

The Google OAuth client, in your organisation

How your organisation creates the Google OAuth client that YAAPI Agent runs on, inside your own Google Workspace and Google Cloud, what that client lets you control, and how you revoke it.

Written for the Workspace or Cloud administrator. About thirty minutes, once. The live version of this document is at agent.yaapi.net/help/enterprise; the privacy statement it refers to is at agent.yaapi.net/privacy.

In one paragraph

YAAPI Agent is a web application at `agent.yaapi.net` that reads and, when allowed, changes the configuration of Google Analytics, Tag Manager, BigQuery, Google Sheets and Google Ads from instructions written in plain language. It never holds a Google password. Signing in uses the application's own Google sign-in and asks for name, email and picture only. Every call to a Google product is then made with an OAuth client that **your organisation creates in its own Google Cloud project**, and with tokens each user grants through that client. Because the client is yours and its audience is *Internal*, Google requires no verification, shows no warning screen, and the consent screen carries your own app name. Deleting that client, in your console, cuts the agent off from every user at once.

What the organisation controls

Control	Where	Effect
Who can grant access at all	Consent screen audience = Internal	Only accounts in your Workspace organisation can complete the consent flow. A personal Gmail account is refused by Google.
Which products are reachable	Cloud Console → APIs & Services → Enabled APIs	An API you do not enable cannot be called, whatever the token says.
What each user can do	The user's own Google role on each property, container, dataset	The agent acts as the user. It can never do more than that person already can in the Google interface.
Read-only or read-and-write	Chosen per product, by each user, in the application's Settings	Read-only is the default and is enforced by Google's own scopes: a read-only token carries no write permission.
Visibility of every call	Cloud Console → APIs & Services → Dashboard; Cloud Audit Logs	Traffic, errors and quota use per API, in your project, under your retention.
Full revocation	Delete the OAuth client, or disable the APIs	Every token issued through the client stops working immediately.
Per-user revocation	Admin console → Users → Security → Connected applications	One person's grant is removed without touching anyone else's.

Two OAuth clients, two jobs

	Sign-in client	Your organisation's client
Owned by	YAAPI, in its own project	You, in your project
Used for	Identifying the person who signs in	Every call to a Google product, and every token renewal

	Sign-in client	Your organisation's client
Scopes requested	<code>openid</code> , <code>email</code> , <code>profile</code> — nothing else, ever	The product scopes listed below, per product, per level
Google verification	Not needed: identity scopes only	Not needed: an Internal app is never reviewed by Google. The sensitive Analytics and Tag Manager scopes are carried by this client, in your project, under your name.
What YAAPI stores	Email, name, picture	Client ID, client secret and the refresh tokens users grant — encrypted at rest (AES-256-GCM), in a database hosted in the European Union

The consequence for your review: no third party ever holds a credential that can reach your Google data on its own. The client secret alone cannot mint a token; a token exists only because one of your users granted it, through your client, from an account you administer.

Prerequisites

- A **Google Workspace organisation**. The Internal audience is only offered to projects that belong to one; a project under « No organization » cannot use it.
- A **Google Cloud project inside that organisation**, new or existing, and an account with the right to create credentials in it (the predefined role *OAuth Config Editor* or *Project Editor*).
- If your organisation restricts which third-party apps may access Workspace data (*Admin console* → *Security* → *API controls*), an administrator who can add an app to the trusted list. An Internal app of your own is usually trusted by default; check before rollout.
- The users who will work with the agent already hold their roles on the Analytics properties, Tag Manager containers, datasets and Ads accounts they are meant to touch. The agent adds no access of its own.

Creating the client

Every step happens in Google's consoles, under your accounts. YAAPI is involved only at step 5, when the client is registered in the application.

1 Create or pick the Google Cloud project

console.cloud.google.com/projectcreate, with **Organization** set to your organisation and a name your inventory will recognise, for instance `yaapi-agent`. An existing project reserved for analytics tooling is fine. Apply your usual labels, budget alerts and ownership: BigQuery queries run by the agent are billed to this project.

2 Enable the APIs you intend to allow

APIs & Services → *Library*. Enable only what the teams will use; an API left disabled cannot be reached, whatever the token carries.

- Google Analytics Admin API — `analyticsadmin.googleapis.com`
- Google Analytics Data API — `analyticsdata.googleapis.com`
- Tag Manager API — `tagmanager.googleapis.com`
- BigQuery API — `bigquery.googleapis.com`
- Google Sheets API — `sheets.googleapis.com`
- Google Ads API — `googleads.googleapis.com`

Enabling an API costs nothing by itself. Google Ads additionally requires a developer token, which belongs to the YAAPI deployment, not to your project; enabling the API on your side is still needed for the OAuth grant.

3 Configure the consent screen with an Internal audience

APIs & Services → OAuth consent screen, which Google now presents as Google Auth Platform → Branding and Audience.

1. **Branding:** app name as your users should see it on the consent screen (for instance « YAAPI Agent — Company »), a support email in your domain, and your organisation's home page. The privacy and terms links may point at <https://agent.yaapi.net/privacy> and <https://agent.yaapi.net/terms>, or at your own pages.
2. **Audience → User type: Internal.** This is the decision the whole document rests on. Internal means: only accounts in your Workspace organisation can grant access; Google performs no verification; no « unverified app » screen is shown; tokens do not expire on a seven-day schedule as they do for an External app left in testing.
3. **Data access (scopes):** nothing to declare. The application names the scopes it needs at the moment a user connects a product, and the consent screen lists them to that user. Declaring them here is optional and changes nothing for an Internal app.

Do not choose External. External is the setting for apps published to the public; it triggers Google's verification for the Analytics and Tag Manager scopes, or an « unverified » warning if skipped. None of that applies to an application your organisation runs for itself.

4 Create the OAuth client

APIs & Services → Credentials → Create credentials → OAuth client ID.

1. Application type: **Web application.** Name it after the team or site it is for (see the note on scoping below).
2. **Authorized redirect URIs:** add exactly
<https://agent.yaapi.net/api/auth/google/callback>
Google compares it character by character: same scheme, no trailing slash. No JavaScript origin is needed.
3. **Create.** Google shows a Client ID (ending in [.apps.googleusercontent.com](https://www.googleapis.com/oauth2/v1/userinfo)) and a Client secret (starting with G0CSPX-). Store both in your secret manager; the secret is shown once.

One client, or one per team? The application registers a client per user, so any number of users may share one client. One client is the simplest to operate. One client per business unit or site lets you revoke a whole team by deleting a single credential, and shows traffic per team in the dashboard. Both work; pick the one that matches how you already segment access.

5 Hand the client to the users, who register it once

Each user signs in at <https://agent.yaapi.net> with their Workspace account, opens *Settings → Connections → Your Google Cloud project → OAuth client*, pastes the Client ID and the Client secret, and saves. The secret is encrypted before it is stored and is never displayed again. Distribute it through your secret manager or password vault, never by email or chat.

Nothing else is configured on the YAAPI side. There is no organisation object to create and no administrator role to assign: your Workspace is the directory, and your Cloud project is the policy.

6 Users connect the products they need, read-only by default

Still in Settings, each Google product has its own card with **Read-only** and **Read & write**. Google opens your consent screen for that product only and the token carries exactly the scopes of that level. Read-only is enforced by Google, not by the application: the token has no write permission to misuse. Grant write access product by product, to the people whose role already allows it.

What each level allows, product by product

All scopes are under <https://www.googleapis.com/auth/>. These are the scopes the application requests; the ones Google actually grants are read back from the token, and a token that carries any write scope is treated as read-and-write.

Product	Read-only	Read & write
Google Analytics	<code>analytics.readonly</code> <code>analytics.manage.users.readonly</code>	<code>analytics.edit</code> <code>analytics.readonly</code> <code>analytics.manage.users</code>
Tag Manager <code>tagmanager.delete.containers</code> is requested only when a user turns container deletion on explicitly; it is never part of the default grant.	<code>tagmanager.readonly</code>	<code>tagmanager.edit.containers</code> <code>tagmanager.edit.containerversions</code> <code>tagmanager.publish</code> <code>tagmanager.manage.accounts</code> <code>tagmanager.manage.users</code>
BigQuery Google publishes no BigQuery-specific read scope; cloud-platform.read-only is the narrowest one that still allows a SELECT.	<code>cloud-platform.read-only</code>	<code>bigquery</code>
Google Sheets Sheets are only ever read, on either level.	<code>spreadsheets.readonly</code>	<code>spreadsheets.readonly</code>
Google Ads Google publishes a single Ads scope covering read and write. On a read-only connection the application refuses every change before it is sent.	<code>adwords</code>	<code>adwords</code>

Whatever the level, the agent acts **as the user**: a change appears under that person's name in Google Analytics' change history and Tag Manager's version history, and a user who is Viewer on a property cannot edit it through the agent either. Destructive operations, publishing a Tag Manager version and every Google Ads change also require the user's explicit approval inside the application before they are sent.

What you can see, and where

Every API call, per product

Cloud Console → *APIs & Services* → *Dashboard* in your project: traffic, error rate, latency and quota use for each enabled API. The agent has no other route to Google.

Who granted what

Admin console → *Users* → (user) → *Security* → *Connected applications* lists the app and the scopes that person granted. The user sees the same at `myaccount.google.com/permissions`.

What changed in Analytics and Tag Manager

Google's own change history and version history, under the user's name. Inside the application, an action log records every write with the values it replaced; the user can export it from Settings.

BigQuery access

Cloud Audit Logs (Data Access) in your project, and the job history in the BigQuery console. Queries are billed to this project, so cost alerts apply.

Quotas

Your project's own, shared with nobody. Tag Manager's is the tightest; it can be raised from *APIs & Services* → *Quotas*.

Revoking access

Three levels, each independent of the others, each taking effect at once.

Scope of revocation	Action	Effect on the agent
Whole organisation	Cloud Console → Credentials → delete the OAuth client. Or disable the APIs in the project.	Every refresh token issued through the client is invalid. Every user's product cards show « reconnect », and no call can be made until a new client is registered.
One user	Admin console → Users → Security → Connected applications → remove. Suspending or deleting the Workspace account does the same.	That person's tokens stop working; nobody else is affected.
One user, one product	The user, in Settings → Connections → Disconnect on that card, or from <code>myaccount.google.com/permissions</code> .	The token for that product is deleted on the YAAPI side and revoked at Google.

Rotating the client secret

Google Cloud lets you add a second secret to a client and disable the first, so the rotation can be staged. Because each user registered the secret in their own Settings, they must paste the new one; the old one stops renewing tokens the moment you disable it. A quarterly rotation, aligned with your other credentials, is a reasonable cadence.

Offboarding YAAPI entirely

Delete the client, then let users delete their accounts from Settings, which removes their conversations, logs and credentials immediately. Whatever is not deleted by hand expires on the retention windows stated on the privacy page.

Where the data goes

The privacy page at agent.yaapi.net/privacy sets this out in full and is kept in step with the running configuration. The points a security review usually asks first:

- **Google** receives the API requests, under your client and your users' tokens. Nothing about your Google data is cached by YAAPI beyond the abridged conversation log described on the privacy page.
- **The model provider** (Anthropic) receives the conversation and the results of the calls made for the user. It is called with a credential the organisation brings: an API key with Anthropic's commercial terms in the chat, or the organisation's own Claude workspace when the agent is used from a connected assistant. YAAPI has no key of its own in that path. Retention and training rules are therefore those of your Anthropic agreement, which is where a zero-data-retention arrangement is negotiated.
- **YAAPI** keeps the account profile, an abridged copy of the conversations, the technical log and the action log, in a database hosted in the European Union, for the windows the privacy page states. Credentials and tokens are encrypted at rest.
- **Never held:** Google passwords, Workspace directory data beyond the signing-in user's own profile, or a Google credential that works without one of your users having granted it.

Questions a review tends to ask

Does Google have to verify the application?

No. Verification applies to apps published to users outside their own organisation. Your client is Internal, so Google does not review it, and your users never see a warning screen. The sign-in client that YAAPI owns requests

identity scopes only, which need no verification either.

Can YAAPI reach our Google data without a user?

No. The client secret cannot obtain a token by itself. A token exists only after one of your users completes your consent screen, and it inherits exactly that user's rights.

Can a service account replace the OAuth client?

Yes for Analytics, Tag Manager, BigQuery and Sheets: a service account in your project, granted access asset by asset, registered by its JSON key. Changes are then signed by the robot rather than by a person, and Google Ads is out of reach, since the Ads API accepts no service account. Domain-wide delegation is not used and not needed.

Is there single sign-on or SCIM?

Sign-in is Google sign-in with your Workspace accounts; there is no separate password to manage. There is no SAML or SCIM integration: a user exists in YAAPI once they sign in, and stops being able to act the moment their Workspace account is suspended or their grant removed.

Where do the users work, and under whose Anthropic account?

Either in the chat at `agent.yaapi.net`, with an Anthropic API key you issue, or from your organisation's Claude workspace, which reaches YAAPI as a tool. In both cases the model is called under your agreement with Anthropic, never a personal subscription. The privacy page explains why that choice matters.

Who do we contact?

support@yaapi.net. Include the Client ID if the question concerns a client; never the secret.