api0

GUIDE 04

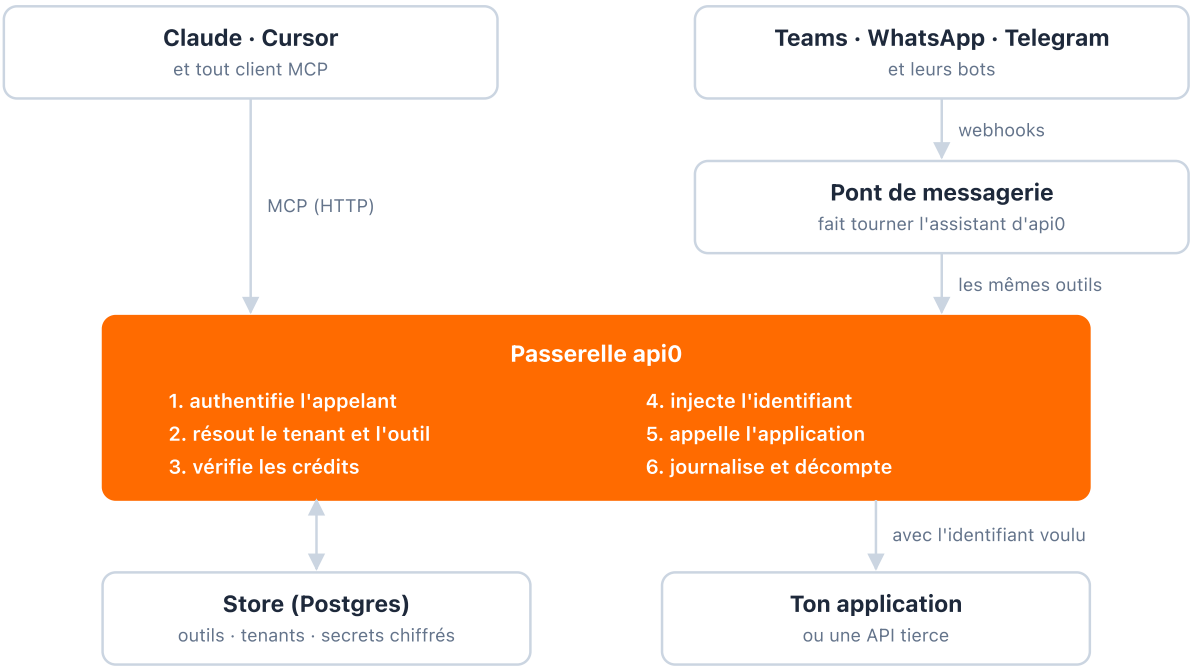
Architecture d'api0 : le chemin d'une requête

Le guide technique : comment une requête traverse api0, de Claude, Teams, WhatsApp ou Telegram jusqu'à ton application, ce que fait chaque composant, comment les secrets sont protégés, et les domaines en jeu.

Mis à jour le 11 novembre 2026 · Lire en ligne : api0.ai/fr/guides/architecture

Ce guide s'adresse aux équipes techniques et sécurité qui évaluent api0. Il décrit le chemin d'une requête et ce que fait chaque composant. Les deux questions d'authentification ont chacune leur guide : **qui appelle api0**, dans l'authentification entrante (guide 10), et **avec quel identifiant api0 appelle ton application**, dans l'authentification sortante (guide 11).

Vue d'ensemble



La passerelle est le seul point de passage vers tes systèmes.

Trois composants, écrits en **Rust** :

- **La passerelle** reçoit chaque appel, l'authentifie, le route et le journalise. C'est le seul point de passage vers tes systèmes.
- **Le store** garde le registre des outils, les tenants, les identifiants (chiffrés) et le registre d'usage.
- **Le pont de messagerie** reçoit les webhooks de Teams, WhatsApp et Telegram et fait tourner l'assistant, qui appelle les mêmes outils via la passerelle.

La passerelle MCP

`/mcp` est un serveur **Model Context Protocol** standard sur HTTP (Streamable HTTP). Rien de propriétaire côté protocole : tout client MCP fonctionne.

Méthode	Comportement
`initialize`	Négocie la version du protocole, annonce les outils, renvoie les instructions du tenant
`tools/list`	Les outils du tenant de l'appelant, et seulement eux
`tools/call`	Route vers l'application, décompte, renvoie le résultat
`ping`	Réponse vide

Versions du protocole acceptées : `2025-06-18`, `2025-03-26`, `2024-11-05`.

Instructions du tenant. `initialize` renvoie un texte (4 000 caractères max) que le client place devant le modèle : des règles d'usage communes (charger un outil avant de l'appeler, prendre les identifiants dans un outil de listing plutôt que de les deviner…), le texte propre au tenant (2 000 caractères max) et, pour chaque groupe d'outils, les outils de départ.

Erreurs. Un refus de ton application revient comme un **résultat** marqué `isError`, avec le message de ton application : le modèle le lit et peut corriger. Les erreurs JSON-RPC sont réservées aux refus d'api0 (outil inconnu, paramètres invalides, crédits épuisés). **Un appel en échec n'est jamais facturé.**

Ce qu'est un outil

Un outil est une entrée du registre, rattachée à un tenant :

Champ	Rôle
nom, description, schéma d'entrée	Ce que le modèle voit
URL de l'application	Cible, avec des `{paramètres}` substitués
verbe HTTP	Défini → appel REST. Vide → l'application parle MCP nativement
content-type, modèle de corps	Pour les APIs dont le corps n'est pas l'objet d'arguments (JSON Patch…)
en-têtes fixes	Constantes propres à l'outil
transmission d'identité	Envoyer ou non l'email et le tenant de l'utilisateur
coût, timeout	Décompte et patience

Importer une spec OpenAPI crée ces entrées automatiquement. Les **modèles de corps** couvrent les cas difficiles sans code : un champ optionnel non fourni disparaît du corps (une opération JSON Patch entière dans un tableau, une seule clé dans un objet). Les **noms de paramètres** que les clients MCP refusent (`$filter`, `page[size]`) sont exposés sous un nom sûr et renvoyés à l'application sous leur vrai nom.

Ordre des en-têtes

Appliqués dans cet ordre, le dernier gagne :

1. **identité api0** (`X-User-Email`, `X-Tenant-Id`) — seulement si la transmission d'identité est activée ;
2. **en-têtes fixes** de l'outil ;
3. **authentification sortante** — gagne toujours en cas de collision ;
4. `Content-Type`.

La passerelle n'envoie jamais de secret interne à une application.

Protection des secrets

- Chiffrement **AES-256-GCM**, scellé selon le tenant et l'usage du secret : un secret copié vers un autre contexte ne s'ouvre pas.
- Les secrets ne sont jamais renvoyés au client MCP, ni au modèle, ni au tableau de bord (lecture des métadonnées seulement).
- Sans clé de chiffrement configurée, le store **refuse** d'enregistrer un identifiant plutôt que de le garder en clair.

Données personnelles

La transmission de l'identité (`X-User-Email`, `X-Tenant-Id`) est un réglage **par outil**. Elle dit pour qui est l'appel, pas qu'il vient d'api0 : une application qui doit le savoir vérifie l'identifiant sortant (le jeton signé). **Désactive-la pour tout outil pointé vers un tiers** : les emails de tes utilisateurs restent dans la plateforme.

Usage, crédits et journal

- Chaque appel est journalisé : outil, utilisateur, tenant, statut, latence.
- Chaque appel réussi décompte des crédits ; un échec n'est jamais facturé.
- Mode de facturation par tenant : **le tenant paie**, ou **chaque personne paie** avec son plan.

Les domaines

Pour une règle de pare-feu ou une revue de sécurité, voici les adresses en jeu. Celles de Microsoft ne concernent que les tenants qui utilisent la connexion Microsoft (Entra ID) ou Teams.

Domaine	Système	Rôle
`app.api0.ai`	Tableau de bord api0	Configuration, **Channels**, **Settings → My access tokens** (Connect)

Domaine	Système	Rôle
`gateway.api0.ai`	Passerelle api0	Reçoit chaque appel, choisit l'identifiant, appelle l'application
`gateway.api0.ai/mcp`	Passerelle api0	L'URL du connecteur à ajouter dans Claude
`gateway.api0.ai/oauth/downstream/callback`	Passerelle api0	Retour après **Connect**, quand chacun connecte son compte en OAuth
`gateway.api0.ai/oauth/idp/callback`	Passerelle api0	Retour après la connexion à api0 par le fournisseur d'identité du tenant
Messaging endpoint et webhooks (affichés dans **Channels**)	Pont de messagerie api0	Où Teams, WhatsApp et Telegram livrent les messages
`api0.ai`	Site d'api0	Politique de confidentialité et conditions, vers lesquelles pointe l'application Teams
`login.microsoftonline.com/<tenant-id>`	Entra ID	Connexion et consentement ; émet et renouvelle le jeton de chacun
`login.botframework.com`, `smba.trafficmanager.net`	Microsoft Bot Framework	Signature des messages Teams, et envoi des réponses
`entra.microsoft.com`, `portal.azure.com`, `admin.microsoft.com`, `admin.teams.microsoft.com`	Consoles Microsoft	Configuration seulement : application Entra, Azure Bot, licences, approbation de l'application Teams

L'authentification

Deux questions, deux guides :

- **Qui appelle api0 ?** OAuth 2.1 pour Claude et les clients MCP, Google ou le fournisseur d'identité de l'entreprise pour les personnes, des clés API pour les serveurs, la liaison de compte pour les messageries : l'authentification entrante (guide 10).
- **Avec quel identifiant api0 appelle ton application ?** Le jeton d'identité signé par api0, une clé de service, ou l'identifiant de chaque personne : l'authentification sortante (guide 11).