api0

GUIDE 03

Connect Your Internal Systems to Claude, Securely

Claude Enterprise is rolled out, but it doesn't know your ERP, your CRM or your HR tools. Here's how to open them to Claude without exposing secrets, with each employee's own permissions and a log of every call.

Updated 7 November 2026 · Read online: api0.ai/guides/connect-internal-systems-to-claude

---

You've rolled out Claude across the company. Teams use it to write, summarize, analyze. But as soon as a question touches company data — "where is order 4512?", "how many days off do I have left?", "open a ticket for the login bug" — Claude can't do anything. People copy and paste from their tools.

The technical answer exists: the **Model Context Protocol (MCP)**. Claude connects to MCP servers that give it tools. The problem is everything needed around them for security to say yes.

What your security team will ask

1. **Who is acting?** Each call must be made as the employee, with their permissions — not with an all-powerful service account.
2. **Where are the secrets?** Your systems' credentials must never pass through the AI.
3. **What was done?** You need a log: who, which tool, when.
4. **Who can sign in?** Employees, through the company identity provider. Not anyone with a link.
5. **Who uses what?** Usage must be measurable per team.

Writing an MCP server that ticks these five boxes, for every system, is an infrastructure project. api0 is that layer, ready-made, in front of your existing APIs.

The architecture

```
Employee in Claude
   │  OAuth 2.1 (PKCE) — sign-in via Entra ID / OIDC
   ▼
api0 (MCP gateway)
   │  checks identity · picks the credential · logs · meters
   ▼
Your internal APIs (ERP, CRM, HR, Azure DevOps…)
```

Claude only talks to api0. api0 talks to your systems, with the credential **you** chose.

Step 1 — Connect your identity provider

In the **Access** tab, set your tenant's sign-in to your **OIDC** provider: Entra ID, Okta, Google Workspace or another. You need an issuer, a client ID and a secret, created in your IdP for api0.

Result: when an employee adds the connector in Claude, they sign in with their usual company account (MFA, conditional access and the rest included). The IdP client secret is stored encrypted.

Step 2 — Import your internal APIs

Tools tab: import each system's OpenAPI spec, or describe the endpoints in plain English. Each operation becomes a tool, tested before it's enabled.

Import only what's useful. A tool you don't import doesn't exist for Claude. Start with reads (look up an order, a balance, a ticket), add writes later.

Your APIs must be reachable by api0. If they're internal, put a gateway in front that only accepts calls signed by api0 (see step 3).

Step 3 — Choose how api0 authenticates to your systems

This is the decision that matters most.

Best: each employee's identity. Two ways:

- **Their own account, via OAuth**: the employee connects their account in the target system once; api0 keeps and refreshes the token.
- **Their own token** (an Azure DevOps PAT, for example): the employee adds it once; it's encrypted and can't be read back, only replaced.

Either way, the target system sees **the employee**, applies **their** permissions, and attributes the action to **them** (the author of an Azure DevOps work item really is the person who asked). An employee who hasn't linked their account yet gets a message explaining how — never a call made as someone else.

For your own services: the identity token signed by api0. Your application verifies a Google identity token signed by api0, whose audience names your tenant. No secret to store on your side, and no other api0 customer can produce a token valid for you.

Last resort: a shared credential (fixed token, headers, OAuth client credentials). Simple, but every action is taken by the same account. Keep it for reading non-sensitive data.

Step 4 — Protect personal data

By default, api0 can pass the user's email and tenant ID to your services (`X-User-Email`, `X-Tenant-Id`). Useful for your own applications; **turn it off for any tool pointed at a third party**: your employees' emails then stay inside the platform.

Step 5 — Roll out in Claude

1. **Channels** tab → **MCP**: server URL and Client ID.
2. Add the connector in Claude (Settings → Connectors). On Claude Enterprise, an admin can make it available to the whole organization.
3. Create a **Claude Project** per team, with domain instructions: vocabulary, confirmation rules, answer format. A template is provided.

You can also write **tenant instructions** in api0 (up to 2,000 characters) that Claude receives on every connection: the main project, naming conventions, what to always check.

Step 6 — Track usage

- **Log** (Activity tab): every call, the tool, the person, the result, the latency.
- **Credits**: every successful call is metered; a failed call is never billed. You choose whether the tenant pays or each person on their own plan.

The security checklist

Requirement	api0's answer
Employee sign-in	OAuth 2.1 with PKCE, through your IdP (Entra ID / OIDC)
Actions as the employee	Per-person credential (OAuth or personal token)
Secrets kept out of the AI	Stored in api0, encrypted (AES-256-GCM), never returned to Claude
Least privilege	Only imported operations exist; the target system applies its permissions
Personal data	Identity forwarding can be turned off per tool
Traceability	A log of every call
Implementation	Gateway and store written in Rust

Where to start

One system, one team, two weeks:

1. **Days 1–3**: import one internal API, Entra ID sign-in, per-person credentials.
2. **Days 4–5**: Claude connector and Project for a pilot team.
3. **Days 6–12**: real use, add tools as requests come in.
4. **Days 13–14**: review with the usage log.

Next: read the detailed architecture.