api0

GUIDE 04

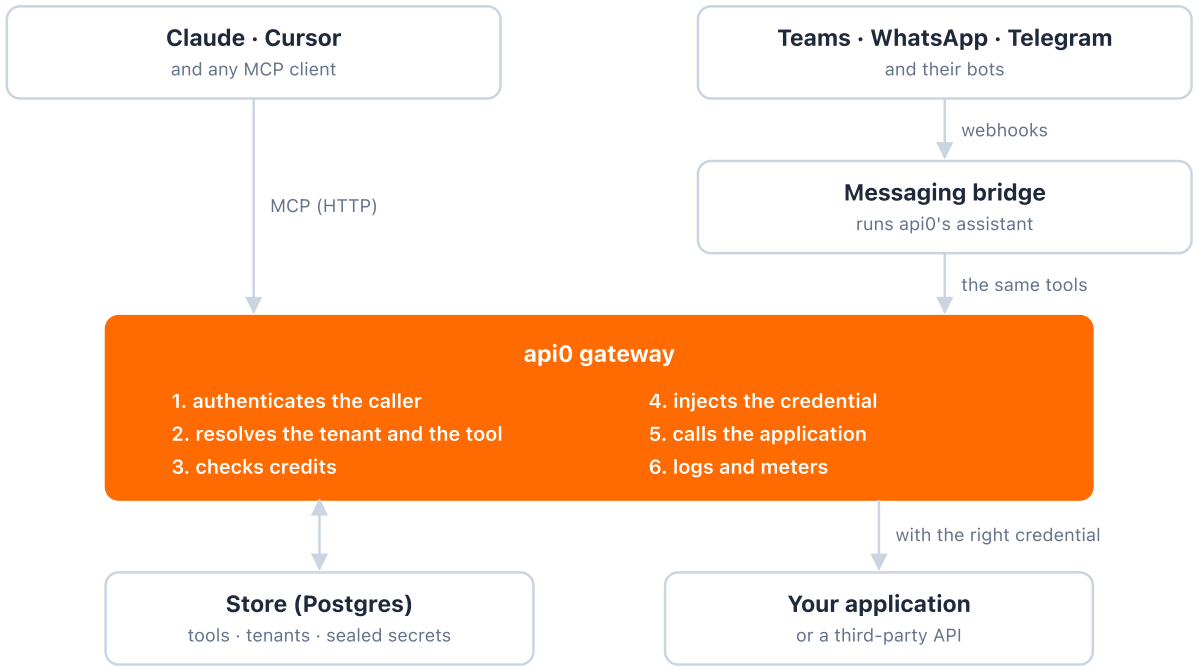
api0 Architecture: How a Request Travels

The technical guide: how a request travels through api0, from Claude, Teams, WhatsApp or Telegram to your application, what each component does, how secrets are protected, and the domains involved.

Updated 11 November 2026 · Read online: api0.ai/guides/architecture

This guide is for technical and security teams evaluating api0. It describes the path of a request and what each component does. The two authentication questions each have their own guide: **who is calling api0**, in inbound authentication (guide 10), and **which credential api0 uses to call your application**, in outbound authentication (guide 11).

Overview



The gateway is the only way through to your systems.

Three components, written in **Rust**:

- **The gateway** receives every call, authenticates it, routes it and logs it. It's the only path to your systems.
- **The store** holds the tool registry, tenants, credentials (encrypted) and the usage ledger.
- **The messaging bridge** receives webhooks from Teams, WhatsApp and Telegram and runs the assistant, which calls the same tools through the gateway.

The MCP gateway

`/mcp` is a standard **Model Context Protocol** server over HTTP (Streamable HTTP). Nothing proprietary at the protocol level: any MCP client works.

Method	Behaviour
`initialize`	Negotiates the protocol version, advertises tools, returns the tenant's instructions
`tools/list`	The caller's tenant's tools, and only those
`tools/call`	Routes to the application, meters, returns the result
`ping`	Empty result

Supported protocol versions: `2025-06-18`, `2025-03-26`, `2024-11-05`.

Tenant instructions. `initialize` returns a text (4,000 characters max) that the client places in front of the model: shared usage rules (load a tool before calling it, take IDs from a listing tool rather than guessing…), the tenant's own text (2,000 characters max) and, for each tool group, its starting tools.

Errors. A refusal from your application comes back as a **result** flagged `isError`, with your application's own message: the model reads it and can correct. JSON-RPC errors are reserved for api0's own refusals (unknown tool, invalid parameters, no credits). **A failed call is never billed.**

What a tool is

A tool is an entry in the registry, scoped to one tenant:

Field	Role
name, description, input schema	What the model sees
application URL	Target, with `{parameters}` substituted
HTTP verb	Set → REST call. Empty → the application speaks MCP natively
content type, body template	For APIs whose body isn't the arguments object (JSON Patch…)
static headers	The tool's own constants
identity forwarding	Whether to send the user's email and tenant
cost, timeout	Metering and patience

Importing an OpenAPI spec creates these entries automatically. **Body templates** cover the hard cases without code: an optional field that wasn't supplied disappears from the body (a whole JSON Patch operation in an array, a single key in an object). **Parameter names** MCP clients reject (`$filter`, `page[size]`) are exposed under a safe name and sent to the application under their real name.

Header order

Applied in this order, last wins:

1. **api0 identity** (`X-User-Email`, `X-Tenant-Id`) — only if identity forwarding is on;

2. the tool's **static headers**;
3. **outbound authentication** — always wins a name collision;
4. `Content-Type`.

The gateway never sends an internal secret to an application.

Protecting secrets

- **AES-256-GCM** encryption, sealed to the tenant and the secret's purpose: a secret copied into another context won't open.
- Secrets are never returned to the MCP client, the model or the dashboard (metadata only).
- With no encryption key configured, the store **refuses** to save a credential rather than keep it in plain text.

Personal data

Identity forwarding (`X-User-Email`, `X-Tenant-Id`) is a **per-tool** setting. It says who the call is for, not that it came from api0: an application that needs to know that verifies the outbound credential (the signed token). **Turn it off for any tool pointed at a third party**: your users' emails stay inside the platform.

Usage, credits and logs

- Every call is logged: tool, user, tenant, status, latency.
- Every successful call uses credits; a failure is never billed.
- Billing mode per tenant: **the tenant pays**, or **each person pays** on their own plan.

Domains

For a firewall rule or a security review, these are the addresses involved. Microsoft's only matter for tenants that use Microsoft (Entra ID) sign-in or Teams.

Domain	System	Role
`app.api0.ai`	api0 dashboard	Setup, **Channels**, **Settings** → **My access tokens** (Connect)
`gateway.api0.ai`	api0 gateway	Receives every call, picks the credential, calls the application
`gateway.api0.ai/mcp`	api0 gateway	The connector URL you add in Claude

Domain	System	Role
`gateway.api0.ai/oauth/downstream/callback`	api0 gateway	Where people come back after **Connect**, when each person connects their account with OAuth
`gateway.api0.ai/oauth/idp/callback`	api0 gateway	Where people come back after signing in to api0 with the tenant's identity provider
Messaging endpoint and webhooks (shown in **Channels**)	api0 messaging bridge	Where Teams, WhatsApp and Telegram deliver messages
`api0.ai`	api0 website	Privacy policy and terms the Teams app links to
`login.microsoftonline.com/<tenant-id>`	Entra ID	Sign-in and consent; issues and renews each person's token
`login.botframework.com`, `smba.trafficmanager.net`	Microsoft Bot Framework	Signs Teams messages, and carries the replies
`entra.microsoft.com`, `portal.azure.com`, `admin.microsoft.com`, `admin.teams.microsoft.com`	Microsoft consoles	Setup only: Entra app, Azure Bot, licences, approving the Teams app

Authentication

Two questions, two guides:

- **Who is calling api0?** OAuth 2.1 for Claude and MCP clients, Google or the company's identity provider for people, API keys for servers, account linking for messaging apps: inbound authentication (guide 10).
- **Which credential does api0 use to call your application?** The identity token signed by api0, a service key, or each person's own credential: outbound authentication (guide 11).