Items that fail carry an `error` string and an empty `match`, and are not billed:

- `The number is not a valid UK company number.` - the number is not 8 alphanumeric characters.
- `Companies House service is currently unavailable.` - Companies House could not be reached for that item.

Items where the number is valid but Companies House holds no record return `"match": []` with no `error`, and are billed as an attempt.

Rate limits

Each item may require up to three Companies House requests. Batches are processed sequentially, so large batches take proportionally longer to respond.

Sandbox Environment

Uses the same deterministic fixture set as `/uk_business_check`.

Request body

Field	Description
`requests` REQUIRED	**array of objects [1..50 items]**
`requests[].` `check_id`	**string [max 255 characters]** Optional caller-supplied identifier, echoed back on the matching result. **Example:** `entity-uuid-1`

`requests[].`
`number`
REQUIRED
	string [max 255 characters]
	The 8-character UK Companies House company number.
	Example: `01234567`

Sample request

```
{
    "requests": [
        {
            "check_id": "entity-uuid-1",
            "number": "01234567"
        }
```

```
    ]
}
```

Responses

200 **Batch complete. Inspect each result for `error`.** (application/json)

Field	Description
`message`	**string** **Example:** `Ok`
`api_reference`	**string (uuid)** Audit reference for this call. **Example:** `fe4291ca-d831-4760-96df-c9cb03b3cd95`
`results`	**array of objects**
`results[].` `check_id`	**string or null** **Example:** `entity-uuid-1`
`results[].` `error`	**string** Present only when this item could not be checked. **Example:** `The number is not a valid UK company number.`
`results[].` `match`	**one of 2 shapes** The matched company, or an empty array `[]` when not found or on error.

Field	Description

UK BUSINESS CHECK MATCH

Field	Description
`company_number`	**string** **Example:** `01234567`
`company_name`	**string** **Example:** `ACME TRADING LIMITED`
`company_status`	**string** Companies House status, e.g. `active`, `dissolved`, `liquidation`, `dormant`. **Example:** `active`
`company_status_detail`	**string or null**
`type`	**string** Companies House company type, e.g. `ltd`, `plc`, `llp`. **Example:** `ltd`
`jurisdiction`	**string or null** **Example:** `england-wales`
`date_of_creation`	**string (date) or null** **Example:** `2001-03-14`
`date_of_cessation`	**string (date) or null**
`registered_office_address`	**object**
`registered_office_address.address_line_1`	**string or null** **Example:** `1 High St`
`registered_office_address.address_line_2`	**string or null**
`registered_office_address.locality`	**string or null** **Example:** `London`
`registered_office_address.region`	**string or null**
`registered_office_address.postal_code`	**string or null** **Example:** `SW1A 1AA`
`registered_office_address.country`	**string or null** **Example:** `England`
`previous_company_names`	**array of objects**
	string

Sample response


```
{
    "message": "Ok",
    "api_reference": "fe4291ca-d831-4760-96df-c9cb03b3cd95",
    "results": [
        {
            "check_id": "entity-uuid-1",
            "error": "The number is not a valid UK company number.",
            "match": {
                "company_number": "01234567",
                "company_name": "ACME TRADING LIMITED",
                "company_status": "active",
                "company_status_detail": null,
                "type": "ltd",
                "jurisdiction": "england-wales",
                "date_of_creation": "2001-03-14",
                "date_of_cessation": null,
                "registered_office_address": {
                    "address_line_1": "1 High St",
                    "address_line_2": null,
                    "locality": "London",
                    "region": null,
                    "postal_code": "SW1A 1AA",
                    "country": "England"
                },
                "previous_company_names": [
                    {
                        "name": "ACME LIMITED",
                        "effective_from": "1998-01-01",
                        "ceased_on": "2001-03-14"
                    }
                ],
                "sic_codes": [
                    "62012",
                    "62020"
                ],
                "officers": [
                    {
                        "officer_id": "abc123",
                        "name": "SMITH, John",
                        "officer_role": "director",
                        "appointed_on": "2020-05-01",
                        "resigned_on": null
                    }
                ],
                "filing_history": [
                    {
                        "transaction_id": "MzQ1NjAwMQ",
                        "document_id": "8j9Kx2",
                        "date": "2026-06-30",
                        "type": "AA",
                        "category": "accounts",
                        "description": "accounts-with-accounts-type-micro-entity"
                    }
                ]
            }
        }
    ]
}
```


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 return a JSON object containing the `message` shown below.

Code	Description	Message
`400`	Bad or missing input parameter	
`401`	Unauthorized - The request is not authorized.	`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`
`429`	Rate limited - The request rate limit has been reached	`Rate limited`
`503`	DVS Service is unavailable. This is returned if the DVS returns a System Error result. The request was not successful and should be retried later.	`DVS returned a system error. Please try again later.`
`5XX`	Internal error - Internal system error	`Internal error`

RATE LIMIT HEADERS

Header	Type	Description
`x-ratelimit-limit`	integer	Request limit per hour.
`x-ratelimit-remaining`	integer	The number of requests left for the time window.