api0

GUIDE 01

How to Put Your Existing API in Claude in Under 10 Minutes

Your application already does the work. Here's how to make it usable from Claude — no MCP server to write — with every action taken as the person who asked. The starting point for the Teams, WhatsApp and Telegram guides.

Updated 11 November 2026 · Read online: api0.ai/guides/your-api-in-claude

Your software has an API. It can create a ticket, pull an invoice, list today's jobs. But your users spend their day in Claude. Between the two, there's no bridge.

Building that bridge yourself means writing an MCP server, handling OAuth for every user, storing your application's credentials somewhere the AI can't read them, logging every call. Weeks of work.

This guide shows the short version with api0: import the API once, choose who can sign in, and add it to Claude. The same tools then serve Teams, WhatsApp and Telegram.

What you need

- An HTTP API reachable from the internet (REST/JSON).
- Its description: an OpenAPI spec (YAML or JSON), or failing that, the list of your endpoints.
- A Google account to sign in to app.api0.ai.

No API at hand to try it? Take one of the sample specs (CRM, ticketing, invoicing, field service, HR) and point it at a mock server.

Step 1 — Import the API (3 minutes)

1. Sign in at **app.api0.ai**.
2. **Tools** tab → import your OpenAPI spec.
3. Each operation becomes a **tool**: its name, description and parameters are what Claude sees.
4. A built-in test calls each tool before it counts as configured.

No spec? Describe your endpoints in plain English and api0 turns them into tools.

The tip that changes everything: write good descriptions. Claude picks a tool from its description. Say *when* to use it ("use before creating a contact, to avoid duplicates") and *what to confirm* before an irreversible action ("ask for confirmation before sending an invoice"). A good description is worth ten lines of prompt.

A few details handled for you:

- parameter names MCP clients reject (`$filter`, `page[size]`) are exposed under a clean name (`filter`, `page_size`) and sent to your API under their real name;
- an error from your application comes back to Claude as a readable result, so it can correct and retry;
- a failed call is never billed.

Step 2 — Set up access (3 minutes)

Access tab. Two questions.

Who can sign in? By default, people sign in with Google. A company can plug in its own identity provider (Entra ID or any OIDC provider): its employees then sign in with their usual account.

Which credential is your application called with? This is the key question. The options, simplest to finest:

Option	For
Identity token signed by api0	Your own application: it verifies the call comes from api0 and for which tenant. No secret to store.
Fixed token or headers	A third-party API with a single key.
OAuth client credentials	An API that issues service tokens.
Each person's own credential	Each user adds their own token, or connects their account via OAuth: every action is taken **as them**, with **their** permissions.

In every case, the credential stays in api0. Claude never sees it.

A third setting: **who pays for tool calls** — the tenant, or each person on their own plan.

Step 3 — Claude (2 minutes)

Channels tab → **MCP**. You'll find your tenant's MCP server URL and a Client ID.

In claude.ai → Settings → Connectors → **Add custom connector**:

1. paste the server URL;
2. choose **"Use your own OAuth client"** and paste the **Client ID** (not the URL), with no secret;
3. sign in.

That's it. Ask Claude "what tools do you have?", then ask a real question. For a team, create a **Claude Project** with domain instructions (a template is provided).

Cursor, Claude Desktop and other MCP clients connect the same way.

Next: Teams, WhatsApp, Telegram

The same tools, with the same access, can answer in a messaging app. Each channel has its own guide:

- **Microsoft Teams**, with an Azure Bot in your name: the Teams guide.
- **WhatsApp**, with your WhatsApp Business number: the WhatsApp guide.
- **Telegram**, with a bot made with @BotFather: the Telegram guide.

In each channel, every person links their messaging account to their api0 account once; after that they talk to your API like a colleague, and every action is taken as them.

What you get

- **One setup**: the same tools in Claude and, next, in Teams, WhatsApp and Telegram.
- **Every action taken as the person**, with their permissions in your system.
- **No secrets in the AI**: credentials stay encrypted in api0.
- **A log of every call** (Activity tab) and credits per call.
- **Nothing to host**: no MCP server, no new interface.

FAQ

My API is private, behind a VPN. api0 needs to reach it. Put it behind a gateway that only accepts calls signed by api0 (the "signed identity token" option).

I want to limit what the AI can do. Import only the operations that are useful, and prefer each person's own credential: your application then applies its own permissions.