api0

GUIDE 03

Connecter tes systèmes internes à Claude de façon sécurisée

Claude Enterprise est déployé, mais il ne connaît ni ton ERP, ni ton CRM, ni tes outils RH. Voici comment les lui ouvrir sans exposer de secrets, avec les droits de chaque employé et un journal de chaque appel.

Mis à jour le 7 novembre 2026 · Lire en ligne : api0.ai/fr/guides/claude-enterprise-systemes-internes

Tu as déployé Claude dans l'entreprise. Les équipes l'utilisent pour écrire, résumer, analyser. Mais dès qu'une question touche aux données de l'entreprise — « où en est la commande 4512 ? », « combien de jours de congé me reste-t-il ? », « crée un ticket pour le bug de connexion » — Claude ne peut rien faire. Les gens copient-collent depuis leurs outils.

La réponse technique existe : le **Model Context Protocol (MCP)**. Claude se connecte à des serveurs MCP qui lui donnent des outils. Le problème, c'est tout ce qu'il faut autour pour que la sécurité dise oui.

Ce que ta sécurité va demander

1. **Qui agit ?** Chaque appel doit se faire au nom de l'employé, avec ses droits — pas avec un compte de service tout-puissant.
2. **Où sont les secrets ?** Les identifiants de tes systèmes ne doivent jamais transiter par l'IA.
3. **Qu'est-ce qui a été fait ?** Il faut un journal : qui, quel outil, quand.
4. **Qui peut se connecter ?** Les employés, via le fournisseur d'identité de l'entreprise. Pas n'importe qui avec un lien.
5. **Qui consomme quoi ?** L'usage doit être mesurable par équipe.

Écrire un serveur MCP qui coche ces cinq cases, pour chaque système, c'est un projet d'infrastructure. api0 est cette couche, toute faite, devant tes APIs existantes.

L'architecture

```
Employé dans Claude
   │  OAuth 2.1 (PKCE) — connexion via Entra ID / OIDC
   ▼
api0 (passerelle MCP)
   │  vérifie l'identité · choisit l'identifiant · journalise · décompte
   ▼
Tes APIs internes (ERP, CRM, RH, Azure DevOps…)
```

Claude ne parle qu'à api0. api0 parle à tes systèmes, avec l'identifiant que **tu** as choisi.

Étape 1 — Brancher ton fournisseur d'identité

Dans l'onglet **Access**, configure la connexion de ton tenant avec ton fournisseur **OIDC** : Entra ID, Okta, Google Workspace ou autre. Il te faut un émetteur (issuer), un client ID et un secret, créés dans ton IdP pour api0.

Effet : quand un employé ajoute le connecteur dans Claude, il se connecte avec son compte d'entreprise habituel (MFA, accès conditionnel et tout le reste compris). Le secret client de l'IdP est stocké chiffré.

Étape 2 — Importer tes APIs internes

Onglet **Tools** : importe la spec OpenAPI de chaque système, ou décris les endpoints en langage naturel. Chaque opération devient un outil, testé avant d'être activé.

N'importe que ce qui est utile. Un outil non importé n'existe pas pour Claude. Commence par la lecture (consulter une commande, un solde, un ticket), ajoute l'écriture ensuite.

Tes APIs doivent être joignables par api0. Si elles sont internes, place devant elles une passerelle qui n'accepte que les appels signés par api0 (voir étape 3).

Étape 3 — Choisir comment api0 s'authentifie auprès de tes systèmes

C'est la décision qui compte le plus.

L'idéal : l'identité de chaque employé. Deux façons :

- **Son propre compte, en OAuth** : l'employé connecte une fois son compte du système cible ; api0 garde et rafraîchit le jeton.
- **Son propre jeton** (par exemple un PAT Azure DevOps) : l'employé l'ajoute une fois ; il est chiffré et ne peut plus être relu, seulement remplacé.

Dans les deux cas, le système cible voit **l'employé**, applique **ses** droits, et attribue l'action à **lui** (l'auteur d'un work item Azure DevOps est bien la personne qui a demandé). Un employé qui n'a pas encore relié son compte reçoit un message qui explique comment — jamais un appel fait au nom de quelqu'un d'autre.

Pour tes propres services : le jeton d'identité signé par api0. Ton application vérifie un jeton Google signé par api0, dont l'audience désigne ton tenant. Aucun secret à stocker de ton côté, et un autre client d'api0 ne peut pas produire un jeton valable pour toi.

En dernier recours : un identifiant partagé (jeton fixe, en-têtes, OAuth client credentials). Simple, mais toutes les actions sont faites par le même compte. À réserver à la lecture de données non sensibles.

Étape 4 — Protéger les données personnelles

Par défaut, api0 peut transmettre à tes services l'email de l'utilisateur et l'identifiant du tenant (`X-User-Email`, `X-Tenant-Id`). Utile pour tes propres applications, à **désactiver pour tout outil pointé vers un**

tiers : les emails de tes employés restent alors dans la plateforme.

Étape 5 — Déployer dans Claude

1. Onglet **Channels** → **MCP** : URL du serveur et Client ID.
2. Ajoute le connecteur dans Claude (Paramètres → Connecteurs). Sur Claude Enterprise, un admin peut le rendre disponible pour toute l'organisation.
3. Crée un **Projet Claude** par équipe, avec des instructions métier : vocabulaire, règles de confirmation, format des réponses. Un template est fourni.

Tu peux aussi écrire, dans api0, des **instructions du tenant** (jusqu'à 2 000 caractères) que Claude reçoit à chaque connexion : le projet principal, les conventions de nommage, ce qu'il faut toujours vérifier.

Étape 6 — Suivre l'usage

- **Journal** (onglet Activity) : chaque appel, l'outil, la personne, le résultat, la latence.
- **Crédits** : chaque appel réussi est décompté ; un appel en échec n'est jamais facturé. Tu choisis si c'est le tenant qui paie ou chaque personne avec son plan.

La checklist sécurité

Exigence	Réponse api0
Connexion des employés	OAuth 2.1 avec PKCE, via ton IdP (Entra ID / OIDC)
Actions au nom de l'employé	Identifiant par personne (OAuth ou jeton personnel)
Secrets hors de l'IA	Stockés dans api0, chiffrés (AES-256-GCM), jamais renvoyés à Claude
Moindre privilège	Seules les opérations importées existent ; le système cible applique ses droits
Données personnelles	Transmission de l'identité désactivable par outil
Traçabilité	Journal de chaque appel
Implémentation	Passerelle et stockage écrits en Rust

Par où commencer

Un système, une équipe, deux semaines :

1. **Jours 1–3** : import d'une API interne, connexion Entra ID, identifiants par personne.
2. **Jours 4–5** : connecteur Claude et Projet pour une équipe pilote.
3. **Jours 6–12** : usage réel, ajout d'outils selon les demandes.
4. **Jours 13–14** : bilan avec le journal d'usage.

Pour aller plus loin : lis l'architecture détaillée.