

Documentation OpenLatch

Installez le client, définissez l'Autonomy Zone de chaque agent et déployez sur toute votre flotte.

Installer le client

Comprendre les verdicts →

Démarrer →

Installez le client et obtenez votre premier verdict.

Concepts →

Autonomy Zones, politiques d'intention et verdicts.

Guides →

Une tâche par page, de la rédaction jusqu'à Enforce.

Déployer →

Déployez le client par MDM ou gestion de configuration.

Référence →

Chaque commande, code d'erreur et champ de politique.

Journal des modifications →

Chaque version du client et chaque évolution de la plateforme.

Présentation

Ce parcours vous mène d'une machine neuve à votre première action décidée : vous installez le client à côté d'un agent, vous vous connectez à la console, vous écrivez une politique d'intention et vous la regardez décider.

OpenLatch est la couche de contrôle d'exécution des agents d'IA en entreprise. Le client s'exécute à côté de chaque agent et décide de chaque action couverte avant son exécution. La plateforme, sur `app.openlatch.ai`, est l'endroit où vous définissez l'Autonomy Zone de chaque agent et où vous lisez l'historique de ce qui s'est passé.

- 1

[Installez le client](#) sur une machine où s'exécute un agent de code.
- 2

[Connectez-vous à la console](#) avec la méthode qu'utilise votre organisation.
- 3

[Écrivez votre première politique](#) et ratifiez-la en Monitor.
- 4

Utilisez l'agent comme d'habitude, puis ouvrez **Anomalies** dans la console pour voir ce que la politique a enregistré.

Qui fait quoi

Vous êtes	Commencez par
Développeur avec un agent de code	Installer le client
Administrateur qui déploie le client sur de nombreuses machines	Installer le client , puis la section Déployer
Responsable sécurité ou conformité qui écrit les politiques	Écrire votre première politique , puis Politiques d'intention

Avant de commencer

- Une machine macOS, Linux ou Windows avec au moins un agent de code que le client raccorde. [Niveaux de contrôle](#) les liste, plan par plan.

- Un compte dans une organisation OpenLatch sur `app.openlatch.ai`.

Pour comprendre ce qu'est OpenLatch avant d'installer quoi que ce soit, lisez [Qu'est-ce qu'OpenLatch](#).

Installer le client

Installez le client sur la machine où s'exécute votre agent de code, afin que chaque action couverte soit décidée avant son exécution. Le client s'installe au niveau de l'utilisateur, sans droits d'administrateur. Choisissez la commande adaptée à la machine.

Sur toute machine où Node.js est déjà installé :

```
npx @openlatch/client init
```

Seul npx a besoin de Node.js. Le client lui-même est un binaire natif et ne s'exécute pas sur Node.js.

Aucun runtime requis :

```
curl -LsSf https://openlatch.ai/install.sh | sh
```

Aucun runtime requis :

```
powershell -ExecutionPolicy Bypass -c "irm https://openlatch.ai/install.ps1 | iex"
```

Chaque commande installe la même version du client depuis le registre npm, puis lance `openlatch init`, qui détecte les agents de code de la machine, raccorde leurs hooks et démarre le client.

Enrôler la machine

- 1 Sur une machine où vous êtes devant un terminal, `openlatch init` ouvre `app.openlatch.ai` dans votre navigateur lorsque la machine n'a pas encore d'identifiant. Connectez-vous, et le client enregistre l'identifiant dans le trousseau du système d'exploitation.
- 2 Sur une machine sans personne devant le terminal, créez d'abord une clé d'API sur la page **API Keys** de votre compte dans la console, et définissez-la dans `OPENLATCH_API_KEY` avant de lancer la

commande d'installation.

- 3 Lancez `openlatch status` et vérifiez que chaque agent apparaît avec son niveau de contrôle.

Note

Le téléchargement est vérifié par rapport à l'empreinte publiée par le registre. Cela détecte une corruption ou une troncature, pas un registre compromis.

Vérifier les hooks

Lancez `openlatch doctor`. Il diagnostique la configuration et la connexion à la plateforme et, pour tout ce qui ne va pas, il indique la cause et la commande qui la corrige.

Ce que l'installation modifie

- Elle installe pour votre utilisateur uniquement. Aucun droit d'administrateur n'est nécessaire.
- Avec un script, elle place deux binaires, `openlatch` et `openlatch-hook`, dans `~/.local/bin` sous macOS et Linux, ou dans `%LOCALAPPDATA%\openlatch\bin` sous Windows, et ajoute ce dossier à votre PATH. Un terminal déjà ouvert peut devoir être rouvert une fois.
- Avec `npx`, rien n'est ajouté à votre PATH : le client garde ses binaires dans son répertoire d'état, et un simple `openlatch` peut ne pas être trouvé. Lancez les commandes suivantes par la même commande `npx`, avec la commande voulue à la place de `init`, par exemple `status`.
- Avant d'installer quoi que ce soit, elle compare une empreinte SHA-512 du téléchargement avec celle que le registre publie pour cette version.

Code de sortie

Pour les outils de déploiement qui lisent le code de sortie d'un script d'installation :

Code	Signification
0	Installé et configuré. Le résultat peut encore signaler un point à examiner
6	Installé, mais la configuration n'a pas abouti. Le résultat à l'écran en donne la raison
7	Installé, en attente d'un agent d'IA : la machine n'en a pas encore. Installez-en un et relancez le script
2-5	Rien d'utilisable n'a été installé. Le script affiche un code 0L-21xx qui en donne la raison

Plateformes prises en charge

Plateforme	Statut
macOS, Apple silicon et Intel	Pris en charge
Linux avec glibc, x86_64 et arm64	Pris en charge
Windows, x64	Pris en charge
Linux avec musl, comme Alpine	Non pris en charge. Le script d'installation s'arrête et en donne la raison
Windows sur ARM64	Non pris en charge. Le script d'installation s'arrête et en donne la raison

Ensuite, [écrivez votre première politique](#).

Écrire votre première politique

Dans ce tutoriel, vous transformez une phrase en Agent Intent Policy (AIP), vous la ratifiez en Monitor et vous la regardez enregistrer une anomalie sans arrêter l'agent.

Avant de commencer, [installez le client](#) sur une machine dotée d'un agent de code et [connectez-vous à la console](#).

Écrire la politique

- 1 Dans la console, ouvrez **Autonomy Zone**, puis la liste des politiques, et lancez la création d'une nouvelle politique d'intention.
- 2 À l'étape **Brief**, collez une phrase qui dit ce que vous voulez, ou choisissez un exemple. Ce tutoriel utilise :

```
Don't let the agent fetch anything from pastebin.com.
```
- 3 À l'étape **Policy**, lisez la phrase qu'OpenLatch a générée à partir de votre brief, avec les cas de test affichés à côté. Chaque cas est une situation concrète et le verdict que rendrait la politique. Répondez Oui si la phrase dit ce que vous vouliez dire. Répondez Non et dites avec vos propres mots ce que vous vouliez dire : OpenLatch régénère la phrase et ses cas.
- 4 À l'étape **How it runs**, choisissez le mode **Monitor**.
- 5 À l'étape **Ratify**, choisissez la zone qui porte la politique, puis lancez la relecture. Elle applique la politique à l'activité récente de vos agents et montre ce qu'elle aurait décidé. Ratifiez la politique.

app.openlatch.ai/zone/policies/new

OpenLATCH

Autonomy Zone / New intent policy

Q Ctrl+K

?

99+ 99+ DC

Crispy Pay

MONITOR

- Home
- Agents
- Anomalies
- Inventory
- Activity Intelligence

MANAGE

- Autonomy Zone
- Activity Digests

SYSTEM

- Audit Trail
- Settings

New intent policy

Turn what you already have into an intent policy. Nothing runs until you ratify it.

1 Brief 2 Policy 3 How it runs 4 Ratify

Define the policy intent

Provide a brief or reference material. OpenLATCH will use it to draft an Agent Intent Policy for your review.

☐ Paste a brief
Type or paste what you mean, in your own words.

☐ Upload a policy
A document, an SOP, a procedure. We read it and pull the intents out.

☐ Record a conversation
Describe the rule out loud, or attach the meeting where it was agreed.

Your brief Examples ▾

e.g. Payments engineering change-control SOP:

- No production deployment or database migration without an approved change ticket.
- During declared incidents, the on-call engineer may act without a ticket, recording actions on the incident bridge.
- Destructive commands (DROP TABLE, rm -rf, kubectl delete namespace) are strictly forbidden on production clusters.
- All access to customer secrets or payment credentials must require an active session.

One rule per line. Anything we infer is marked as inferred, on the line itself.

☐ Write what the agent does, not what it intends

- Name the exact action: command, path, domain or tool (e.g. "Block Bash when command.program equals rm").
- Say the outcome: block it, or ask a human before it runs.
- Phrase exceptions as a fact the agent shows: "unless agent.environment equals development".

This policy will be created in Crispy Pay. Turn it into a policy →

L'assistant de création d'une politique d'intention : Brief, Policy, How it runs, Ratify.

Si l'assistant indique que la rédaction n'est pas disponible, un administrateur de votre site OpenLATCH n'a pas encore configuré la rédaction assistée par IA.

La voir enregistrer

- 1 La ratification reconstruit le bundle de politiques. Le client le récupère à sa prochaine interrogation ; pour le récupérer tout de suite, lancez `openlatch system aip refresh` sur la machine de l'agent.
- 2 Demandez à l'agent de récupérer une page sur pastebin.com.
- 3 La politique s'exécute en Monitor : l'action se poursuit et une anomalie est enregistrée. Ouvrez **Anomalies** dans la console pour la lire : l'anomalie indique la politique, sa dimension et la zone dont elle provient.

Ce que vous avez fait

Vous avez écrit une politique d'intention avec vos propres mots, vous l'avez vérifiée sur des cas concrets et vous l'avez exécutée en Monitor, où elle enregistre sans jamais arrêter l'agent. Quand les anomalies montrent que la politique est juste, passez-la en Enforce dans sa zone : le client applique alors son verdict avant l'exécution de l'action. [Modes](#) explique la différence, et [Politiques d'intention](#) explique ce qu'OpenLatch a construit à partir de votre phrase.

Se connecter à la console

Connectez-vous à la console sur `app.openlatch.ai` pour définir des Autonomy Zones, écrire des politiques d'intention et lire ce qu'ont fait vos agents.

Se connecter

- 1 Ouvrez `app.openlatch.ai`. La page de connexion propose les méthodes qu'autorise votre organisation.
- 2 Connectez-vous avec l'une d'elles : l'authentification unique SAML de votre organisation, un compte Google, Microsoft, GitHub, Apple ou LinkedIn, un lien magique envoyé par e-mail, une clé d'accès, ou une adresse e-mail et un mot de passe.
- 3 Si l'authentification à deux facteurs est activée sur votre compte, saisissez le code de votre application d'authentification.

La première fois que vous vous connectez sans invitation, la console vous guide dans la création de votre organisation. Quand un collègue vous a invité, le lien d'invitation vous mène directement à votre organisation.

Relier le client à votre compte

Quand `openlatch init` ou `openlatch system auth login` a besoin d'un identifiant, il ouvre la console dans votre navigateur. Une fois connecté, la console remet au terminal un code à usage unique, jamais une clé. Le client liste vos organisations avec ce code, puis seulement crée une clé d'API, dans l'organisation choisie, et l'enregistre dans le trousseau du système d'exploitation.

Vos organisations	Ce qui se passe
Une seule	Elle est choisie pour vous, et le terminal affiche son nom
Plusieurs, dans un terminal	Le terminal demande laquelle. <code>--org <ORG></code> ou <code>OPENLATCH_ORG</code> la nomme à l'avance, par son slug, son nom ou son identifiant
Plusieurs, sans personne à qui demander (<code>--yes</code> , <code>--json</code> , <code>--quiet</code> ou pas de terminal)	<code>openlatch system auth login</code> s'arrête sur OL-1608 et les liste. <code>openlatch init</code> sans clé d'API ne lance jamais de connexion dans le navigateur ; il s'arrête donc plus tôt, sur OL-1600 : définissez <code>OPENLATCH_API_KEY</code> , ou exécutez <code>init</code> dans un terminal
Une machine reste dans l'organisation où elle est inscrite. <code>openlatch system auth switch --org <ORG></code> la déplace vers une autre de vos organisations.	

Les pages de connexion

Page	À quoi elle sert
<code>/login</code>	Se connecter avec toute méthode qu'autorise votre organisation
<code>/register</code> · <code>/signup</code>	Créer un compte, depuis une invitation ou de vous-même ; <code>/signup</code> mène à <code>/register</code>
<code>/forgot-password</code>	Demander un lien de réinitialisation du mot de passe par e-mail
<code>/reset-password</code>	Choisir un nouveau mot de passe depuis le lien de réinitialisation
<code>/two-factor-verify</code>	Saisir le code de votre application d'authentification
<code>/change-password</code>	Choisir un nouveau mot de passe lorsque votre compte doit le changer avant de continuer
<code>/welcome</code>	Créer votre organisation après votre première connexion
<code>/cli-auth</code>	La page qu'ouvre le client pour recevoir sa connexion ; l'organisation se choisit dans le terminal
<code>/activate</code>	Un écran de configuration unique qu'affiche un nouveau site OpenLatch avant que sa première organisation existe. Sur <code>app.openlatch.ai</code> , vous ne le voyez jamais

Qu'est-ce qu'OpenLatch

Lisez cette page pour comprendre ce que fait OpenLatch avant de l'installer.

OpenLatch est la couche de contrôle d'exécution des agents d'IA en entreprise. Il vous permet de donner plus d'autonomie aux agents sans perdre le contrôle : vous définissez le terrain sur lequel joue chaque agent, et chaque action couverte qu'il entreprend est décidée avant son exécution.

Le problème

Les agents d'IA ne se contentent pas de répondre. Ils exécutent des commandes, modifient des fichiers, appellent des outils et dépensent de l'argent, souvent avec des droits étendus sur la machine où ils s'exécutent. Les instructions d'un prompt orientent ce qu'un agent compte faire, mais elles ne peuvent pas garantir ce qu'il fait : un contexte long, un plan confus ou une instruction injectée peuvent encore mener à une action que personne ne voulait.

L'approche

Principe	Ce que cela signifie
Dans l'agent, pas un proxy devant lui	Le client se branche sur les hooks natifs de l'agent : il voit chaque action tant qu'il est encore temps d'en changer l'issue
Décidé avant l'impact	Chaque action couverte atteint le point de décision avant de s'exécuter, et le verdict est appliqué. Un verdict est déterministe parce qu'une politique explicite l'a produit
Autorité locale	Le client décide sur la machine, à partir d'un bundle de politiques résident. Il continue de décider réseau débranché, et l'agent n'attend jamais le réseau pour un verdict
Votre intention, avec vos mots	Vous écrivez une politique sous forme de phrase. OpenLatch la transforme en Agent Intent Policy et vous montre des cas de test concrets avant que vous la ratifiiez
Des preuves après coup	Le client envoie à la plateforme l'historique de chaque action et de son verdict après la décision, pour l'audit et la revue

Le modèle de menace est un agent qui se comporte mal, pas un opérateur malveillant : OpenLatch contrôle ce que fait un agent autorisé à travers ses propres hooks. Ce n'est pas un bac à sable impossible à contourner.

Les composants

Composant	Où il s'exécute	Ce qu'il fait
Le client	À côté de chaque agent, sur un poste de développement ou partout où l'agent s'exécute	Raccorde les hooks de l'agent, détient le bundle de politiques, décide de chaque action couverte et l'enregistre
La plateforme	<code>app.openlatch.ai</code>	Détient le registre des agents, les Autonomy Zones et leurs politiques d'intention, compile et distribue les bundles de politiques, et conserve les preuves

Étapes suivantes

- [Installez le client.](#)
- Lisez [Comment fonctionne OpenLatch](#) pour suivre une action de bout en bout.
- Voyez quels agents sont raccordés, plan par plan, dans [Niveaux de contrôle](#).

FAQ

Trouvez ici des réponses rapides, chacune avec un lien vers la page qui traite le sujet en entier.

OpenLatch est-il gratuit ?

La tarification de la plateforme est en cours de définition avec nos partenaires de conception, et les pilotes avec ces partenaires sont gratuits.

Ai-je besoin de Node.js ?

Seulement pour `npx @openlatch/client init`, car `npx` fait partie de Node.js. Le client est un binaire natif et ne s'exécute pas sur Node.js. Sur une machine sans Node.js, lancez le script de votre plateforme ; il n'a besoin de rien d'autre que ce que fournit le système d'exploitation :

...

```
curl -Lsf https://openlatch.ai/install.sh | sh
```

...

```
powershell -ExecutionPolicy Bypass -c "irm https://openlatch.ai/install.ps1 | iex"
```

Voir [Installer le client](#).

Quels agents OpenLatch couvre-t-il ?

Claude Code, Codex CLI, Cursor et Cline sont raccordés aujourd'hui. La couverture s'indique par agent et par plan : [Niveaux de contrôle](#) montre quel plan est Enforced, Monitored ou Observed pour chacun.

Quelles données le client envoie-t-il à la plateforme ?

Après chaque décision, le client envoie l'historique de l'action et de son verdict : ce que l'agent a tenté de faire, comme une commande, un chemin de fichier ou une destination réseau, et ce qui a été décidé. Les identifiants sont masqués sur la machine avant que quoi que ce soit ne la quitte. Voir [Confidentialité et télémétrie](#).

Puis-je utiliser OpenLatch hors ligne ?

Oui. Le client décide sur la machine à partir de son bundle de politiques résident, qui continue de décider réseau débranché. Récupérer de nouvelles politiques et envoyer l'historique à la plateforme nécessitent une connexion ; décider, non.

OpenLatch ralentit-il mon agent ?

Le client décide sur la machine, à partir d'un bundle de politiques qu'il garde en mémoire : l'agent n'attend jamais le réseau pour un verdict. L'historique part vers la plateforme ensuite.

Puis-je héberger la plateforme moi-même ?

Pas aujourd'hui. La plateforme est un service géré, sur `app.openlatch.ai`.

Comment désinstaller le client ?

```
openlatch uninstall
```

Cette commande retire les hooks des agents et arrête le client. Ajoutez `--purge` pour supprimer aussi l'état du client, dont le dossier `~/ .openlatch` et l'identifiant enregistré dans votre trousseau.

Aucune de ces commandes ne supprime les binaires : une purge ne désinstalle pas les fichiers que l'installation a placés. Si vous avez installé avec un script, `openlatch` et `openlatch-hook` restent dans `~/ .local/bin` ou `%LOCALAPPDATA%\openlatch\bin`, tout comme l'entrée du PATH. Supprimez-les à la main si vous voulez retrouver la machine exactement comme avant.

Où signaler un bug ou demander une fonctionnalité ?

Écrivez-nous à hello@openlatch.ai.

Concepts

Ces pages expliquent les idées que vous retrouvez partout dans OpenLatch : l'Autonomy Zone, les politiques d'intention, les verdicts, les modes et les anomalies. Lisez-les pour comprendre pourquoi le produit se comporte comme il le fait ; les guides vous disent où cliquer.

OpenLatch fait tourner une boucle de contrôle continue autour de chaque agent. Le contrôle existe pour qu'il soit plus facile de dire oui : plus le terrain de jeu d'un agent est clair, plus on peut lui confier de choses à faire seul.

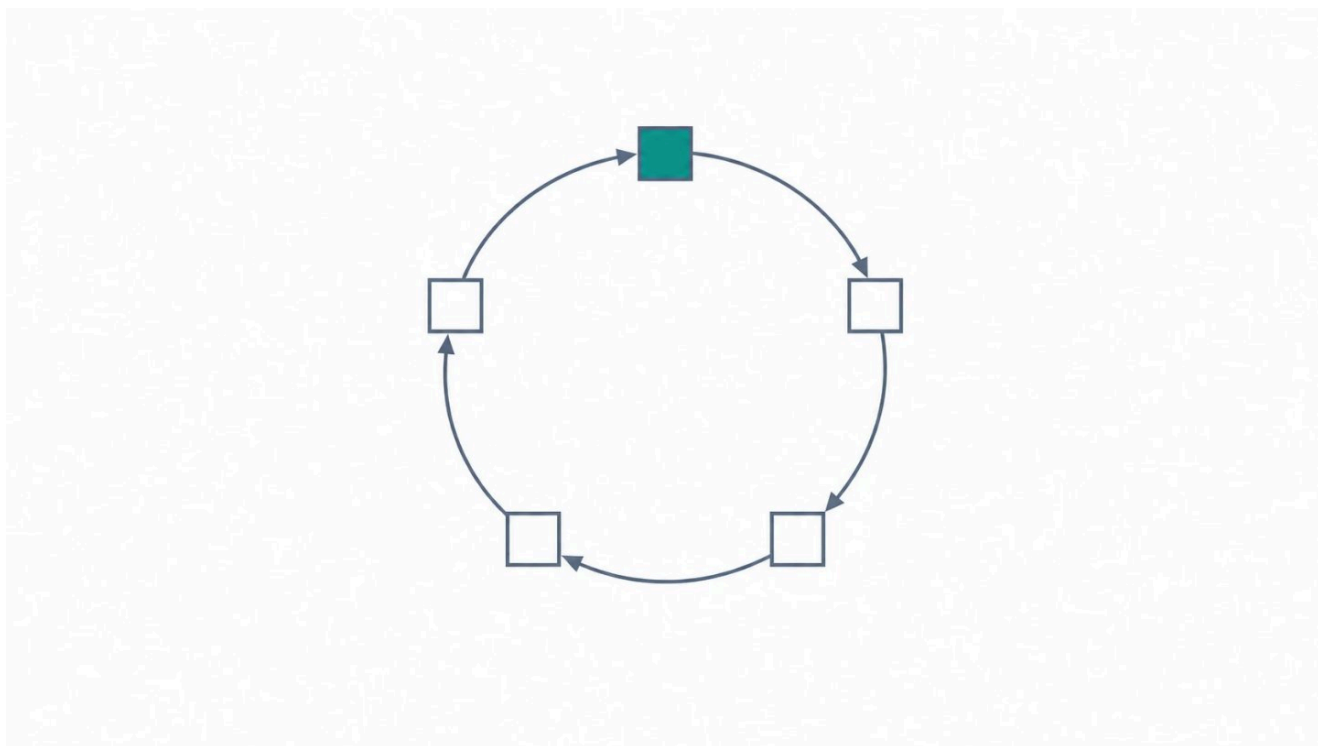


Figure 1. La boucle de contrôle : Identifier, Définir, Détecter, Agir et Améliorer, autour de chaque agent.

Étape	Ce qui se passe	À lire
Identifier	L'agent est relié au contexte de votre entreprise : son identité, son propriétaire, son unité, son type et son environnement, et le niveau de contrôle qu'OpenLatch a sur lui	Niveaux de contrôle
Définir	Chaque agent se résout en une Autonomy Zone effective, construite à partir de politiques d'intention sur cinq dimensions	Autonomy Zone · Dimensions · Politiques d'intention
Détecter	Ce que fait l'agent est capturé sous forme de graphe d'exécution et confronté à sa zone. Une action qui enfreint une politique d'intention est une anomalie	Graphe d'exécution · Anomalies
Agir	Avant l'impact, chaque action couverte reçoit un verdict : Allow, Ask, Block ou Optimize	Verdicts · Modes
Améliorer	L'historique de chaque décision montre quelles politiques sont justes, pour que vous puissiez les faire passer de Monitor à Enforce. OpenLatch n'élargit jamais de lui-même l'autorité d'un agent	Modes

Pour suivre une action, du hook de l'agent jusqu'au verdict, lisez [Comment fonctionne OpenLatch](#).

Autonomy Zone

L'Autonomy Zone d'un agent est le périmètre d'action contextuel dans lequel il agit : ce qu'il cherche à accomplir, les conditions dans lesquelles il peut agir et les conséquences que l'entreprise accepte. À l'intérieur de la zone, l'agent est libre de poursuivre sa mission. Chaque action couverte qui franchirait la limite est décidée avant l'impact.



Figure 1. Une Autonomy Zone est le périmètre d'action d'un agent : ce qu'il peut faire, dans quelles conditions, avec quelles conséquences acceptées.

L'agent apporte la mission ; la zone apporte les règles. Pensez à un terrain de basket : les lignes et les règles du jeu sont fixes, le joueur se déplace librement à l'intérieur, et l'arbitre siffle ce qui franchit une ligne. La zone est le terrain, ses politiques d'intention sont les règles, et le verdict est le coup de sifflet.

Ce que contient une zone

Une zone contient des politiques d'intention réparties sur cinq dimensions : Qualité, Économie, Sécurité, Sûreté et Conformité. [Dimensions](#) explique ce que couvre chacune. Chaque politique s'exécute en mode Disabled, Monitor ou Enforce dans cette zone, comme l'explique [Modes](#).

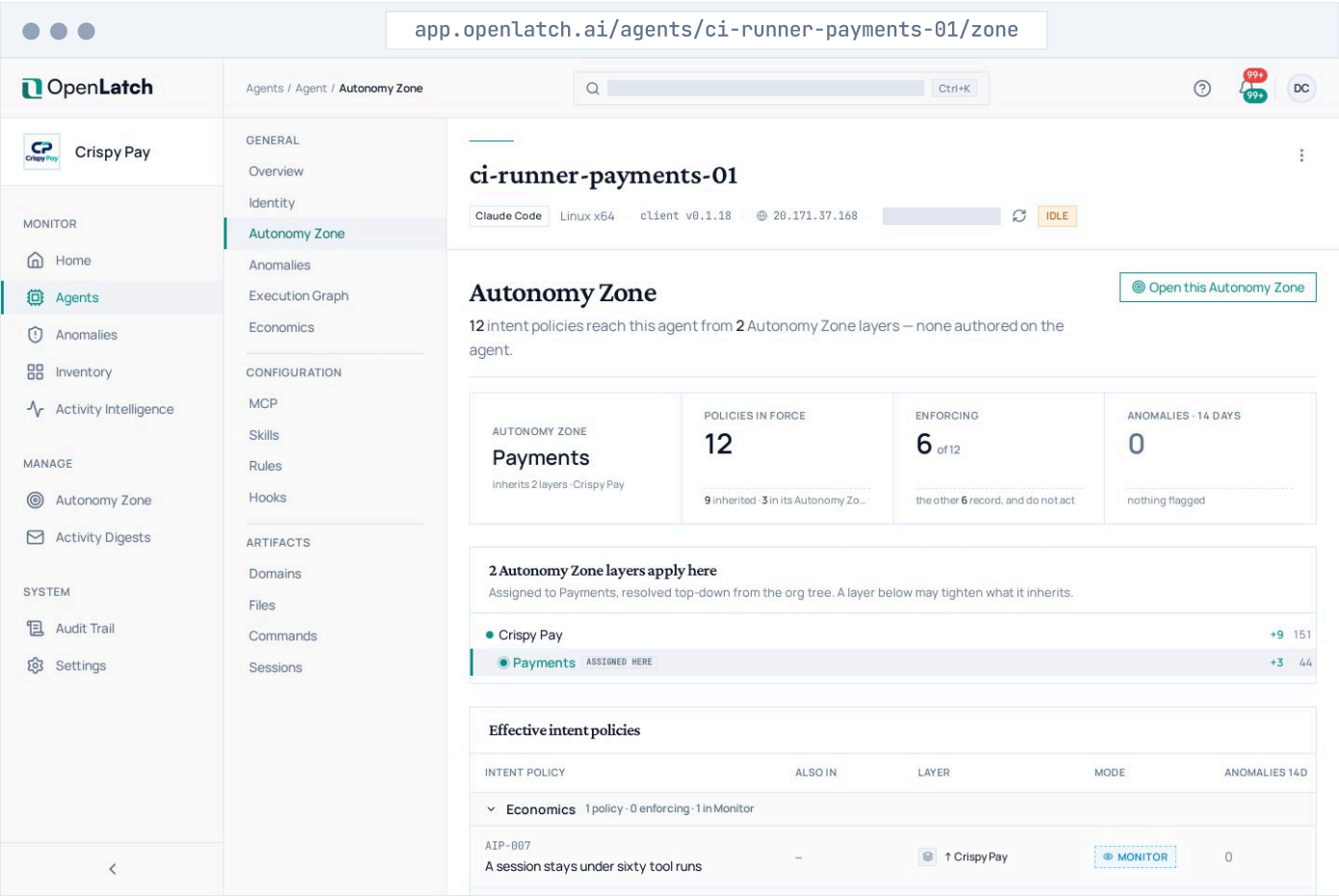
Les zones suivent votre organisation

Les zones se placent sur l'arborescence des unités de votre organisation, en général synchronisée depuis votre annuaire. Vous n'entretenez pas de seconde hiérarchie.

Règle	Ce que cela signifie
Une zone vit sur une unité	Chaque unité peut porter une zone, et la plupart n'en portent pas
L'héritage n'est jamais facultatif	Une unité sans zone hérite de la zone de l'unité la plus proche au-dessus d'elle qui en a une. Chaque zone hérite de toutes les zones au-dessus d'elle
Ajouter, jamais assouplir	Une zone inférieure peut ajouter des politiques, jamais retirer ce dont elle hérite
Un agent, une zone	Chaque agent est rattaché à exactement une zone : par défaut la zone la plus proche au niveau ou au-dessus de l'unité de son propriétaire, ou celle qu'un utilisateur choisit à la main
Pas de ciblage caché	Une politique ne peut pas s'appliquer à certains agents de sa zone et pas à d'autres. Si elle ne doit pas atteindre un agent, déplacez l'agent ou écrivez la politique plus bas dans l'arborescence

La zone effective

Ce sous quoi un agent opère réellement est sa **zone effective** : toutes les politiques de la zone à laquelle il est rattaché et de chaque zone au-dessus, regroupées par dimension. Chaque politique indique la zone dont elle provient. Quand une même politique est portée à plusieurs niveaux sur le chemin d'un agent, elle l'atteint une seule fois, dans le plus strict de ses modes.



L'onglet Autonomy Zone d'un agent : sa zone et son chemin, les politiques en vigueur et la zone dont provient chacune.

Ce qu'une zone n'est pas

Pas	Parce que
Une liste figée de garde-fous	La même action peut recevoir un verdict différent pour un autre agent, une autre mission ou un autre environnement
Des instructions dans le prompt	Un prompt oriente le comportement et peut être oublié. La zone s'applique à la frontière d'exécution de l'agent, hors de sa fenêtre de contexte
Une liste de permissions	Les identités et permissions existantes alimentent la zone, qui juge l'action en contexte
Un bac à sable impossible à contourner	Le modèle de menace est un agent qui se comporte mal, pas un opérateur malveillant

Dimensions

Une Autonomy Zone est délimitée par cinq dimensions. Chacune répond à une question sur l'action d'un agent, et chacune est peuplée de politiques d'intention.

Dimension	La question à laquelle elle répond	Exemples de politiques d'intention
Qualité	Le travail est-il au niveau attendu ?	Le code est testé avant d'être livré. Les résultats atteignent le niveau d'exactitude et d'exhaustivité requis. Le processus imposé est respecté
Économie	Le coût est-il justifié ?	Budgets et plafonds de tokens par période. Pas de boucles incontrôlées ni de tentatives inutiles. Le modèle correspond à la tâche
Sécurité	L'agent peut-il toucher à cela ?	Aucun identifiant ne quitte le périmètre de l'entreprise. Seuls les outils, serveurs MCP et skills approuvés. L'accès aux données reste dans les limites autorisées
Sûreté	La conséquence est-elle acceptable ?	Pas d'accès à la production sans approbation. Aucune opération irréversible ou destructrice. Le rayon d'impact et l'impact sur les clients restent dans ce que l'entreprise accepte
Conformité	Est-ce légal et conforme aux règles ?	Contraintes de protection des données pour les agents qui servent des clients sur des marchés réglementés. Politiques et normes internes. Obligations contractuelles

Une politique peut couvrir plusieurs dimensions

Une politique d'intention se compose d'un ou de plusieurs atomes d'intention, et chaque atome appartient à une dimension. Une politique porte les dimensions de tous ses atomes : une même politique peut donc relever à la fois de la Sécurité et de la Sûreté. La console regroupe les politiques effectives d'un agent par dimension, et chaque anomalie indique la dimension de la politique qu'elle a enfreinte.

Ce qui n'est pas une dimension

- **La mission.** Ce à quoi sert un agent appartient à l'agent, pas à la zone. L'agent apporte la mission ; la zone apporte les règles.
- **Les preuves d'audit.** Les règles de conformité s'exécutent dans la zone, avant l'action. La preuve de ce qui s'est passé, de la politique appliquée, du verdict rendu et de son application est l'historique qu'OpenLatch conserve pour chaque décision, quelle que soit la dimension.

Politiques d'intention

Une politique d'intention exprime ce que vous voulez dire, plutôt que chaque surface qu'un agent pourrait utiliser pour le faire. Dans OpenLatch, vous l'écrivez sous forme d'**Agent Intent Policy (AIP)**, et OpenLatch la compile en règles déterministes que le client applique avant chaque action couverte.

De vos mots à une AIP

Vous partez d'un brief : une phrase, un document de politique collé ou une conversation enregistrée. OpenLatch le découpe en **atomes d'intention**, un par règle, et génère deux choses à partir de chaque atome : une phrase lisible et des cas de test concrets.

Champ de l'atome	Ce qu'il contient	Exemple
who	Les agents concernés par la règle	tout agent
does what	L'action	déployer
to what	La cible	un namespace de production
only if	La condition qui l'autorise	un ticket de changement approuvé est lié
unless	L'exception	une cellule de crise d'incident est active
class	Si l'atome peut être appliqué : applicable, observable seulement ou hors périmètre	applicable

Vous ne remplissez jamais ces champs. OpenLatch les extrait de votre brief, et vous les modifiez en éditant la phrase.

Vous ratifiez la phrase

Pour l'atome ci-dessus, vous lisez cette phrase :

Tant qu'aucune cellule de crise d'incident n'est active, lorsqu'un agent demande un déploiement en production, le système doit exiger un ticket de changement approuvé.

À côté, les cas de test montrent ce que la politique déciderait dans des situations concrètes, par exemple un déploiement avec ticket, un sans ticket, et un sans ticket pendant un incident. Les cas sont là pour que votre réponse soit éclairée. Vous répondez Oui ou Non sur la phrase :

- **Oui** accepte l'atome.
- **Non** ouvre une note où vous dites avec vos propres mots ce que vous vouliez dire. OpenLatch modifie les champs, régénère la phrase et ses cas, et signale les cas dont le verdict a changé. L'atome revient pour un nouveau Oui ou Non, car une réponse donnée sur l'ancienne phrase ne couvre pas la nouvelle.

Vous ne voyez ni n'approuvez jamais les règles compilées. Vous approuvez ce que vous pouvez lire.

D'une AIP au client

Une fois ses atomes acceptés, vous ratifiez l'AIP dans zéro, une ou plusieurs zones, chacune dans son propre mode. La plateforme compile les atomes en règles et construit un bundle de politiques par agent à partir de sa zone effective. Le bundle est vérifié en intégrité, et le client le garde sur la machine : il continue donc de décider réseau débranché.

La classe d'un atome compte à l'exécution : un atome `monitor-only` reste en Monitor même quand sa politique s'exécute en Enforce, parce qu'il peut être observé mais pas appliqué.

Le cycle de vie d'une politique

1. Écrivez un brief ou apportez un document source.
2. Relisez les phrases générées et leurs cas, puis acceptez-les.
3. Rejouez la politique sur l'activité récente de vos agents pour voir ce qu'elle aurait décidé.
4. Ratifiez-la en Disabled, Monitor ou Enforce.
5. Exécutez-la en Monitor, lisez ses anomalies, et passez-la en Enforce quand elle est juste.

[Écrire votre première politique](#) parcourt ces étapes dans la console.

Verdicts

Avant l'exécution d'une action couverte, le client rend l'un de quatre verdicts : Allow, Ask, Block ou Optimize. Un verdict est déterministe parce qu'une politique explicite l'a produit.

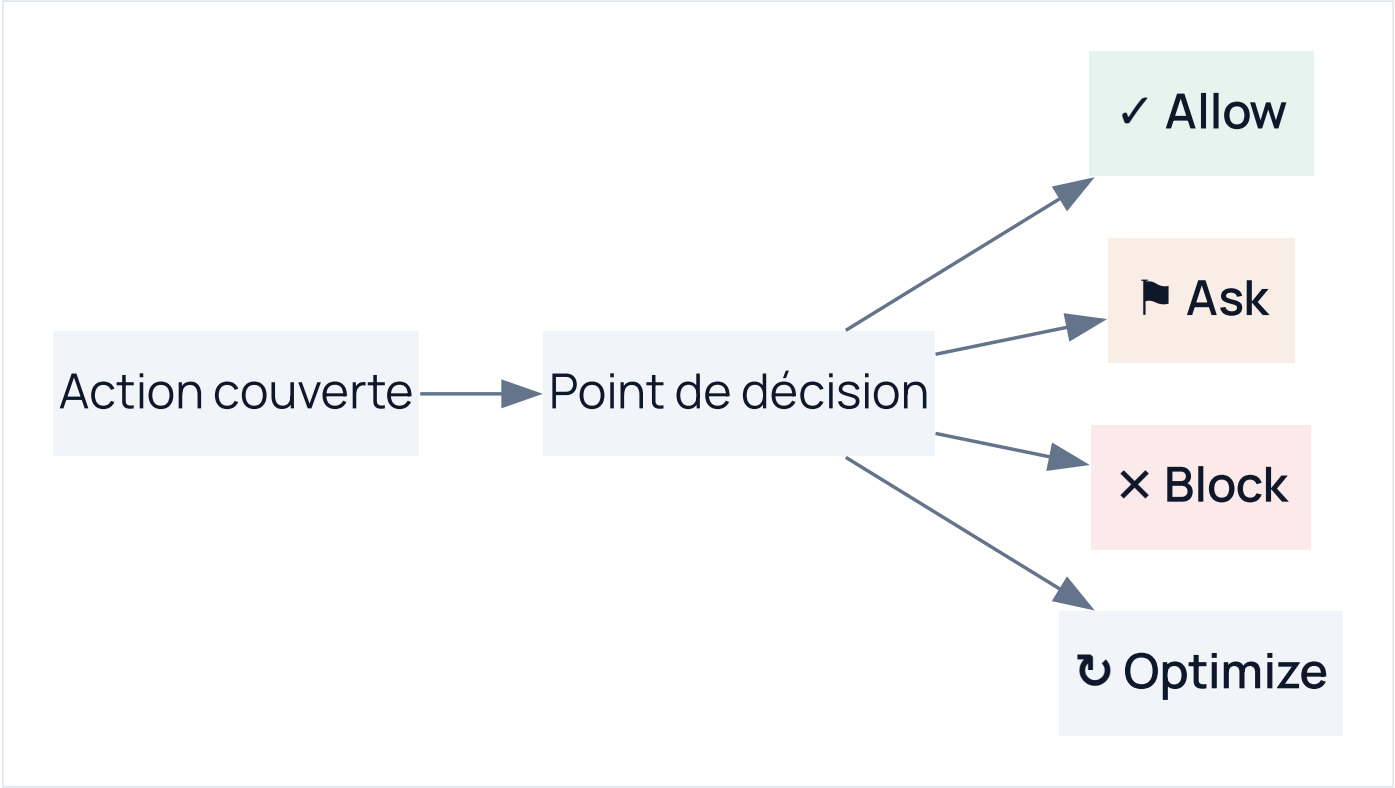


Figure 1. Chaque action couverte atteint le point de décision avant de s'exécuter. Les politiques d'intention de la zone effective produisent un verdict déterministe.

Verdict	Ce qui se passe
Allow	L'action s'exécute. En Monitor, elle peut s'exécuter et être signalée
Ask	Par défaut, l'action s'exécute et est signalée pour une revue asynchrone ; elle n'est retenue que lorsqu'un atome l'active pour une action irréversible
Block	L'action ne s'exécute pas. L'agent reçoit un refus natif qui nomme la règle qui l'a produit
Optimize	Le chemin change. En Enforce, un levier agit ; en Monitor, il mesure ce qu'il aurait fait

Quand plusieurs politiques correspondent à une même action, le verdict le plus restrictif l'emporte, dans cet ordre : Block, puis Ask, puis Optimize, puis Allow.

Ask

Ask laisse l'action s'exécuter et la signale pour revue, afin qu'une personne décide si l'agent doit recommencer. Il empêche la récurrence, pas la première occurrence. Un atome peut choisir de retenir une action irréversible jusqu'à ce qu'on y réponde ; rien d'autre ne dépend d'une personne.

La façon dont Ask atteint l'agent dépend de l'agent. Claude Code le reçoit sous la forme de sa propre demande native, avec la raison de la règle. Codex CLI n'a pas de demande native : il reçoit un message qui porte la raison, et l'appel se poursuit.

Optimize

Optimize change le chemin d'une action au lieu de l'arrêter. Chaque changement passe par un levier, et ce que fait un levier dépend du mode de la politique.

Levier	Statut
loop_stop	Agit en Enforce ; mesure en Monitor
steer	Agit en Enforce ; mesure en Monitor
narrow_output	Agit en Enforce sur Claude Code ; mesure en Monitor
effort_clamp	Mesure par défaut, en Monitor comme en Enforce ; agit en Enforce seulement là où vous l'activez
context_edit	Mesure par défaut, en Monitor comme en Enforce ; agit en Enforce seulement là où vous l'activez
prefix_guard	Mesure par défaut, en Monitor comme en Enforce ; agit en Enforce seulement là où vous l'activez
substitute	Ce levier n'est pas disponible, et en Monitor il mesure seulement

En Monitor, un levier mesure seulement ce qu'il aurait fait, et l'historique l'indique. En Enforce, il est appliqué.

Une action, quatre verdicts

Scénario illustratif. La même action, `git push`, peut recevoir chaque verdict selon son contexte :

Contexte	Verdict
Un agent pousse une branche de fonctionnalité d'un dépôt que son équipe possède	Allow
Un agent pousse sur une branche de release partagée	Ask : le push s'exécute et est signalé pour revue
Un agent force un push sur la branche principale d'un dépôt de production	Block
Un agent relance en boucle le même push qui échoue	Optimize : en Enforce, <code>loop_stop</code> met fin à la boucle

 Note

Ask empêche la récurrence, pas la première occurrence. Sur Codex CLI, il apparaît sous forme de message et l'appel se poursuit.

Modes

Chaque politique d'intention s'exécute dans l'un de trois modes, dans chaque zone qui la porte. Le mode décide si une correspondance est seulement enregistrée ou si son verdict est appliqué avant l'exécution de l'action.

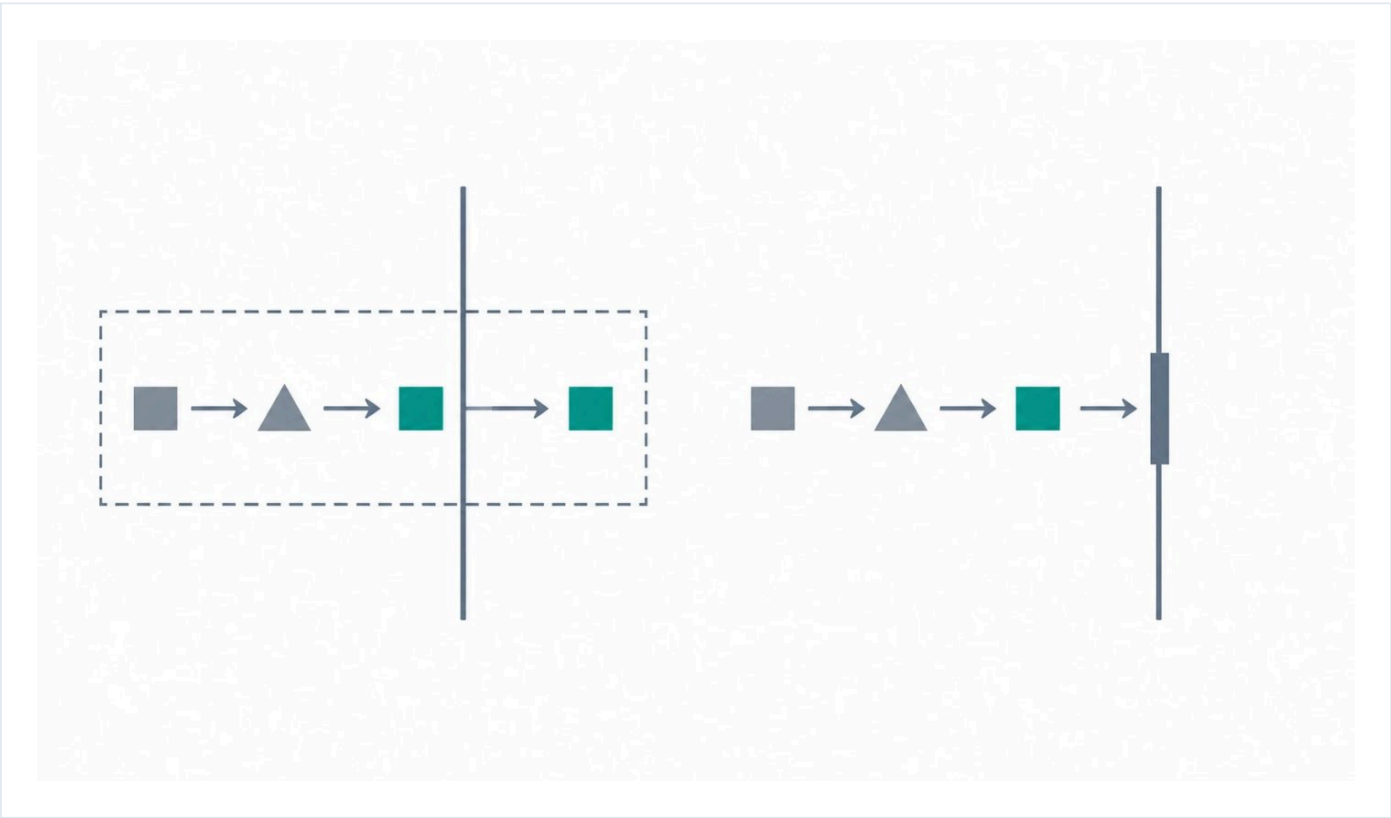


Figure 1. Les mêmes actions en Monitor et en Enforce : en Monitor, elles franchissent la limite et sont enregistrées ; en Enforce, le verdict s'applique d'abord.

Mode	Ce qu'il fait
Disabled	La politique est conservée, avec son historique, mais rien n'est évalué
Monitor	Une correspondance est enregistrée et signalée, et l'action se poursuit
Enforce	Le client applique le verdict de la politique avant l'exécution de l'action, et enregistre s'il l'a fait

Le mode appartient à la politique dans une zone, jamais à un atome isolé. Une exception est inscrite dans l'atome lui-même : un atome dont la classe est `monitor-only` peut être observé mais pas appliqué, il reste donc en Monitor même quand sa politique s'exécute en Enforce.

Mériter Enforce sur des preuves

Lancez une nouvelle politique en Monitor. Elle enregistre chaque action qu'elle aurait décidée, sans changer ce que fait l'agent. Lisez ses anomalies : quand elles montrent que la politique dit ce que vous vouliez dire, passez-la en Enforce. Quand elles montrent le contraire, modifiez-la ou désactivez-la.

OpenLatch n'élargit jamais de lui-même l'autorité d'un agent. Chaque passage de Monitor à Enforce, et chaque nouvelle politique, est un choix fait par quelqu'un de votre organisation.

Une politique, plusieurs zones

Une même politique peut se trouver dans plusieurs zones, chacune avec son propre mode, par exemple Monitor dans une équipe et Enforce dans une autre, sans copie à tenir à jour.

Règle	Ce que cela signifie
Le plus strict l'emporte	Quand le chemin d'un agent porte plusieurs fois la même politique, elle s'exécute une seule fois, dans le mode le plus strict : Enforce avant Monitor, avant Disabled
Une zone ne fait que resserrer	Une politique n'est activée dans une zone que si elle n'assouplit pas les zones au-dessus. La garder en Disabled est toujours possible
Abaissier n'est jamais bloqué	Faire passer une politique d'Enforce à Monitor, ou à Disabled, est toujours possible
Une seule version partout	Chaque zone exécute la version en vigueur de la politique, chacune dans son propre mode

Anomalies

Une **anomalie** est une action d'agent dont OpenLatch détermine qu'elle enfreint une politique d'intention de l'une des dimensions de l'Autonomy Zone de l'agent. C'est la seule chose que signale OpenLatch.

Chaque anomalie indique la politique enfreinte, la dimension de cette politique et la zone dont provient la politique. Elle s'explique par construction : une règle compilée à partir d'une politique que votre organisation a ratifiée l'a produite, et rien d'autre.

Ce que fait une anomalie dans chaque mode

Mode de la politique	Ce qui se passe
Disabled	Rien n'est évalué, donc aucune anomalie n'est levée
Monitor	L'anomalie est enregistrée et signalée, et l'action se poursuit
Enforce	L'action reçoit son verdict avant de s'exécuter, et l'anomalie l'enregistre

Où les voir

La page **Anomalies** de la console liste toutes les anomalies de votre organisation. Chaque agent a aussi un onglet **Anomalies** avec uniquement les siennes.

app.openlatch.ai/anomalies

OpenLatch

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

Anomalies

Ctrl+K

?

99+

99+

DC

Anomalies

Review actions that crossed an intent policy, then archive the evidence when triage is complete.

Export to CSV

Waiting on your answer

All

WAITING ON YOUR ANSWER

0

nothing waiting

ANOMALIES - 7 DAYS

26

across 10 agents

BLOCKED

50

ALLOWED AND FLAGGED

50

Nothing is waiting on your answer

Every flagged action has an answer against it, so the policy knows what to do next time.

La page Anomalies, qui liste les actions ayant enfreint une politique d'intention, avec la politique, sa dimension et le verdict.

Ce que les anomalies ne sont pas

Les anomalies proviennent uniquement des politiques d'intention. OpenLatch ne signale pas une action parce qu'elle diffère de ce que fait habituellement un agent : si aucune politique ratifiée ne couvre une action, il ne lève aucune anomalie. Pour signaler quelque chose de nouveau, écrivez la politique d'intention qui le dit.

Graphe d'exécution

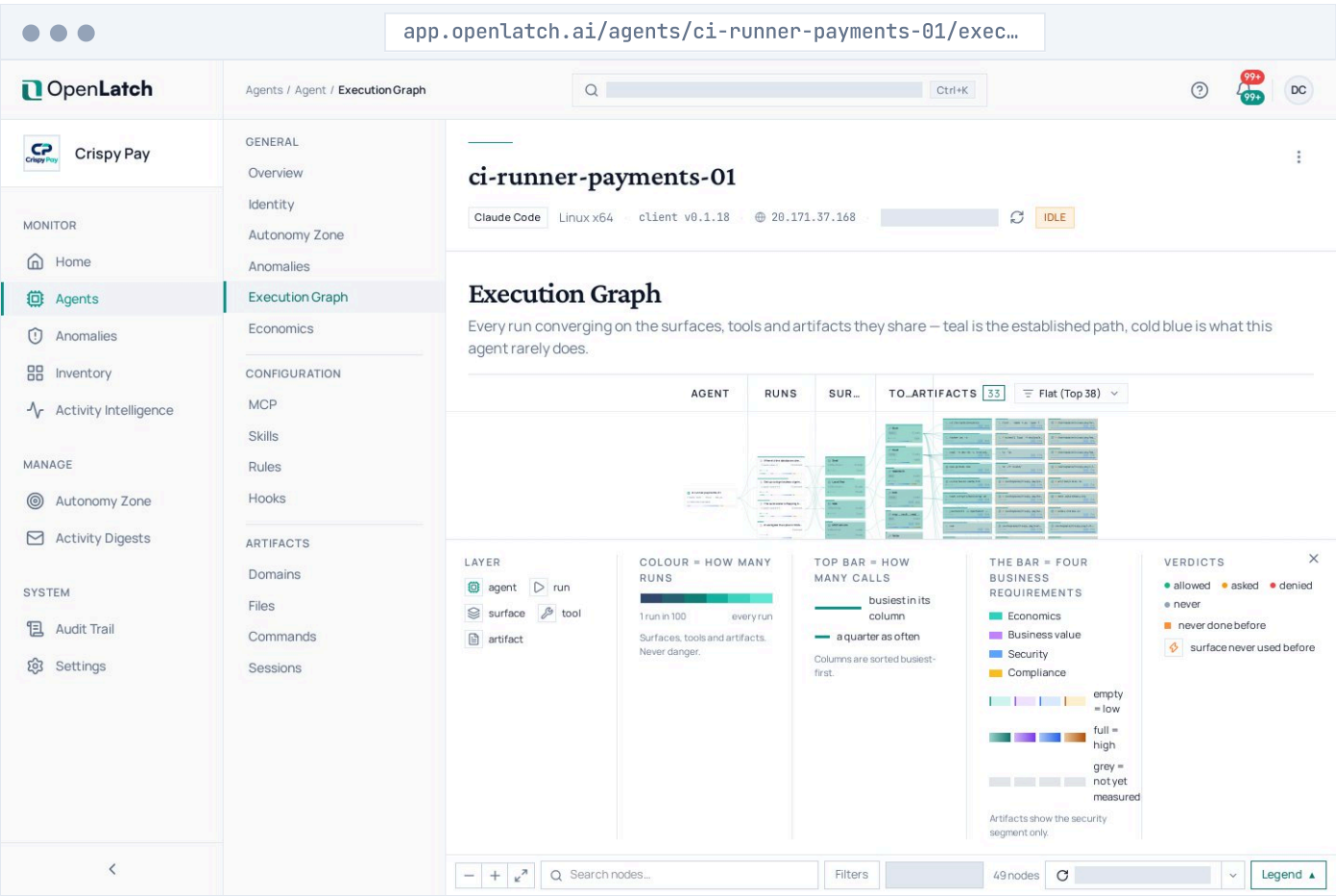
Le graphe d'exécution est l'historique de ce qu'un agent a réellement fait. Il est construit à partir des événements que le client envoie après chaque décision, et c'est lui qu'OpenLatch confronte à l'Autonomy Zone de l'agent.

Ce qu'il montre

Chaque agent a un onglet **Execution Graph**. Il se lit de gauche à droite, de l'agent jusqu'à ce qu'il a touché :

Couloir	Ce qu'il contient
Agent	L'agent concerné par le graphe
Exécutions	Ses sessions, une carte par exécution
Surfaces	Ce vers quoi il s'est tourné, déduit du nom des outils
Outils	Ce qu'il a invoqué
Artefacts	Ce qu'il a touché ou lancé : fichiers, domaines réseau et commandes

Les couloirs convergent vers ce que les exécutions partagent : vous voyez les chemins qu'un agent emprunte le plus et ceux qu'il emprunte rarement.



L'onglet Execution Graph d'un agent : exécutions, surfaces, outils et artefacts, de gauche à droite.

Deux plans l'alimentent

Plan	Ce que capture le client
Plan de commande	Chaque appel d'outil et chaque commande shell, interceptés au hook natif de l'agent avant leur exécution
Plan de requête	Chaque requête sortante vers un modèle, interceptée par un écouteur sur la même machine, avec son modèle, ses tokens et son coût

Les autres onglets d'un agent découpent le même historique par type : commandes, fichiers, domaines, serveurs MCP, skills, hooks, sessions et économie.

D'une action à une politique

Quand le graphe montre une commande que vous voulez encadrer, ouvrez-la et lancez une nouvelle politique d'intention à partir de cette action. Le brief part de ce que l'agent a réellement lancé, et la politique passe par la même relecture et la même ratification que toute autre. [Politiques d'intention](#) explique ce chemin.

Une action qu'aucune politique n'a décidée est enregistrée comme autorisée par défaut, et non comme permise par une décision. Le graphe distingue les deux.

Niveaux de contrôle

Chaque agent du registre des agents porte un niveau de contrôle. Il vous dit ce qu’OpenLatch peut réellement faire avec cet agent, pour que vous n’ayez jamais à supposer qu’un agent est contrôlé parce qu’il figure dans la liste.

Niveau de contrôle	Ce que cela signifie
Enforced	Le client se trouve dans le chemin d'exécution de l'agent. Un verdict atteint l'agent et il est respecté
Monitored	L'intégration est installée et les événements circulent, mais un verdict ne peut pas encore être transmis : rien n'est empêché
Known	L'agent est identifié et classé, et les limites du contrôle sont indiquées

La couverture est un résultat de déploiement. Le niveau de contrôle est ce que le déploiement a obtenu pour cet agent, pas une promesse sur tous les agents de son type.

Deux plans

Un agent est couvert sur deux plans, et le niveau de contrôle s'indique toujours par plan.

Plan	Ce qu'il intercepte	Ce qu'il peut porter aujourd'hui
Plan de commande	Un appel d'outil ou une commande shell, intercepté au hook natif de l'agent	Allow, Ask, Block ou Optimize, décidé avant l'exécution de l'action. Optimize commence en Monitor
Plan de requête	Une requête sortante vers un modèle, interceptée par un écouteur sur la même machine	Allow et enregistrement par défaut : il est donc Observed, c'est-à-dire capturé et mesuré. Ask n'y a pas de sens, et Block n'est pas disponible sur ce plan

Agents raccordés aujourd’hui

Agent	Distribution	Plan de commande	Plan de requête
Claude Code	Toutes	Enforced sur le plan de commande	Observed sur le plan de requête
Codex CLI	Gérée, via MDM, un réglage système ou une configuration cloud	Enforced sur le plan de commande, et le développeur ne peut pas le désactiver	Observed sur le plan de requête
Codex CLI	Libre-service	Monitored sur le plan de commande jusqu'à ce que le développeur approuve le hook OpenLatch dans Codex CLI, puis Enforced sur ce plan	Observed sur le plan de requête
Cursor	Libre-service, dans l'IDE et la CLI, sous macOS et Windows	Enforced sur le plan de commande	Observed sur le plan de requête
Cline	Libre-service, dans VS Code, sous macOS	Monitored sur le plan de commande	Observed sur le plan de requête

`openlatch status` et `openlatch doctor` indiquent chaque agent de la machine avec son niveau de contrôle.

Comment chaque agent reçoit un verdict

Tous les agents ne peuvent pas exprimer nativement chaque verdict. Un verdict qu'un agent ne peut pas exprimer est transmis sous la forme la plus proche qu'il sait exprimer, enregistré comme tel, et jamais plus strict que ce que la règle prévoyait.

Agent	Ask	Block
Claude Code	Sa propre demande native	Son refus natif
Codex CLI	Un message qui porte la raison de la règle ; l'appel se poursuit	Son refus natif
Cursor	Natif sur les commandes shell et les appels MCP dans certains modes d'exécution de l'IDE ; ailleurs, l'action est autorisée et l'historique indique qu'Ask n'a pas pu être transmis	Son refus natif

Comment fonctionne OpenLatch

Cette page suit une action, du moment où l'agent la tente jusqu'au moment où son historique atteint la plateforme. Les politiques s'écrivent une fois, sur la plateforme. Chaque décision se prend sur la machine, par le client, sans appel réseau.



Figure 1. La plateforme compile et distribue les politiques. Le client décide localement de chaque action couverte, puis envoie son historique à la plateforme.

1. Les politiques atteignent le client

Vous ratifiez des politiques d'intention dans la console. La plateforme les compile et construit un bundle de politiques par agent, à partir de la zone effective de cet agent. Le client demande régulièrement son bundle à la plateforme :

La plateforme répond	Le client
Rien n'a changé	Garde le bundle qu'il détient
Un nouveau bundle	Vérifie son intégrité, puis le remplace d'un seul bloc
Rien, ou un bundle qui échoue à la vérification	Garde le dernier bundle valide

Pour récupérer le bundle tout de suite, lancez `openlatch system aip refresh`.

2. Le hook intercepte l'action

Le client se branche sur les hooks natifs de l'agent. Quand l'agent tente une action couverte, comme lancer une commande shell ou appeler un outil, le hook la transmet au client avant son exécution. Les

requêtes sortantes vers un modèle passent par un écouteur sur la même machine.

3. Le client décide

Le client évalue l'action à partir de son bundle résident, en mémoire, sur la machine. Ce chemin ne fait aucun appel réseau : l'agent n'attend jamais le réseau, et le verdict n'en dépend jamais. Quand plusieurs politiques correspondent, le verdict le plus restrictif l'emporte. [Verdicts](#) et [Modes](#) expliquent ce que fait chaque verdict en Monitor et en Enforce.

4. L'historique suit

Après la décision, le client masque les identifiants sur la machine et envoie à la plateforme l'historique de l'action et de son verdict. La plateforme conserve cet historique pour la piste d'audit, le graphe d'exécution et les rapports. Elle ne décide jamais une seconde fois et ne modifie jamais un verdict que le client a déjà rendu.

Hors ligne

Le bundle résident fait autorité ; ce n'est pas un cache. Réseau débranché, le client continue d'appliquer le dernier bundle valide, indéfiniment. Récupérer de nouvelles politiques et envoyer l'historique nécessitent une connexion ; décider, non.

Deux situations laissent une action sans décision, et l'action se poursuit alors : la machine n'a jamais détenu de bundle valide, ou le hook ne peut pas joindre le client sur sa propre machine. `openlatch doctor` signale les deux.

Confidentialité et télémétrie

Cette page explique ce qui quitte votre machine et nos sites, et comment désactiver ce que vous ne voulez pas voir envoyé.

Ce que le client envoie à la plateforme

Après chaque décision, le client envoie à la plateforme l'historique de l'action et de son verdict : ce que l'agent a tenté de faire, comme une commande, un chemin de fichier, un outil ou une destination réseau, et ce qui a été décidé. C'est cet historique que la console affiche dans le graphe d'exécution, les anomalies et la piste d'audit.

- Les identifiants sont masqués sur la machine avant que quoi que ce soit ne la quitte.
- Le client tient aussi un journal d'audit local sur la machine.
- La décision elle-même ne dépend jamais de la plateforme : voir [Comment fonctionne OpenLatch](#).

Télémétrie d'usage du client

Le client envoie une télémétrie d'usage et des rapports de plantage pour nous aider à l'améliorer. Tant que le client n'est pas connecté, la télémétrie d'usage ne porte qu'un identifiant aléatoire de l'installation. Une fois connecté, le client relie cet identifiant aux identifiants de votre compte et de votre organisation : sa télémétrie d'usage n'est alors plus anonyme. Les deux partent vers l'adresse de la plateforme elle-même : ils n'ajoutent aucune destination à vos règles réseau. Vous pouvez désactiver l'un comme l'autre.

Pour	Faites ceci
Savoir si la télémétrie d'usage est active, et pourquoi	<code>openlatch system telemetry status</code>
Désactiver la télémétrie d'usage	<code>openlatch system telemetry disable</code> , ou définissez <code>OPENLATCH_TELEMETRY_DISABLED=1</code> ou <code>DO_NOT_TRACK=1</code>
Réactiver la télémétrie d'usage	<code>openlatch system telemetry enable</code>
Désactiver les rapports de plantage	Définissez <code>OPENLATCH_CRASH_REPORTING=0</code>

Statistiques d'openlatch.ai et de cette documentation

`openlatch.ai` et `docs.openlatch.ai` n'ont pas de connexion et ne chargent aucun traceur tiers. Ils enregistrent des statistiques produit anonymes via [PostHog](#), hébergé dans l'UE, par un chemin interne à `openlatch.ai`.

Nous enregistrons :

- les pages que vous visitez et le temps que vous y passez ;
- les événements de conversion, comme la copie d'une commande d'installation ou le suivi d'un lien vers la console ;
- les Core Web Vitals, pour garder les sites rapides ;
- un échantillon de replays de session, avec tous les textes et champs de formulaire masqués ;
- les erreurs JavaScript non gérées, après qu'un filtre dans votre navigateur a remplacé tout ce qui ressemble à une adresse e-mail ou à un jeton d'API.

Nous ne collectons jamais ce que vous saisissez dans un formulaire, nous ne vous identifions jamais depuis ces sites, et nous ne vendons ni ne partageons ces données.

Refuser

Ajoutez `?ph_opt_out=1` à l'adresse de n'importe quelle page pour désactiver les statistiques sur cet appareil. Le choix est enregistré dans votre navigateur et survit aux rechargements. Vous pouvez aussi

bloquer le chemin `/ingest/` avec n'importe quel bloqueur de contenu.

En passant à la console

Quand vous suivez un lien de ces sites vers `app.openlatch.ai`, l'identifiant anonyme voyage dans le fragment de l'adresse, après le `#`, pour que la console puisse rattacher vos visites précédentes à votre compte une fois inscrit. Le fragment n'atteint jamais nos serveurs.

Guides

Chaque guide couvre une tâche dans la console à `app.openlatch.ai`, avec la page où vous la faites. Si vous débutez, commencez par [Écrire votre première politique](#) ; pour comprendre pourquoi la console se comporte ainsi, lisez les [Concepts](#).

Politiques et zones

Guide	Pages de la console
Rédiger une Agent Intent Policy	Autonomy Zone > Policies , l'assistant de création d'une politique d'intention
Faire passer une politique en Enforce	Autonomy Zone > Policies
Gérer les Autonomy Zones	Autonomy Zone , ses onglets Agents et Audit
Examiner les anomalies	Anomalies

Agents

Guide	Pages de la console
Trouver et lire un agent	Agents , et les onglets Overview , Identity et Autonomy Zone d'un agent
Enquêter sur ce qu'a fait un agent	Les onglets Sessions , Execution graph , Commands , Files , Domains et Anomalies d'un agent
Examiner la configuration d'un agent	Les onglets Hooks , MCP servers , Rules et Skills d'un agent
Suivre les dépenses d'un agent	L'onglet Economics d'un agent
Rechercher dans l'inventaire	Inventory et ses onglets
Suivre les événements en direct	Events
Lire la page d'accueil	Home

Rapports

Guide	Pages de la console
Planifier des synthèses d'activité	Digests
Configurer les canaux de notification	Settings > Notification Channels
Consulter le journal d'audit	Audit Trail , Settings > Audit Trail

Réglages de l'organisation

Guide

Pages de la console

[Gérer les utilisateurs et les équipes](#)

Settings > Users, Settings > Teams

[Configurer l'authentification unique](#)

Settings > SAML SSO, Settings > Auth Providers

[Connecter un annuaire pour la recherche des propriétaires](#)

Settings > Identity Providers

[Modifier le profil de l'organisation](#)

Settings > Organization Profile

[Ajuster le catalogue](#)

Settings > Catalogue

[Lire les licences et l'usage](#)

Settings > Licensing

Votre compte

Guide

Pages de la console

[Gérer les clés d'API](#)

Account > API Keys

[Gérer votre compte](#)

Account > Profile, Account > Security

Déployer OpenLatch sur un parc fait l'objet d'une section à part : [Déployer](#).

Rédiger une Agent Intent Policy

Rédigez une Agent Intent Policy (AIP) quand vous voulez qu'une Autonomy Zone autorise, interroge ou arrête un type d'action d'agent. Il vous faut le rôle editor ou supérieur. Si l'assistant indique que la rédaction n'est pas disponible, un administrateur de votre site OpenLatch n'a pas encore configuré la rédaction assistée par IA.

Une AIP est un ensemble d'atomes d'intention. À partir d'eux, OpenLatch génère une phrase et des cas de test concrets. Vous ratifiez la phrase, par Oui ou Non, avec les cas de test affichés à côté ; vous ne modifiez jamais les règles compilées. [Politiques d'intention](#) explique le modèle.

Créer la politique

- 1 Ouvrez **Autonomy Zone**, sélectionnez l'unité dont la zone portera la politique, puis ouvrez **Policies** et lancez la création d'une nouvelle politique d'intention.
- 2 **Brief.** Dites ce que vous voulez en une ou deux phrases, avec vos propres mots, ou partez d'un exemple.
- 3 **Policy.** Lisez la phrase générée par OpenLatch, avec les cas de test à côté. Chaque cas est une situation concrète et le verdict que rendrait la politique. Répondez Oui si la phrase dit ce que vous vouliez dire. Répondez Non et dites ce que vous vouliez dire à la place : OpenLatch régénère la phrase et ses cas.
- 4 **How it runs.** Choisissez le mode dans lequel la politique est ratifiée : Disabled, Monitor ou Enforce. Commencez en Monitor, sauf si vous avez déjà confiance dans la politique. Là où une règle peut interroger une personne, choisissez ce qui se passe en attendant la réponse : par défaut, l'action s'exécute et est signalée pour revue ; pour une action irréversible, vous pouvez activer, par atome, son arrêt jusqu'à ce que quelqu'un réponde.
- 5 **Ratify.** Lancez la relecture, qui applique la politique à l'activité récente de vos agents et montre ce qu'elle aurait décidé, puis ratifiez. Rien n'est appliqué tant que vous ne l'avez pas fait.

L'assistant de création d'une politique d'intention : Brief, Policy, How it runs, Ratify.

La ratification reconstruit le bundle de politiques. Chaque client le récupère à sa prochaine interrogation ; pour le récupérer tout de suite sur une machine, lancez `openlatch system aip refresh`.

Gérer les politiques dans la liste

L'onglet **Politiques** liste les politiques d'intention rédigées dans l'unité sélectionnée, avec leurs dimensions, leur couche, leur mode et les anomalies que chacune a levées sur les sept derniers jours. Activez **Show inherited policies** pour voir les politiques qu'une unité hérite des couches au-dessus d'elle.

OpenLatch

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

app.openlatch.ai/zone/policies

Autonomy Zone / Policies

Q

Ctrl+K

?

99+

99+

DC

Autonomy Zone

The operating envelope an agent works inside – what it may do, under which conditions, and which consequences the business accepts. The deployed client evaluates each covered action locally before it runs; OpenLatch records the outcome.

Find a unit...

Add unit

Unit (50)

Autonomy Zone (18)

Crispy Pay

+9

151

Corporate

49

Finance

IT

Legal

People

+2

1

Security

+3

8

AppSec

GRC

SecOps

Detection & Response

Data & Analytics

+1

6

BI

MLPlatform

Merchant Success

4

Onboarding

Support Engineering

No directory connected

Crispy Pay

Defines its own Autonomy Zone

Autonomy Zone

Policies (9)

Agents (0)

Audit

Add policies

Add policy preset

The intent policies attached to Crispy Pay.

Search intent policies

Dimension

Mode

INTENT POLICY	DIMENSION	LAYER	MODE	ANOMALIES 7D
☐ The agent only reaches appr... AIP-001 OWASP Agentic Security - ve...	Security	Attached here	MONITOR	11
☐ No destructive data-store o... AIP-002 OWASP Agentic Security - ve...	Safety	Attached here	ENFORCE	9
☐ Secret stores are never read AIP-003 OWASP Agentic Security - ve...	Security	Attached here	ENFORCE	13
☐ Opaque downloaded code is ... AIP-004 OWASP Agentic Security - ve...	Security	Attached here	ENFORCE	11
☐ Granting a file the execute b... AIP-005	Security	Attached here	MONITOR	0

L'onglet Policies : chaque politique d'intention avec sa dimension, sa couche, son mode et ses anomalies récentes.

Pour	Faites ceci
Lire une politique	Ouvrez-la. Le panneau montre ses intentions, les cas de test générés, son activité, sa provenance et l'historique de ses versions ratifiées
Changer son mode	Utilisez Mode dans les actions de la ligne, ou sélectionnez plusieurs politiques et choisissez Enforce , Monitor ou Disable
Changer une politique héritée	Ouvrez la couche qui l'a rédigée. Les actions groupées ne s'appliquent qu'aux politiques rédigées dans l'unité sélectionnée
Partir de modèles	Choisissez Add policy preset , examinez chaque politique du modèle, choisissez le mode de chacune, et appliquez-les
Supprimer une politique	Choisissez Delete . La politique est retirée pour cette couche et toutes les couches en dessous, et son historique reste dans l'audit de la zone

Une politique dont toutes les intentions sont réservées au Monitor peut tourner en Monitor mais pas en Enforce : Monitor enregistre ce qui se passe, et Enforce demande au moins une intention applicable.

Laisser la couverture resserrer une politique

OpenLatch apprend quels outils, programmes et cibles concernent les actions de vos agents, et peut proposer des changements qui gardent une politique couvrante pour ce qu'elle visait. Le **Coverage mode** de la politique décide du sort de ces propositions :

Coverage mode	Ce qui se passe
Off	La couverture n'est pas analysée pour cette politique
Strict	L'apprentissage est compté et affiché ; rien n'est proposé ni appliqué
Explicit	Les propositions attendent votre revue
Auto	Les changements restrictifs et les extensions du catalogue s'appliquent automatiquement, et chacun est inscrit dans l'historique de la politique

Examinez les propositions depuis la section **Coverage** de la politique, approuvez-les ou écartez-les, puis appliquez-les, ce qui ratifie une nouvelle version. [Ajuster le catalogue](#) couvre les entrées que la couverture apprend.

Ensuite, [faites passer la politique en Enforce](#) quand son historique montre qu'elle est juste.

Faire passer une politique en Enforce

Faites tourner une politique en Monitor jusqu'à ce que son historique montre qu'elle décide ce que vous attendez, puis passez-la en Enforce. Il vous faut le rôle editor ou supérieur, sur l'unité qui a rédigé la politique.

En Monitor, une politique enregistre ses anomalies et n'intervient jamais. En Enforce, le client applique son verdict avant que l'action ne s'exécute. [Modes](#) explique la différence.

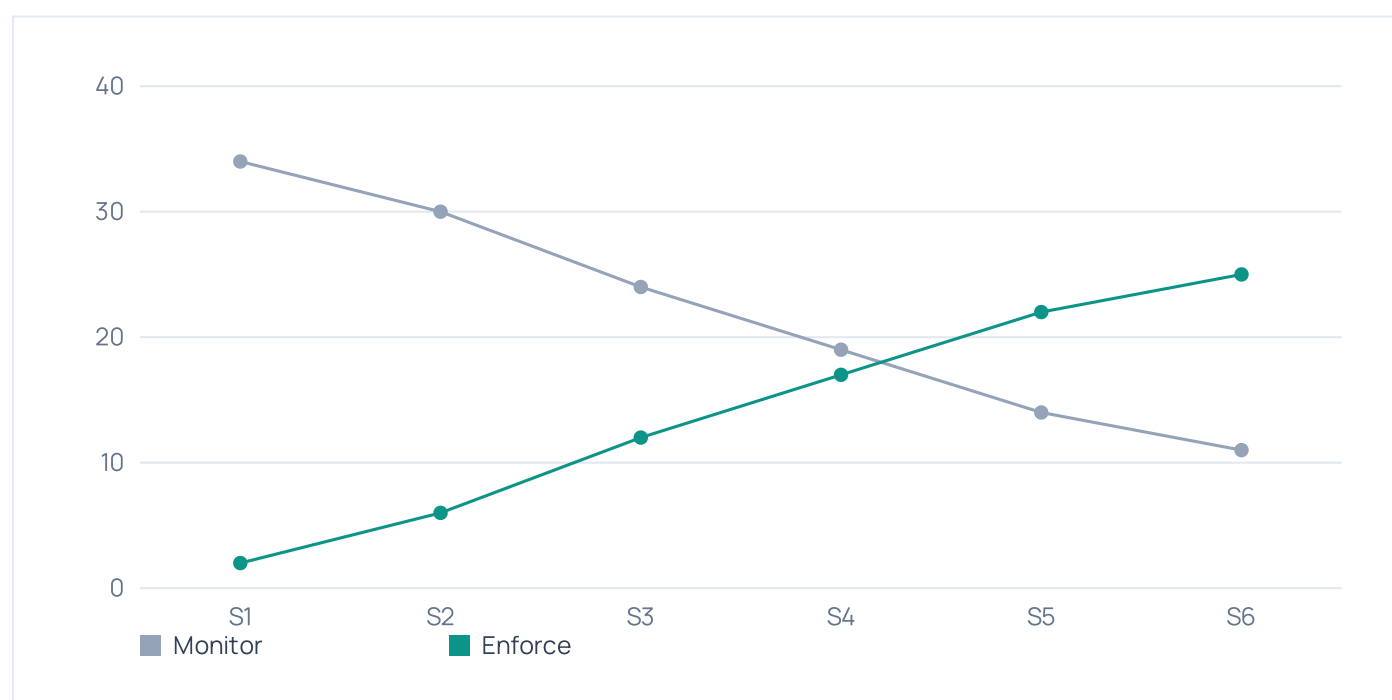


Figure 1. Les politiques de Crispy Pay sur leurs six premières semaines : la plupart passent de Monitor à Enforce une fois leurs verdicts jugés fiables. Données de démonstration illustratives.

Décider quand passer en Enforce

Faites passer une politique en Enforce quand son historique, sur une période représentative, une semaine de travail normal pour la plupart des équipes, ne contient aucune surprise :

Vérification	Où
Chaque anomalie levée est une action sur laquelle vous voulez qu'elle agisse	L' Activity de la politique, puis Anomalies filtré sur la politique
Elle n'a levé aucune anomalie sur un travail que vous jugez normal	La même liste, lue avec les propriétaires des agents
Ses intentions atteignent le seuil d'application	Le panneau de la politique. Une intention marquée réservée au Monitor lève des anomalies et n'intervient jamais, quel que soit le mode

Quand l'historique montre un faux positif, modifiez la politique au lieu de la faire passer en Enforce : ouvrez-la, répondez Non à sa phrase en disant ce que vous vouliez dire, et ratifiez la nouvelle version en Monitor.

Changer le mode

- 1 Ouvrez **Autonomy Zone**, sélectionnez l'unité qui a rédigé la politique, et ouvrez **Policies**.
- 2 Dans la ligne de la politique, ouvrez ses actions et réglez **Mode** sur **Enforce**. Pour en passer plusieurs à la fois, sélectionnez-les et choisissez **Enforce**.
- 3 Chaque client applique le changement à sa prochaine interrogation. Pour l'appliquer tout de suite sur une machine, lancez `openlatch system aip refresh`.

Dès lors, le client décide des intentions applicables de la politique avant que l'action ne s'exécute : Allow, Ask, Block ou Optimize. Un levier Optimize n'agit qu'en Enforce ; en Monitor, il mesure ce qu'il aurait fait. Le mode de chaque politique est inscrit dans le journal d'audit avec l'auteur du changement.

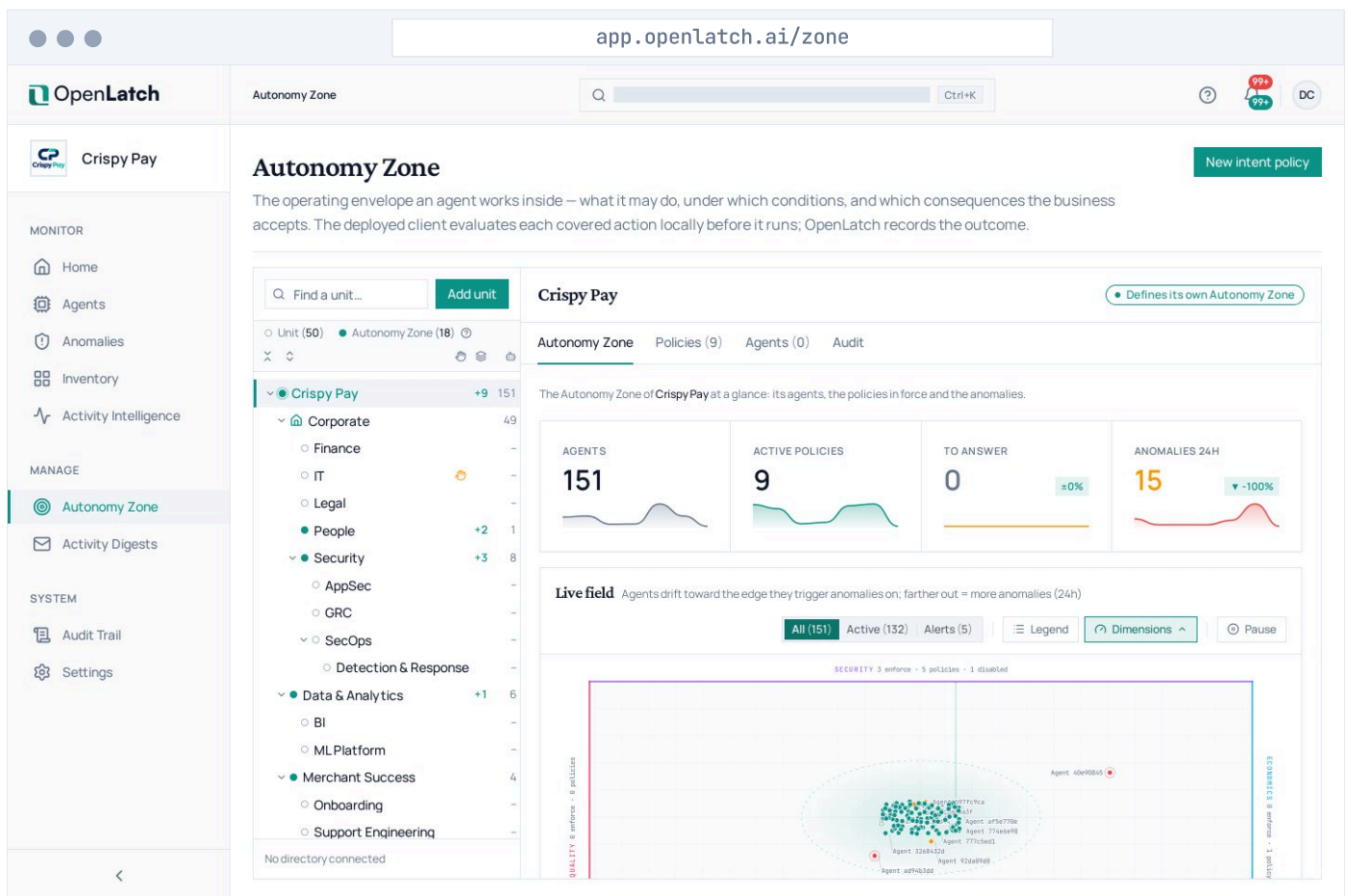
Revenir en arrière

Pour faire revenir une politique, réglez son mode sur **Monitor**. Elle continue d'enregistrer ses anomalies et cesse d'intervenir à la prochaine interrogation. Réglez-la sur **Disabled** pour cesser complètement de l'évaluer ; son historique reste disponible.

Gérer les Autonomy Zones

Utilisez la page **Autonomy Zone** pour décider quelles parties de votre organisation portent leurs propres règles, quels agents se trouvent où, et ce que chaque zone a décidé. Il vous faut le rôle editor ou supérieur pour modifier une zone ; tout le monde peut la lire.

Les Autonomy Zones se placent sur l'arbre des unités de votre organisation. N'importe quelle unité peut porter une zone, et la plupart n'en portent pas : une unité sans zone hérite de la zone la plus proche au-dessus d'elle, et chaque agent se résout en une seule zone effective. [Autonomy Zone](#) explique le modèle.

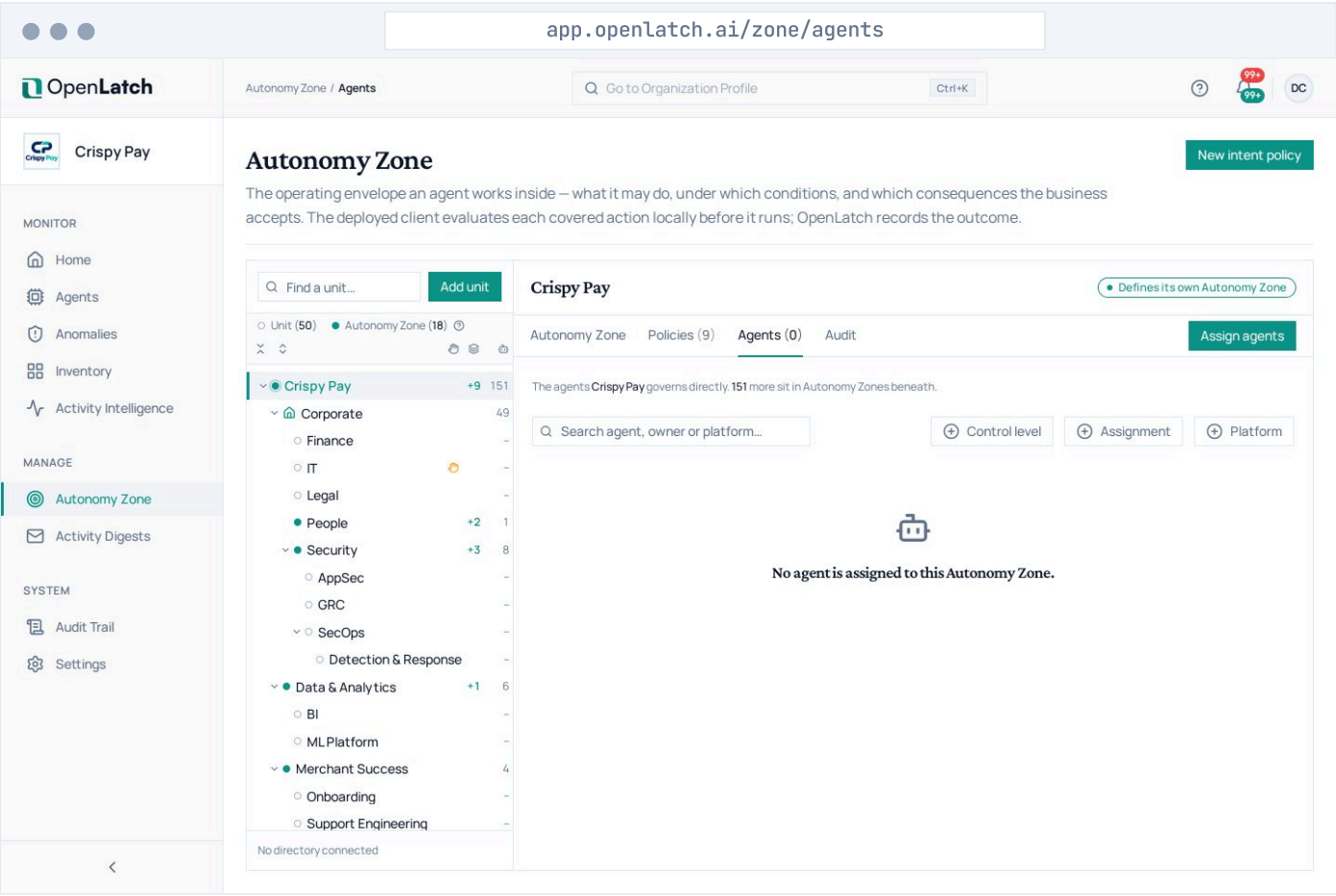


La page Autonomy Zone : l'arbre des unités de l'organisation, et la zone de l'unité sélectionnée en un coup d'œil.

Lire une zone

Sélectionnez une unité dans l'arbre. La page montre sa zone en un coup d'œil : ses agents, les politiques d'intention en vigueur et les anomalies qu'elles ont levées. Les onglets vont plus loin :

Onglet	Ce qu'il montre
Policies	Les politiques d'intention rédigées dans l'unité, et à la demande celles qu'elle hérite. Voir Rédiger une Agent Intent Policy .
Agents	Les agents que l'unité gouverne directement
Audit	Chaque action tentée par un agent de l'unité, avec la règle qui l'a concernée et la décision



L'onglet Agents : les agents qu'une unité gouverne directement.

The screenshot shows the OpenLatch application interface. The top navigation bar includes the OpenLatch logo and a search bar. The left sidebar contains navigation links for 'MONITOR' (Home, Agents, Anomalies, Inventory, Activity Intelligence) and 'MANAGE' (Autonomy Zone, Activity Digests). The 'SYSTEM' section includes 'Audit Trail' and 'Settings'. The main content area is titled 'Autonomy Zone' and shows the 'Crispy Pay' unit. A list of units is displayed on the left, including 'Corporate', 'Finance', 'IT', 'Legal', 'People', 'Security', 'AppSec', 'GRC', 'SecOps', 'Detection & Response', 'Data & Analytics', 'BI', 'ML Platform', 'Merchant Success', 'Onboarding', and 'Support Engineering'. The 'Audit' tab is selected, showing a table of decisions. The table has columns for 'WHEN', 'AGENT', 'DIMENSION', 'MATCHED', 'ACTION VALUE', 'MODE', and 'DECISION'. A message states 'No runtime decisions in this window.' and 'Decisions appear here after a client reports evaluated activity for this Autonomy Zone.'

L'onglet Audit : chaque action tentée par un agent de l'unité, avec la règle qui l'a concernée et la décision.

Donner sa propre zone à une unité

- 1 Sélectionnez l'unité et ouvrez ses réglages.
- 2 Activez sa propre Autonomy Zone. La zone commence vide et se comporte exactement comme la zone au-dessus d'elle tant que vous n'y rédigez pas de politique : chaque unité hérite de toutes les couches au-dessus d'elle dans tous les cas.
- 3 Rédigez les politiques qui ne s'appliquent qu'ici. Voir [Rédiger une Agent Intent Policy](#).

Pour retirer sa propre zone à une unité, utilisez **Delete Autonomy Zone** dans les réglages de l'unité. Les politiques rédigées dans la zone sont retirées, et ses agents passent à la zone du dessus.

Ajouter ou modifier une unité

Les unités viennent en général de votre annuaire, par [Connecter un annuaire pour la recherche des propriétaires](#). Le nom et la place d'une unité issue de l'annuaire appartiennent à l'annuaire et sont écrasés

à la synchronisation suivante ; son Autonomy Zone vous appartient.

Ajoutez une unité à la main pour une partie de l'activité que l'annuaire ne modélise pas, comme une ligne de production ou une coentreprise. Une unité ajoutée à la main n'est jamais écrasée par une synchronisation, et c'est le seul type d'unité que vous pouvez supprimer. La supprimer fait passer ses enfants et ses agents à son parent. L'arbre va jusqu'à dix niveaux de profondeur.

Placer les agents

Par défaut, un agent se trouve dans l'Autonomy Zone de son propriétaire, selon l'unité du propriétaire dans l'annuaire. Pour placer des agents ailleurs, ouvrez l'onglet **Agents** de la zone et choisissez **Assign agents** : le panneau fait entrer des agents dans cette zone, ou les rend à la zone de leur propriétaire. Un agent affecté à la main reste là où vous l'avez mis jusqu'à ce que vous le rendiez ; **Remove manual assignment** laisse de nouveau l'annuaire décider de sa zone.

L'onglet **Autonomy Zone** de l'agent montre la zone effective en laquelle il se résout, couche par couche. Voir [Trouver et lire un agent](#).

Choisir où un Ask atteint une personne

ASK delivery définit où un Ask issu des politiques d'intention de la zone atteint une personne :

Réglage	Où l'Ask apparaît
---------	-------------------

Platform	Sur la page Anomalies
----------	------------------------------

Agent	Dans l'invite de l'agent lui-même, là où l'agent peut poser la question ; ailleurs, comme Platform
-------	--

Both	Dans l'invite de l'agent et sur la page Anomalies
------	--

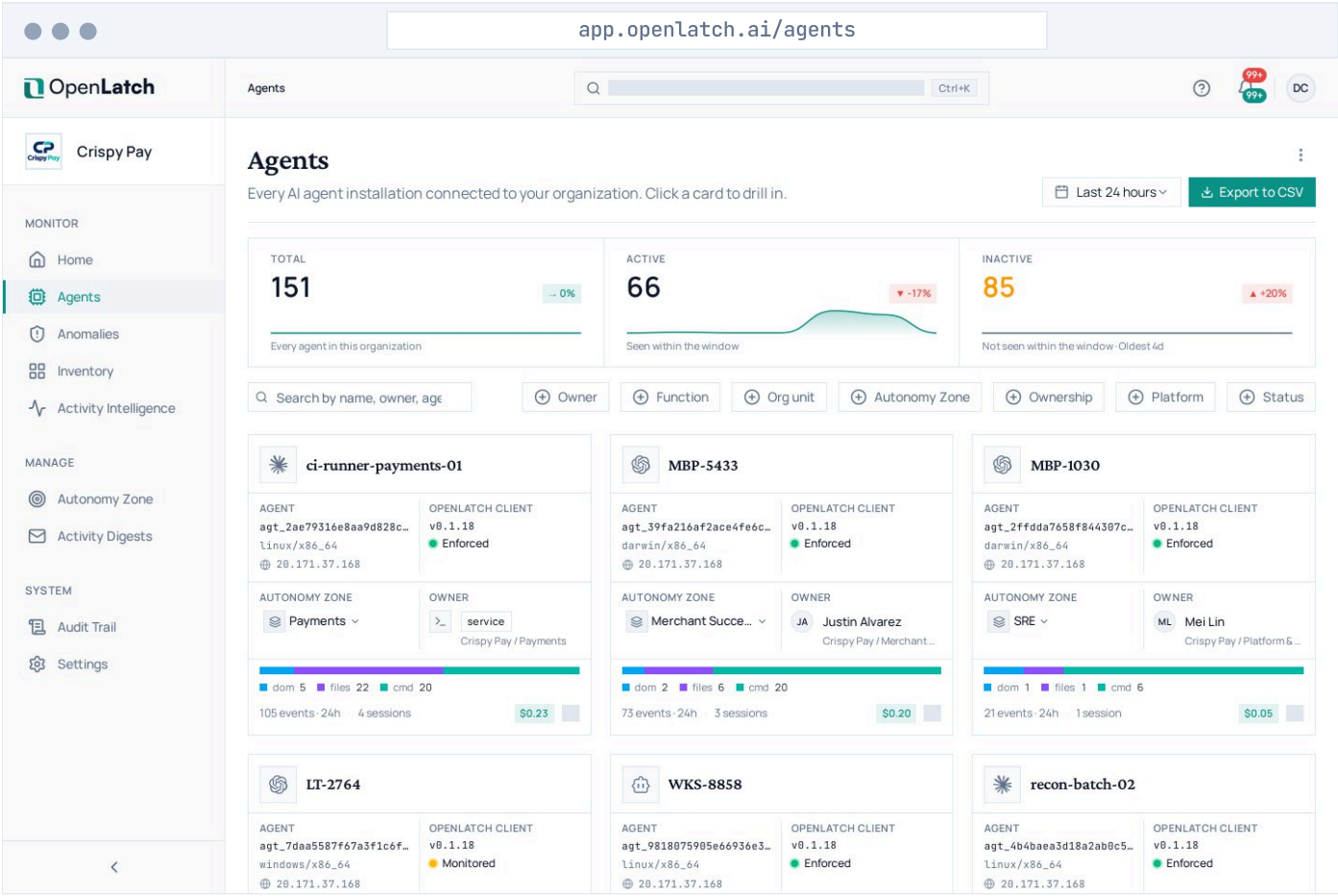
Les zones enfants héritent du réglage sauf si elles définissent le leur. Les agents qui ne peuvent pas poser la question dans leur propre invite sont listés à côté du réglage. Par défaut, un Ask laisse l'action s'exécuter et la signale pour revue. Voir [Verdicts](#).

Trouver et lire un agent

Utilisez la page **Agents** pour trouver n'importe quel agent qui a remonté des données à votre organisation, et ses onglets pour voir pour qui il agit et sous quelles règles il fonctionne. Tous les rôles peuvent la lire.

Trouver un agent

La page **Agents** liste chaque agent de l'organisation. Les chiffres en haut comptent tous les agents, les agents vus dans la fenêtre sélectionnée, et les agents non vus dans cette fenêtre. Cherchez par nom, propriétaire, plateforme ou hôte, et filtrez la liste pour la restreindre.



La page Agents : chaque agent de l'organisation, avec son propriétaire, son niveau de contrôle et sa dernière activité.

Chaque agent porte un [niveau de contrôle](#) : Enforced, Monitored ou Known. Il vous dit ce qu'OpenLatch peut faire de cet agent, pour que vous ne supposiez jamais qu'un agent est contrôlé parce qu'il est listé.

Pour connecter un nouvel agent, choisissez **Install an agent** et lancez la commande affichée sur la machine. [Installer le client](#) explique les commandes.

Lire la vue d'ensemble

Ouvrez un agent. Son onglet **Overview** résume ce qu'il a fait sur la fenêtre sélectionnée : ses actions, celles qui étaient sensibles, les points de terminaison qu'il a atteints, et les anomalies qu'il a levées.

The screenshot shows the OpenLatch web interface. The browser address bar displays `app.openlatch.ai/agents/ci-runner-payments-01/over...`. The interface has a sidebar on the left with sections: MONITOR (Home, Agents, Anomalies, Inventory, Activity Intelligence), MANAGE (Autonomy Zone, Activity Digests), and SYSTEM (Audit Trail, Settings). The main content area is titled 'ci-runner-payments-01' and shows an 'Overview' tab. Below the title, it indicates 'Claude Code', 'Linux x64', 'cClient v0.1.18', and '20.171.37.168'. The 'Overview' section includes a bar chart titled 'Activity pulse · last 7 days' showing action volume over time, and a 'Recent Sessions' table listing various events and their counts.

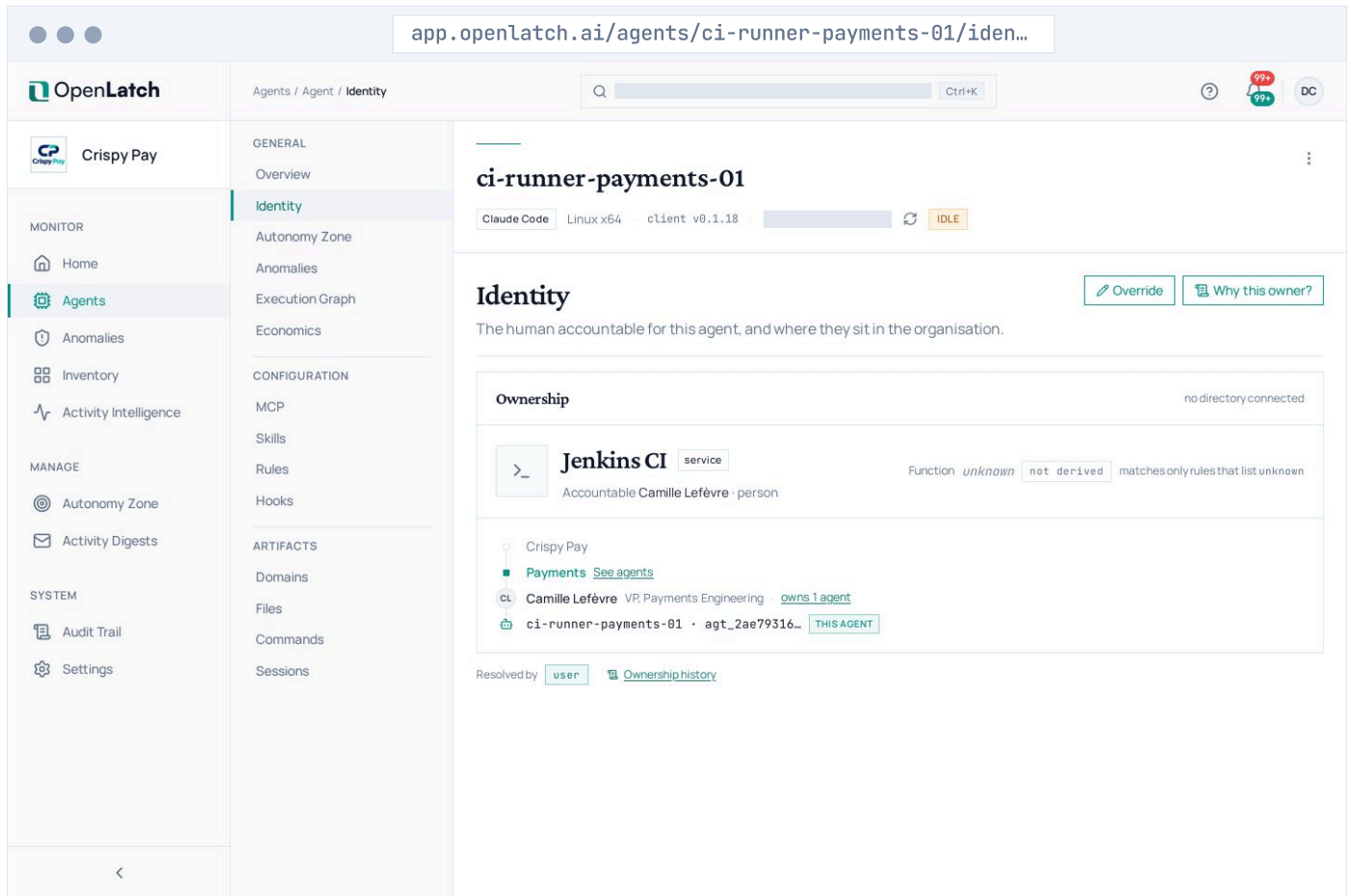
Session Name	Events	Details
Nightly dependency audit	2 events	46205075...
Set up cosign keyless signing for our container images in the build workflow	35 events	4d22ef28...
Where is the database schema defined?	51 events	d8e06603...
The autoscaler is flapping between 3 and 8 replicas every minute – tune the cooldown	17 events	c00f60d2...

L'onglet Overview d'un agent : son activité sur la fenêtre et ce qui mérite un coup d'œil.

Les autres onglets sont couverts par [Enquêter sur ce qu'a fait un agent](#), [Examiner la configuration d'un agent](#) et [Suivre les dépenses d'un agent](#).

Vérifier son propriétaire

L'onglet **Identity** nomme la personne responsable de l'agent et sa place dans l'organisation. Avec un annuaire connecté, le propriétaire est déterminé par une recherche en lecture seule ; voir [Connecter un annuaire pour la recherche des propriétaires](#).



L'onglet Identity d'un agent : son propriétaire, et sa place dans l'organisation.

- 1 Choisissez **Why this owner?** pour voir comment le propriétaire a été déterminé, des signaux remontés par la machine jusqu'à la fiche d'annuaire qui leur correspond.
- 2 Si le propriétaire est faux, choisissez **Override ownership** et désignez la bonne personne. Si elle n'est pas dans l'annuaire, ajoutez-la à la main avec son adresse e-mail et, facultativement, son nom et son intitulé de poste. Pour un agent qui agit pour un service plutôt que pour une personne, nommez le service et la personne qui en répond.

Remplacer un propriétaire demande le rôle editor ou supérieur, et c'est inscrit dans le journal d'audit.

Voir son Autonomy Zone

L'onglet **Autonomy Zone** montre la zone effective en laquelle l'agent se résout : les couches qu'il hérite, de l'organisation jusqu'à sa propre unité, et les politiques d'intention en vigueur sur lui. Un agent au niveau de contrôle Known est listé avec sa zone, mais sa zone ne lui est pas appliquée.

app.openlatch.ai/agents/ci-runner-payments-01/zone

OpenLatch

Agents / Agent / Autonomy Zone

Q

Ctrl+K

?

99+

99+

DC

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

GENERAL

Overview

Identity

Autonomy Zone

Anomalies

Execution Graph

Economics

CONFIGURATION

MCP

Skills

Rules

Hooks

ARTIFACTS

Domains

Files

Commands

Sessions

ci-runner-payments-01

Claude CodeLinux x64client v0.1.1820.171.37.168IDLE

Autonomy Zone

12 intent policies reach this agent from 2 Autonomy Zone layers — none authored on the agent.

AUTONOMY ZONE

Payments

inherits 2 layers - Crispy Pay

POLICIES IN FORCE

12

9 inherited - 3 in its Autonomy Zo...

ENFORCING

6 of 12

the other 6 record, and do not act

ANOMALIES - 14 DAYS

0

nothing flagged

2 Autonomy Zone layers apply here

Assigned to Payments, resolved top-down from the org tree. A layer below may tighten what it inherits.

Crispy Pay

Payments

Effective intent policies

INTENT POLICY	ALSO IN	LAYER	MODE	ANOMALIES 14D
Economics 1 policy - 0 enforcing - 1 in Monitor				
AIP-007 A session stays under sixty tool runs	-	Crispy Pay	MONITOR	0

L'onglet Autonomy Zone d'un agent : les couches qu'il hérite et les politiques d'intention en vigueur.

Pour déplacer un agent vers une autre zone, voir [Gérer les Autonomy Zones](#).

Supprimer un agent

Delete agent, dans le menu d'actions de l'agent, supprime définitivement l'agent avec chaque session et chaque événement enregistrés pour lui ; l'historique ne peut pas être récupéré. Si le client de cette machine envoie un nouvel événement, l'agent revient avec un nouvel identifiant. Retirez d'abord le client de la machine si l'agent doit disparaître pour de bon ; voir [Exploiter](#). Le même menu exporte l'agent en JSON.

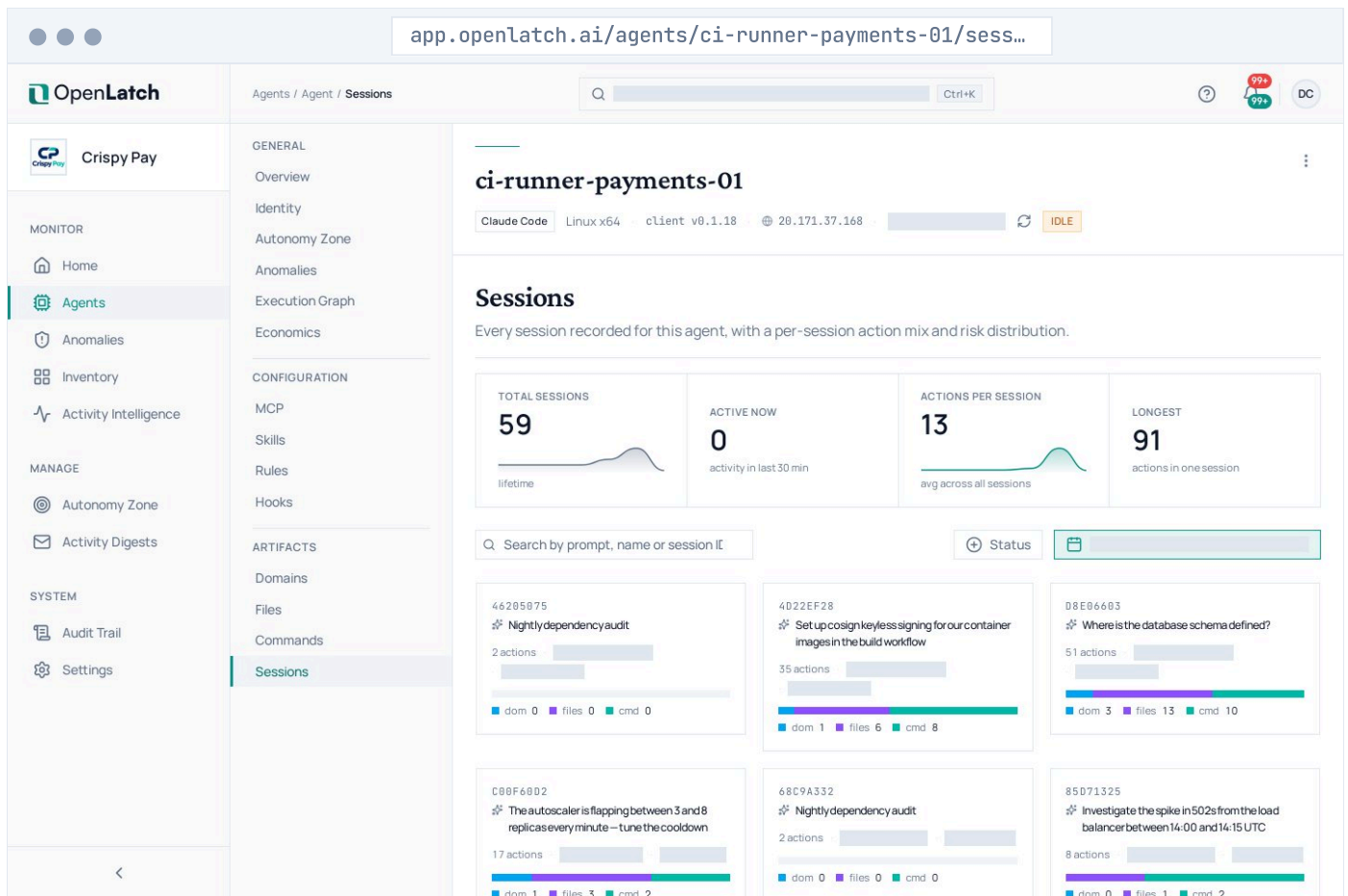
Enquêter sur ce qu’a fait un agent

Utilisez les onglets d’activité d’un agent pour répondre à la question « qu’a fait cet agent, et pourquoi cela a-t-il été décidé ainsi ? ». Ouvrez l’agent depuis la page **Agents**, réglez la fenêtre de temps en haut, puis choisissez l’onglet qui correspond à votre question. Tous les rôles peuvent les lire.

Votre question	Onglet
Qu’a-t-il fait pendant une session de travail ?	Sessions
Comment ses exécutions, ses outils et ce qu’il a touché s’articulent-ils ?	Execution graph
Quelles commandes shell a-t-il lancées ?	Commands
Quels fichiers a-t-il lus, écrits ou supprimés ?	Files
Quels sites web a-t-il atteints ?	Domains
Lesquelles de ses actions ont enfreint une politique d’intention ?	Anomalies

Sessions

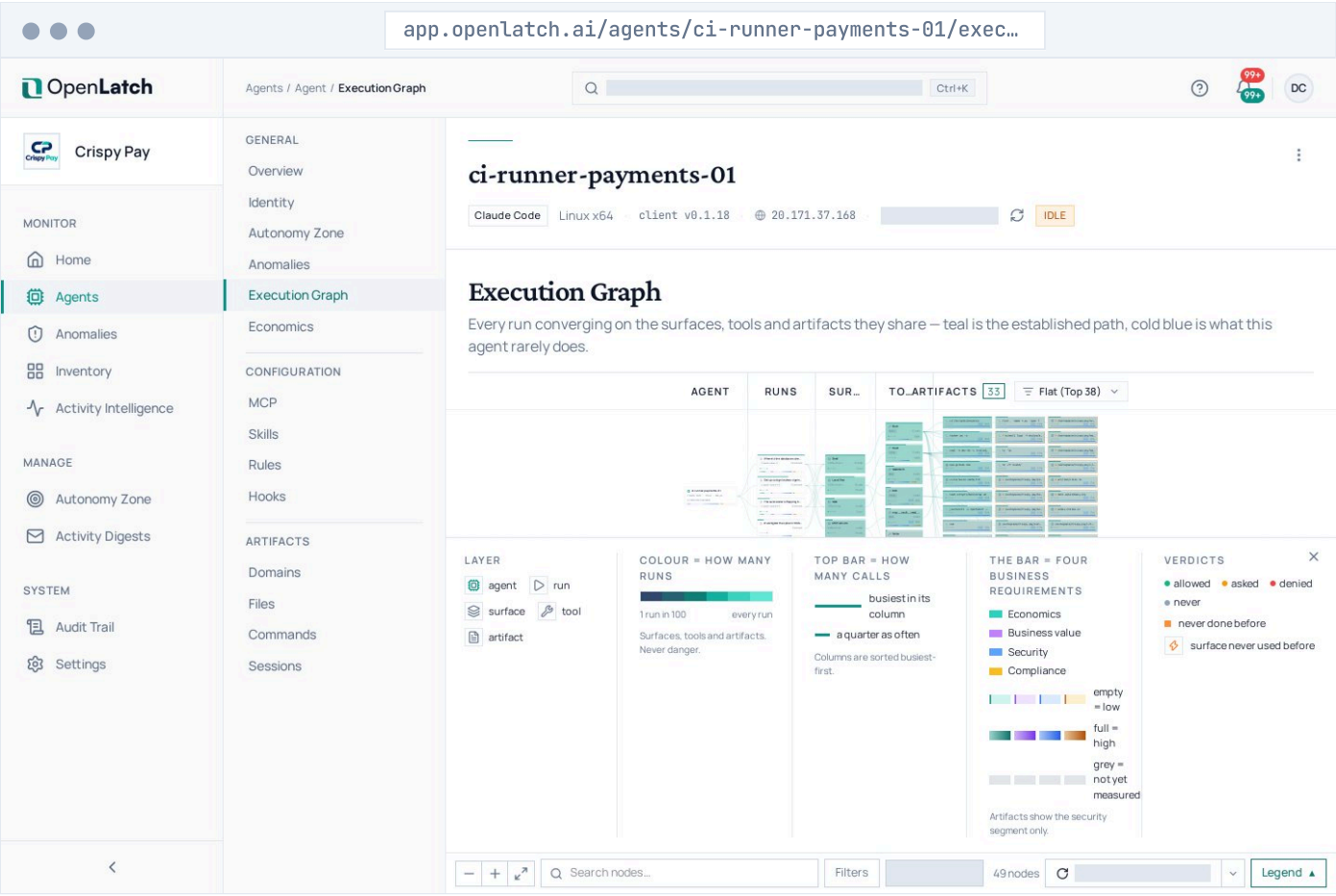
L’onglet **Sessions** liste les sessions de travail de l’agent, les plus récentes en premier. Ouvrez-en une pour lire ses actions dans l’ordre, chacune avec le verdict reçu et la règle qui l’a décidé.



L'onglet Sessions d'un agent : chaque session de travail, avec ses actions dans l'ordre.

Graphe d'exécution

L'onglet **Execution graph** dessine les exécutions de l'agent sur la fenêtre, convergeant vers les surfaces, outils et artefacts qu'elles partagent : l'agent, puis ses exécutions, puis ce que chaque exécution a sollicité, ce qu'elle a invoqué, et les fichiers, hôtes et commandes qu'elle a touchés ou lancés. La couleur d'un chemin indique sa fréquence dans la fenêtre. Sélectionnez un nœud pour lire les actions qui s'y rattachent. [Graphe d'exécution](#) explique le modèle.



L'onglet Execution graph d'un agent : les exécutions convergeant vers les surfaces, outils et artefacts qu'elles partagent.

Commandes, fichiers et domaines

Ces trois onglets listent chaque type d'action de l'agent, avec des chiffres sur la fenêtre en haut. Sélectionnez une ligne pour ouvrir son détail, avec chaque occurrence et la décision que chacune a reçue.

Onglet	Ce qu'il liste
Commands	Chaque commande shell lancée par l'agent, décomposée en programme et arguments
Files	Chaque fichier lu, écrit ou supprimé par l'agent, avec les lectures de fichiers d'identifiants et les écritures dans la configuration et le code source comptées à part
Domains	Chaque site web atteint par l'agent, avec les domaines atteints pour la première fois

app.openlatch.ai/agents/ci-runner-payments-01/comm...

OpenLatch

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

Agents / Agent / Commands

GENERAL

Overview

Identity

Autonomy Zone

Anomalies

Execution Graph

Economics

CONFIGURATION

MCP

Skills

Rules

Hooks

ARTIFACTS

Domains

Files

Commands

Sessions

ci-runner-payments-01

Claude CodeLinux x64cClient v0.1.1828.171.37.168IDLE

Commands

Every shell command this agent ran, ordered by time and scored per occurrence.

TOTAL COMMANDS

152

15 unique binaries

Q Search by command text...

Binary

TIME	BINARY	COMMAND
	cd	cd /scripts/projects
	cd	cd /src/tmp
	psql	psql -h dev-db -c truncate table scratch_events
	find	find . -name *.py -type f
	bash	bash scripts/bootstrap.sh

5 of 152 commands

1

2

...

7

L'onglet Commands d'un agent : chaque commande shell lancée.

OpenLatch

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

Agents / Agent / Files

Ctrl+K

?

99+

99+

DC

GENERAL

Overview

Identity

Autonomy Zone

Anomalies

Execution Graph

Economics

CONFIGURATION

MCP

Skills

Rules

Hooks

ARTIFACTS

Domains

Files

Commands

Sessions

ci-runner-payments-01

Claude CodeLinux x64cClient v0.1.1828.171.37.168IDLE

Files

Every unique file this agent touched, classified by intent and ordered by risk.

UNIQUE PATHS5853 actions across all paths

SECRET FILES READ0credential-risk reads

CONFIGS MODIFIED4configuration writes / edits

SOURCE FILES EDITED11source-code writes / edits

Q Search by path...

Intent: All (58)

Action

Last 24 Hours

NAME ↕	INTENT ↕	ACTION MIX	TOUCHES ↕	LAST SEEN ↕
OrganizationContext.tsx /workspace/crispy-pay/webapp/src...	SOURCE		1	
biome.json /workspace/crispy-pay	CONFIG		1	
ttsl kv/ci/build-cache	DOCS		12	
.gitconfig	DOCS		1	

Showing 1 to 25 of 58 files

<

1

2

3

>

L'onglet Files d'un agent : chaque fichier lu, écrit ou supprimé.

app.openlatch.ai/agents/ci-runner-payments-01/doma...

OpenLatch

Agents / Agent / Domains

CrispyPay

MONITOR

- Home
- Agents**
- Anomalies
- Inventory
- Activity Intelligence

MANAGE

- Autonomy Zone
- Activity Digests

SYSTEM

- Audit Trail
- Settings

GENERAL

- Overview
- Identity
- Autonomy Zone
- Anomalies
- Execution Graph
- Economics

CONFIGURATION

- MCP
- Skills
- Rules
- Hooks

ARTIFACTS

- Domains**
- Files
- Commands
- Sessions

ci-runner-payments-01

Claude Code Linux x64 cClient v0.1.18 28.171.37.168 IDLE

Domains

Every website this agent has touched, grouped by host and ranked by risk.

UNIQUE DOMAINS: 30 (37 total visits)

NEW LAST 24H: 2 (domains not seen before by this agent)

Search domain, country...

All (30) New last 24h (2)

DOMAIN	COUNTRY - ASN	IPS	TOUCHES	LAST SEEN	TREND
index.crates.io	UNK	151.101.46.137	1		
api.github.com	UNK	140.82.112.6 +3	7		
docs.sqlalchemy.org	UNK	104.26.11.194	1		
api.cohere.ai	UNK	34.96.76.122	1		
cloud.google.com	UNK	172.253.63.100	1		

Showing 1 to 25 of 30 domains

L'onglet Domains d'un agent : chaque site web atteint.

Le texte des commandes et les chemins de fichiers sont masqués sur la machine avant d'être envoyés : un identifiant n'apparaît jamais ici. Voir [Traitement des données et sécurité](#).

Anomalies

L'onglet **Anomalies** ne liste que les anomalies de cet agent : ses actions qui ont enfreint une politique d'intention de sa zone, chacune avec la politique, sa dimension et le verdict. [Examiner les anomalies](#) explique comment les traiter à l'échelle de l'organisation.

The screenshot displays the OpenLatch web application interface. The top navigation bar shows the URL `app.openlatch.ai/agents/ci-runner-payments-01/anom...`. The left sidebar contains a menu with sections: MONITOR (Home, Agents, Anomalies, Inventory, Activity Intelligence), MANAGE (Autonomy Zone, Activity Digests), and SYSTEM (Audit Trail, Settings). The main content area is titled 'ci-runner-payments-01' and shows the 'Anomalies' tab. It includes a status bar with 'Claude Code', 'Linux x64', 'client v0.1.18', and an 'IDLE' button. Below this, the 'Anomalies' section shows 'Intent-policy violations raised by this agent.' with a filter for 'Waiting on your answer' and 'All'. A summary table displays four categories: 'WAITING ON YOUR ANSWER' (0), 'ANOMALIES - 7 DAYS' (0), 'BLOCKED' (0), and 'ALLOWED AND FLAGGED' (0). A message box states: 'Nothing is waiting on your answer. Every flagged action has an answer against it, so the policy knows what to do next time.'

L'onglet Anomalies d'un agent : ses actions qui ont enfreint une politique d'intention.

Transformer une observation en politique

Quand une action montre quelque chose que vous voulez voir décidé autrement la prochaine fois, rédigez la politique d'intention qui le dit : voir [Rédiger une Agent Intent Policy](#). OpenLatch ne lève une anomalie que là où une politique ratifiée couvre l'action.

Examiner la configuration d'un agent

Utilisez les onglets de configuration d'un agent pour voir ce que l'agent est configuré pour faire sur sa machine, avant qu'il ne le fasse : les hooks qu'il déclare, les serveurs MCP auxquels il se connecte, les fichiers de règles qu'il lit et les skills qu'il peut lancer. Le client remonte cette configuration depuis la machine : un changement fait là-bas apparaît ici. Tous les rôles peuvent lire ces onglets.

Ouvrez l'agent depuis la page **Agents** et choisissez un onglet.

Onglet	Ce qu'il liste	Chiffres en haut
Hooks	Les hooks de cycle de vie que l'agent déclare, dont ceux qui le relient au client	Blocs de hooks, leurs gestionnaires, et les sources modifiées
MCP servers	Les serveurs connectés à l'hôte MCP de l'agent, avec les outils que chacun expose	Serveurs, outils sur l'ensemble, et sources modifiées
Rules	Les fichiers de règles et la mémoire de l'agent trouvés sur l'hôte	Fichiers, sources modifiées, et le plus gros fichier
Skills	Chaque skill et commande slash que l'agent peut lancer	Skills, et sources modifiées

Sélectionnez une entrée pour ouvrir son détail : où elle a été déclarée, son contenu, et sa dernière modification.

app.openlatch.ai/agents/ci-runner-payments-01/hooks

OpenLatch

Agents / Agent / Hooks

Q

Ctrl+K

?

99+

99+

DC

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

GENERAL

Overview

Identity

Autonomy Zone

Anomalies

Execution Graph

Economics

CONFIGURATION

MCP

Skills

Rules

Hooks

ARTIFACTS

Domains

Files

Commands

Sessions

ci-runner-payments-01

Claude CodeLinux x64cClient v0.1.18IDLE

Hooks

Lifecycle hooks declared by this agent.

TOTAL HOOK BLOCKS

4

Across this host

HANDLERS

7

Across all blocks

MODIFIED 7D

4

Sources mutated

Q

Search lifecycle hooks...

Lifecycle

Last change

NAME ↕	LIFECYCLE ↕	HANDLERS ↕	LAST CHANGED ↕
Claude Code hooks	PreToolUsePostToolUse+1	3	
Claude Code hooks	PostToolUse	1	
Claude Code hooks	PostToolUse	1	
Claude Code hooks	StopSessionStart	2	

Showing 1 to 4 of 4 sources

L'onglet Hooks d'un agent : les hooks de cycle de vie qu'il déclare.

app.openlatch.ai/agents/ci-runner-payments-01/mcp

OpenLatch

Agents / Agent / MCP

Q

Ctrl+K

?

99+

99+

DC

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

GENERAL

Overview

Identity

Autonomy Zone

Anomalies

Execution Graph

Economics

CONFIGURATION

MCP

Skills

Rules

Hooks

ARTIFACTS

Domains

Files

Commands

Sessions

ci-runner-payments-01

Claude CodeLinux x64cClient v0.1.1828.171.37.168IDLE

MCP servers

Servers connected to this agent's MCP host.

TOTAL SERVERS

12

Across this host

TOOLS REGISTERED

19

Across all servers

CHANGED 7D

12

Sources mutated

Q Search servers...

Scope: AllEnterprisePersonalProjectLocal

Last change

NAME	SCOPE	TOOLS	LAST CHANGED
mongodb	project	1	
postgres	project	2	
datadog	project	1	
twilio	personal	1	

Showing 1 to 12 of 12 sources

L'onglet MCP servers d'un agent : les serveurs connectés à son hôte MCP et leurs outils.

app.openlatch.ai/agents/ci-runner-payments-01/rules

OpenLatch

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

Agents / Agent / Rules

GENERAL

Overview

Identity

Autonomy Zone

Anomalies

Execution Graph

Economics

CONFIGURATION

MCP

Skills

Rules

Hooks

ARTIFACTS

Domains

Files

Commands

Sessions

ci-runner-payments-01

Claude CodeLinux x64cClient v0.1.1828.171.37.168IDLE

Rules files

Rules files and agent memory observed on this host.

TOTAL FILES6

LARGEST FILE340

MODIFIED 7D6

CLAUDE / AGENTS / memory

Body characters

Sources mutated

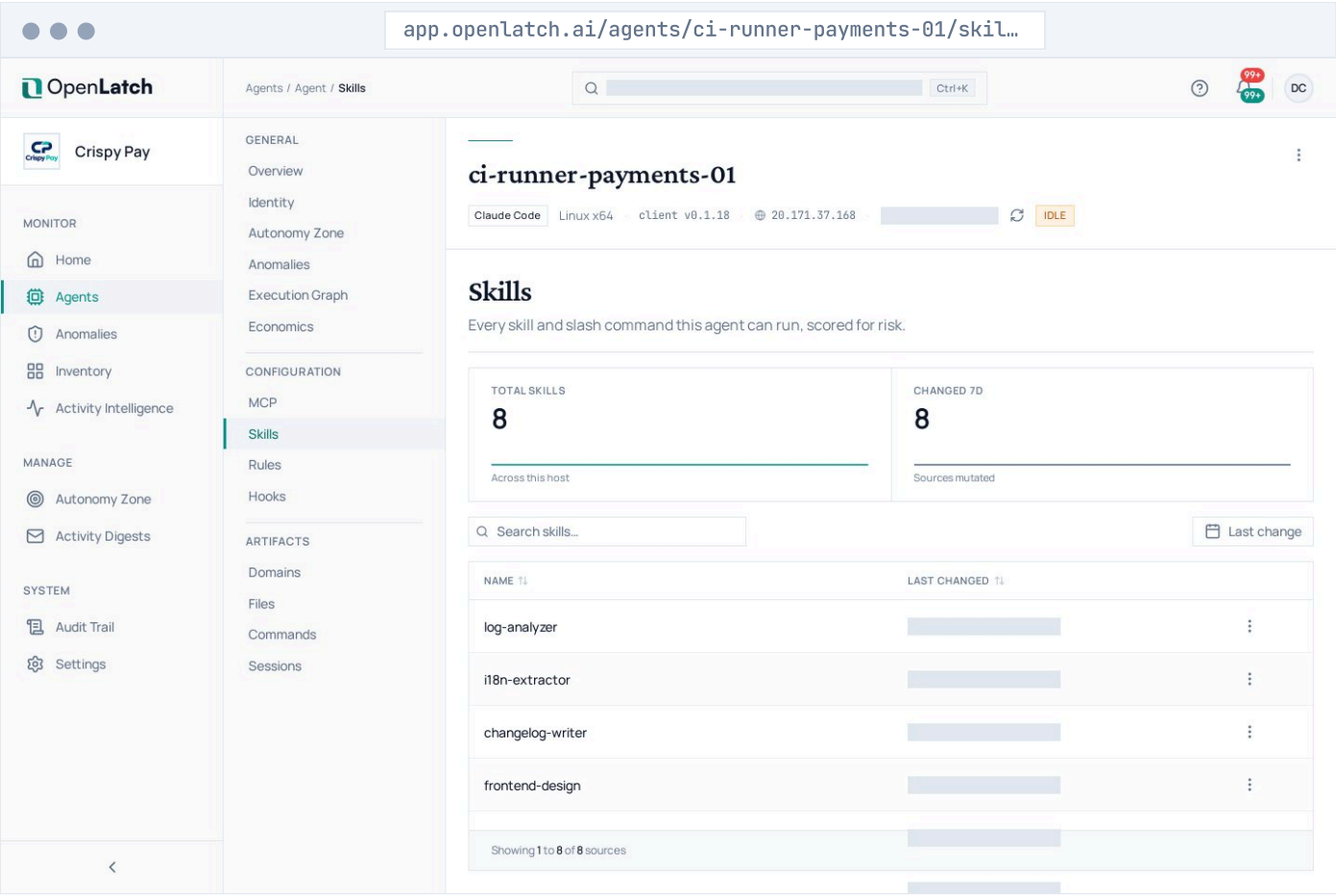
Q Search files...

Last change

NAME	PATH	LAST CHANGED
MEMORY.md	MEMORY.md	
AGENTS.md	AGENTS.md	
CLAUDE.md	CLAUDE.md	
CONTRIBUTING.md	CONTRIBUTING.md	

Showing 1 to 6 of 6 sources

L'onglet Rules d'un agent : les fichiers de règles et la mémoire trouvés sur son hôte.



L'onglet Skills d'un agent : chaque skill et commande slash qu'il peut lancer.

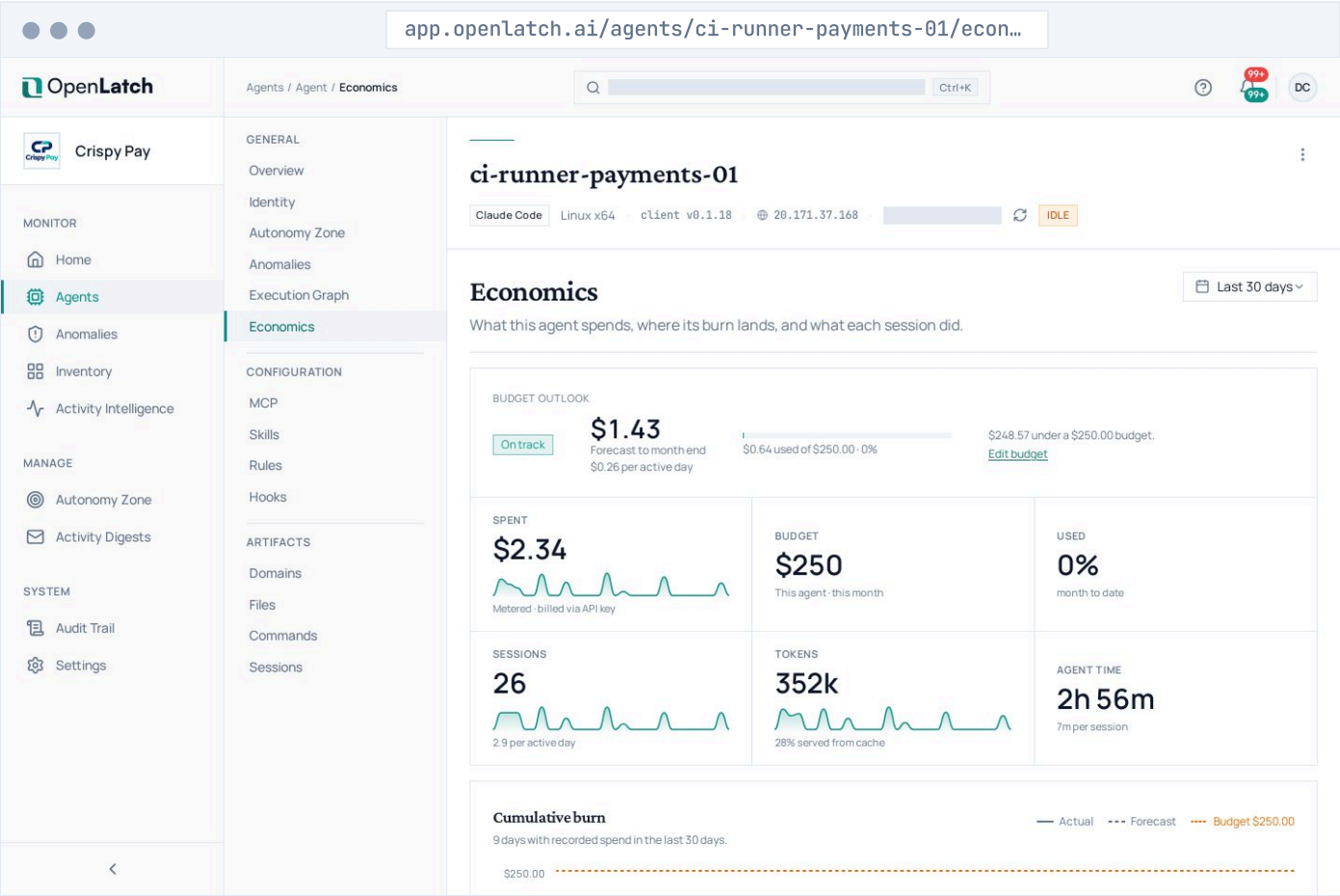
Ce qu'il faut regarder

Vérification	Pourquoi
Le hook qui relie l'agent au client est présent	Sans lui, les actions de l'agent n'atteignent pas le point de décision. Lancez <code>openlatch doctor</code> sur la machine s'il manque
Un serveur MCP ou un skill inattendu	Un nouveau serveur ou skill donne à l'agent de nouveaux outils pour agir. Rédigez une politique d'intention si son usage doit être décidé
Une source modifiée récemment	Un changement dans un fichier de règles ou un hook change le comportement de l'agent dès sa prochaine session

Un onglet vide signifie que l'agent n'a encore rien déclaré de ce type. Pour voir la même configuration sur tous les agents à la fois, utilisez l'[inventaire](#).

Suivre les dépenses d'un agent

Utilisez l'onglet **Economics** d'un agent pour voir ce qu'il dépense en modèles, où cette dépense se concentre, et ce que chaque session a produit. Tous les rôles peuvent le lire ; définir un budget demande le rôle admin ou supérieur.



L'onglet Economics d'un agent : dépenses, perspectives budgétaires et coût de chaque session.

Lire les dépenses

Section	Ce qu'elle montre
Dépenses	Ce que l'agent a dépensé sur la fenêtre. Une dépense facturée par une clé d'API est mesurée ; une dépense sous abonnement est estimée, et indiquée comme telle
Perspectives budgétaires	Comment les dépenses du mois se comparent au budget de l'agent
Consommation cumulée	La dépense réelle à date, projetée jusqu'à la fin de la période
Coût par modèle	La répartition de la dépense entre les modèles utilisés par l'agent
Part de l'entrée servie depuis le cache	La part de l'entrée de l'agent que le fournisseur de modèles a servie depuis son cache
Sessions	Chaque session avec son coût, sa durée et les outils utilisés. Une session encore en cours affiche un total provisoire, pas un chiffre final

Les requêtes et réponses de modèles ne sont jamais envoyées à OpenLatch ; les dépenses viennent du nombre de jetons, du modèle utilisé et de la répartition du cache que le client enregistre. Voir [Traitement des données et sécurité](#). Quand le prix d'un modèle est inconnu, l'onglet indique qu'une partie des dépenses manque dans ses chiffres.

Définir un budget

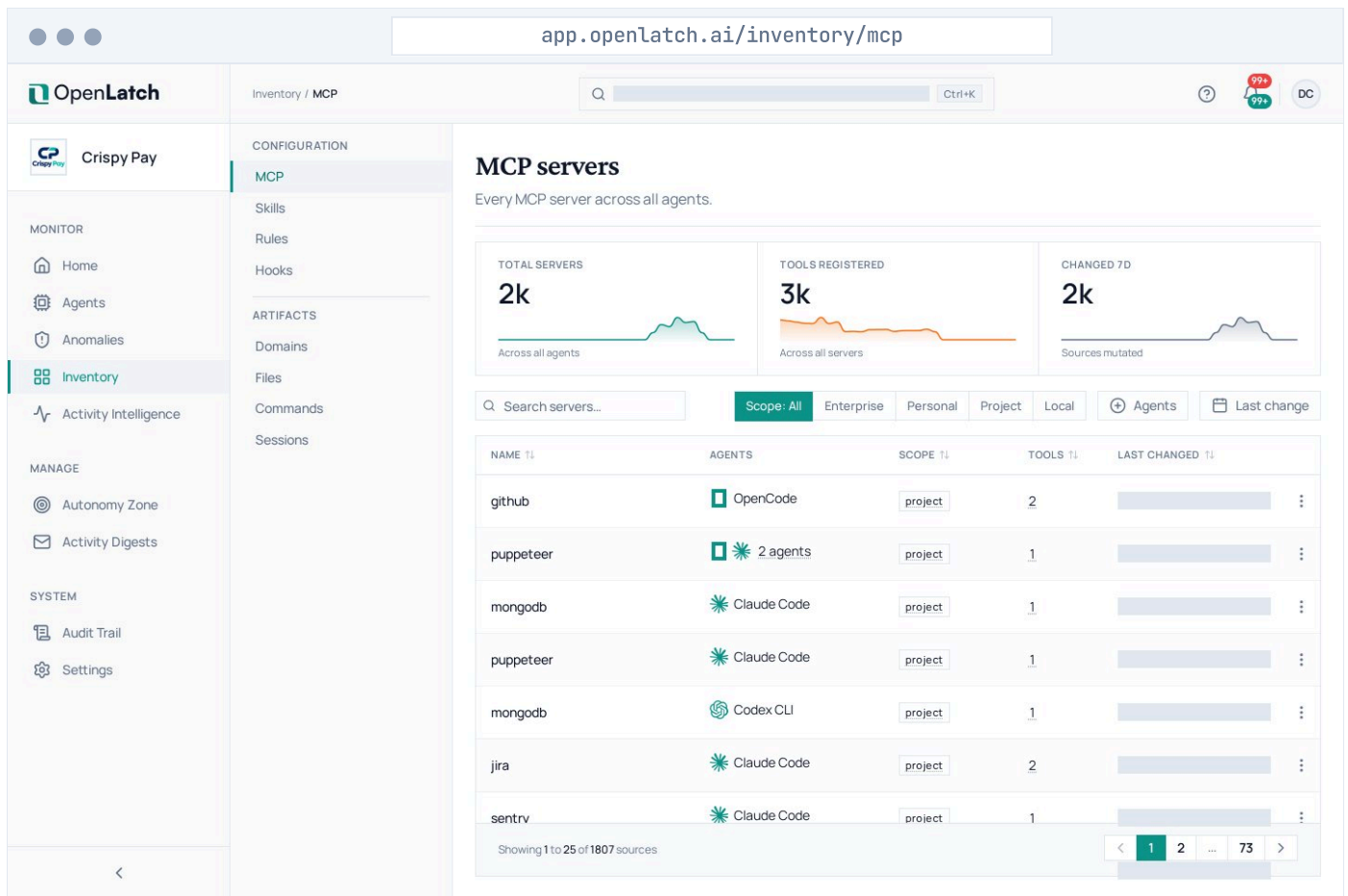
- 1 Dans l'onglet **Economics**, ouvrez le budget mensuel de l'agent.
- 2 Saisissez le montant du mois et enregistrez. Le budget repart à zéro le premier de chaque mois ; rien n'est reporté.

Un agent sans budget propre utilise le budget mensuel par défaut de votre organisation, défini dans **Settings > Organization Profile**. Ce budget par défaut s'applique à chacun de ces agents séparément : un défaut de 500 \$ donne à chaque agent 500 \$ par mois, pas 500 \$ partagés. Voir [Modifier le profil de l'organisation](#).

Rechercher dans l’inventaire

Utilisez la page **Inventory** quand votre question part d’une chose plutôt que d’un agent : quels agents se connectent à ce serveur MCP, lesquels ont lancé cette commande, lesquels ont atteint ce domaine. Chaque onglet liste un type de chose sur tous les agents de l’organisation, compté une seule fois quel que soit le nombre d’agents qui le partagent. Tous les rôles peuvent la lire.

Onglet	Liste, sur tous les agents
Commands	Chaque commande shell lancée par les agents
Files	Chaque fichier lu, écrit ou supprimé par les agents
Domains	Chaque site web atteint par les agents
Sessions	Chaque session de travail, avec l’agent auquel elle appartient
Hooks	Chaque bloc de configuration de hook déclaré par les agents
MCP servers	Chaque serveur MCP auquel les agents se connectent
Rules	Chaque fichier de règles lu par les agents
Skills	Chaque skill que les agents peuvent lancer

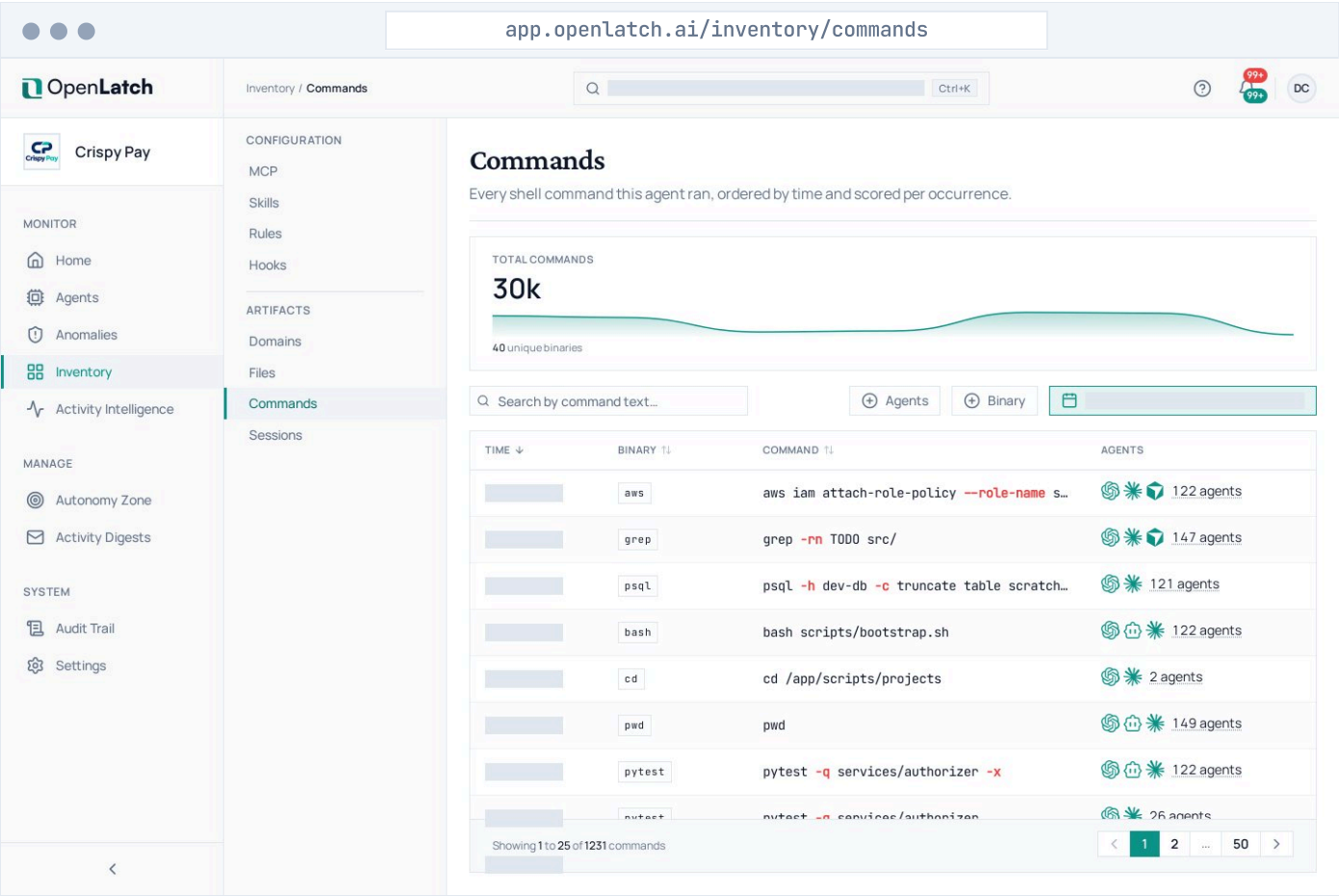


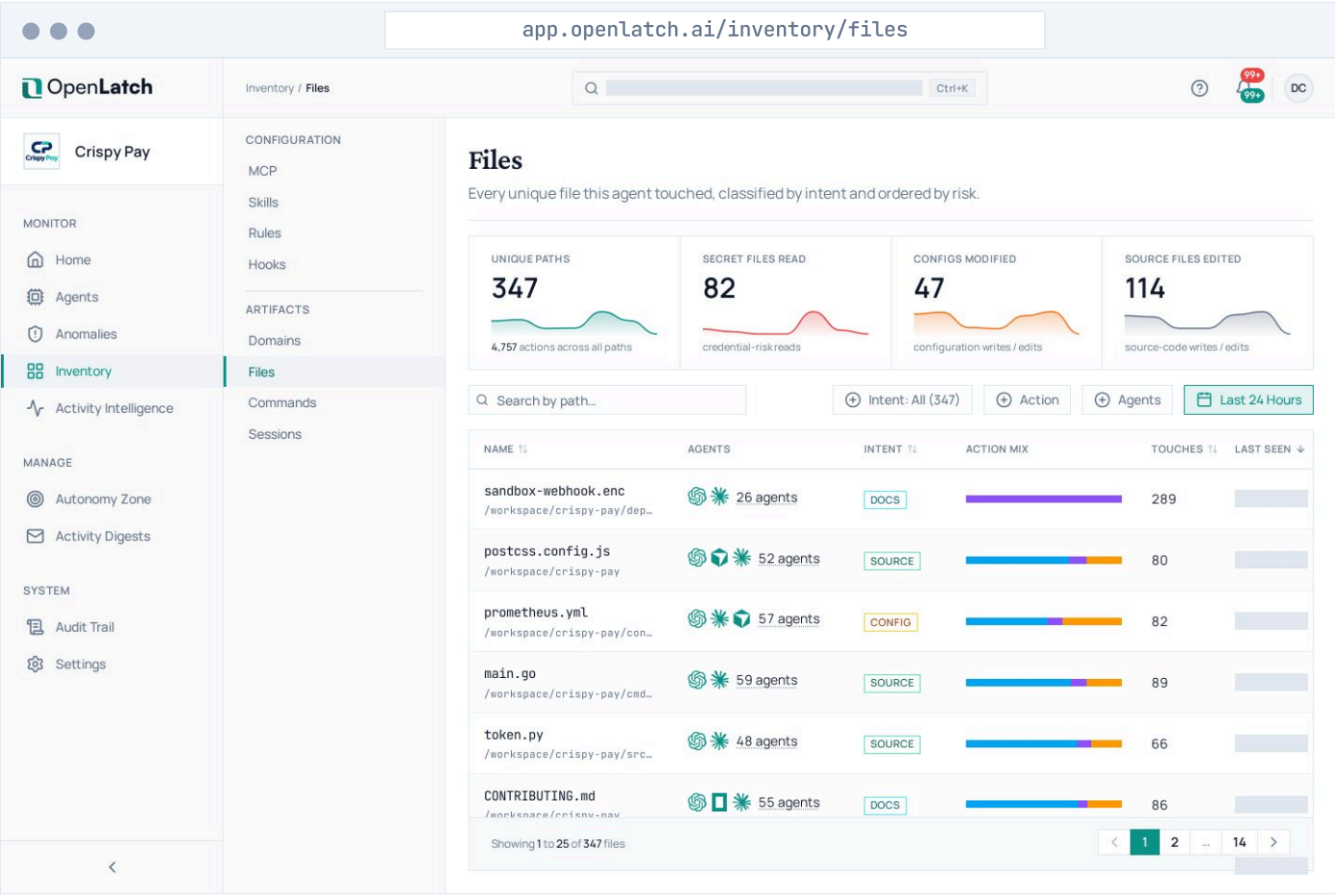
L'inventaire MCP servers : chaque serveur, avec le nombre d'agents sur lesquels il a été vu.

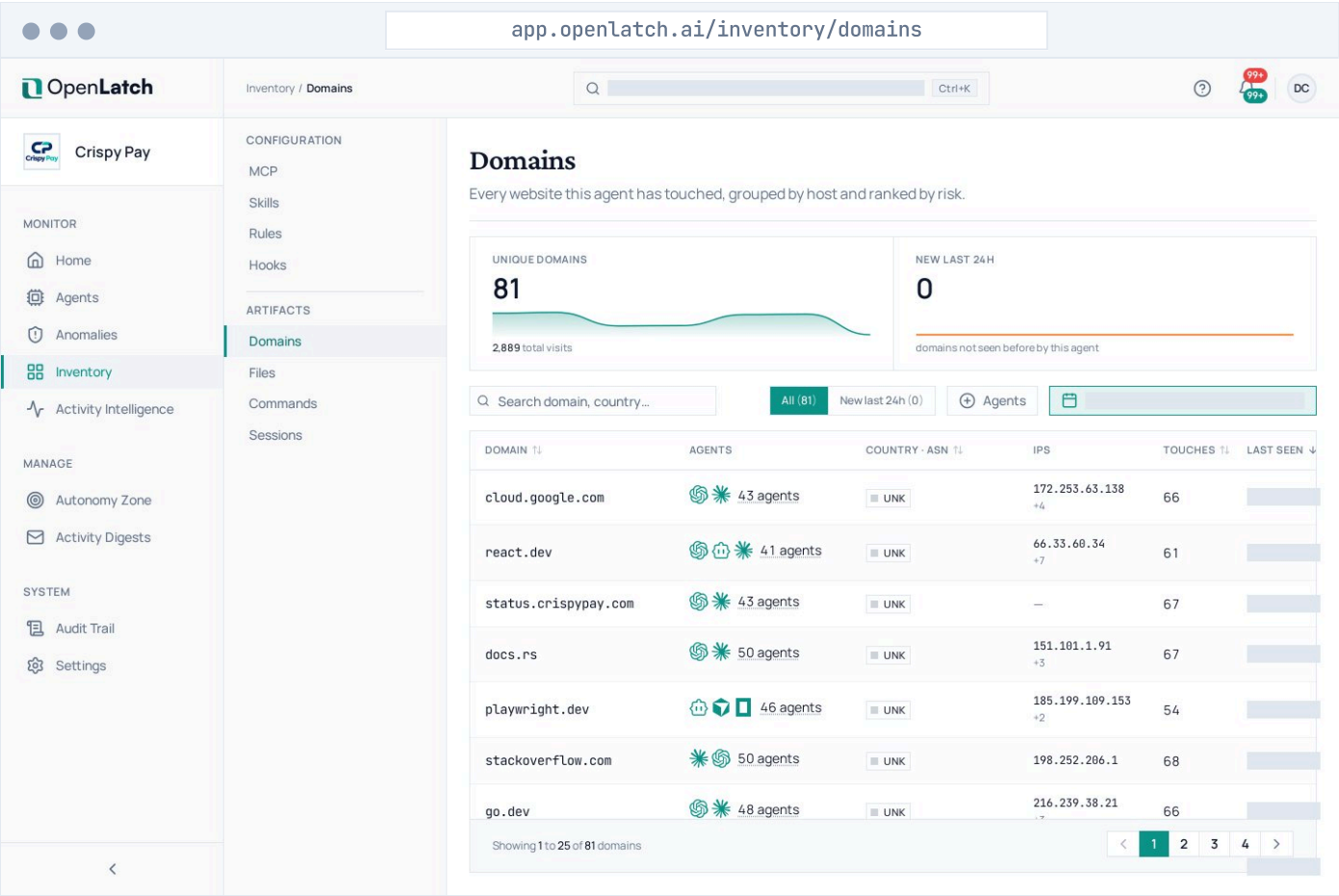
Trouver quels agents utilisent quelque chose

1. Ouvrez **Inventory** et l'onglet du type de chose que vous cherchez.
2. Cherchez-la, ou filtrez la liste.
3. Lisez la colonne **Agents**. Survolez-la pour voir sur quels agents l'entrée a été vue, ou ouvrez l'entrée pour chaque occurrence, chacune reliée à son agent.

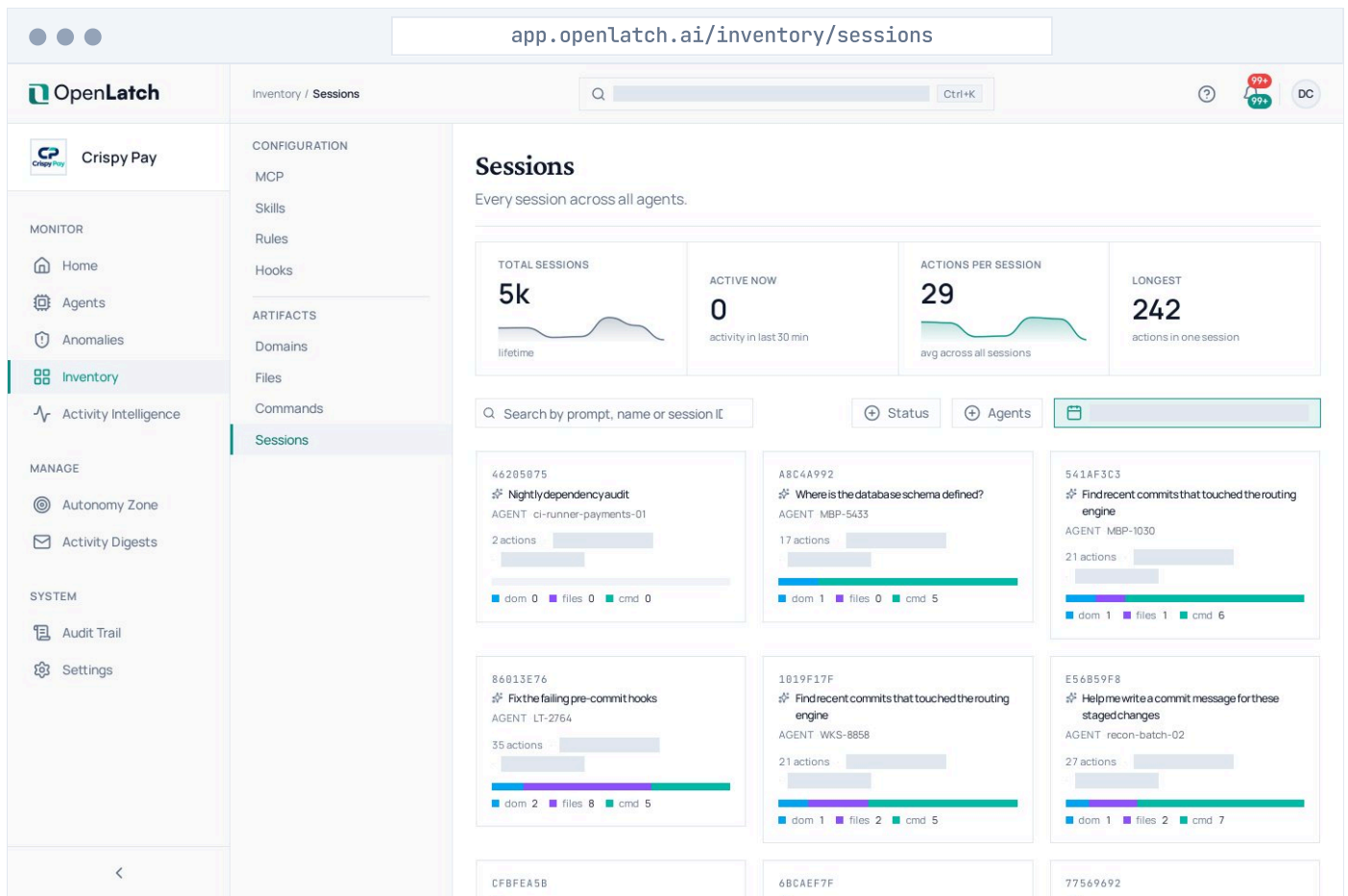
Depuis une occurrence, ouvrez l'agent pour lire l'action dans son contexte ; voir [Enquêter sur ce qu'a fait un agent](#).



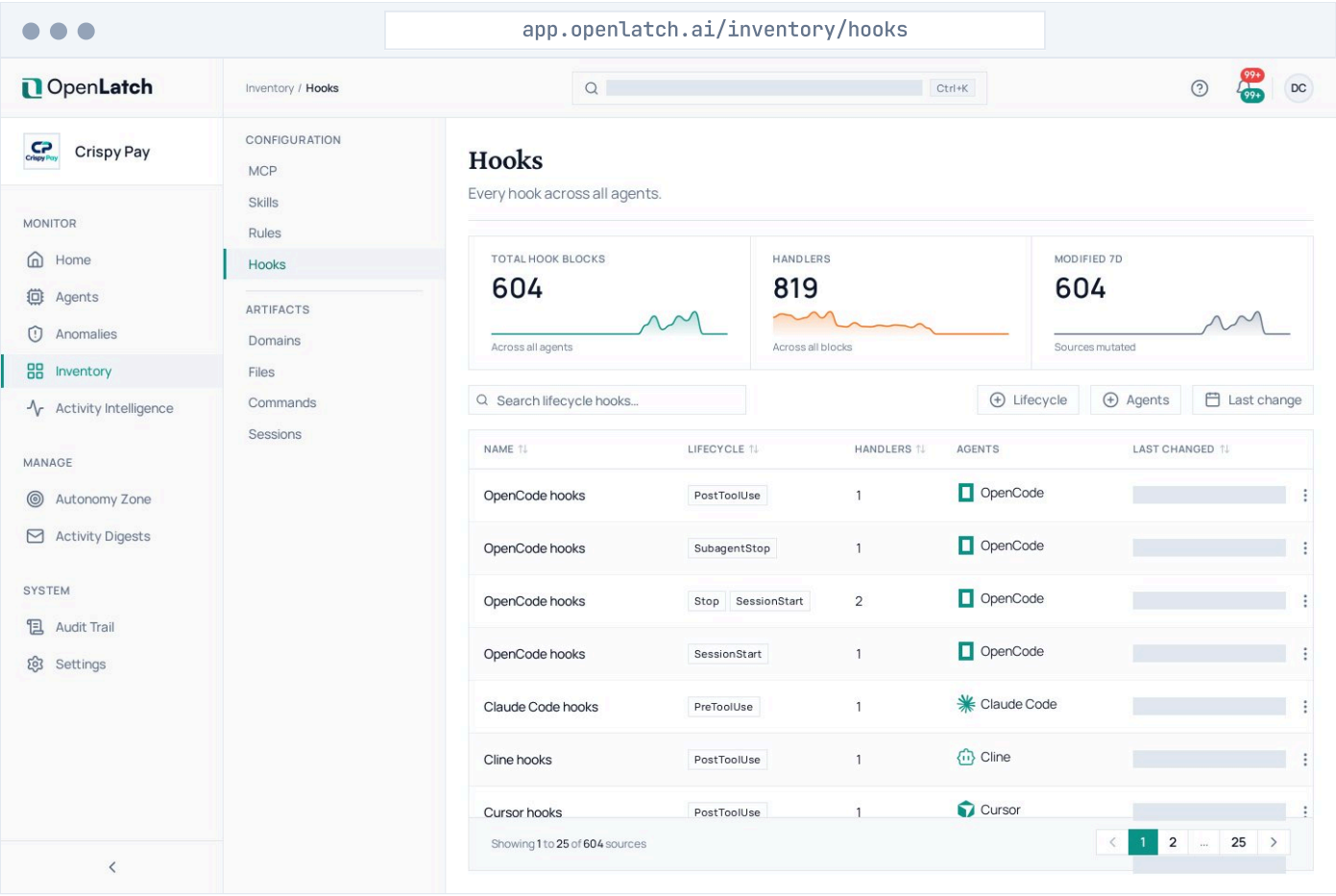


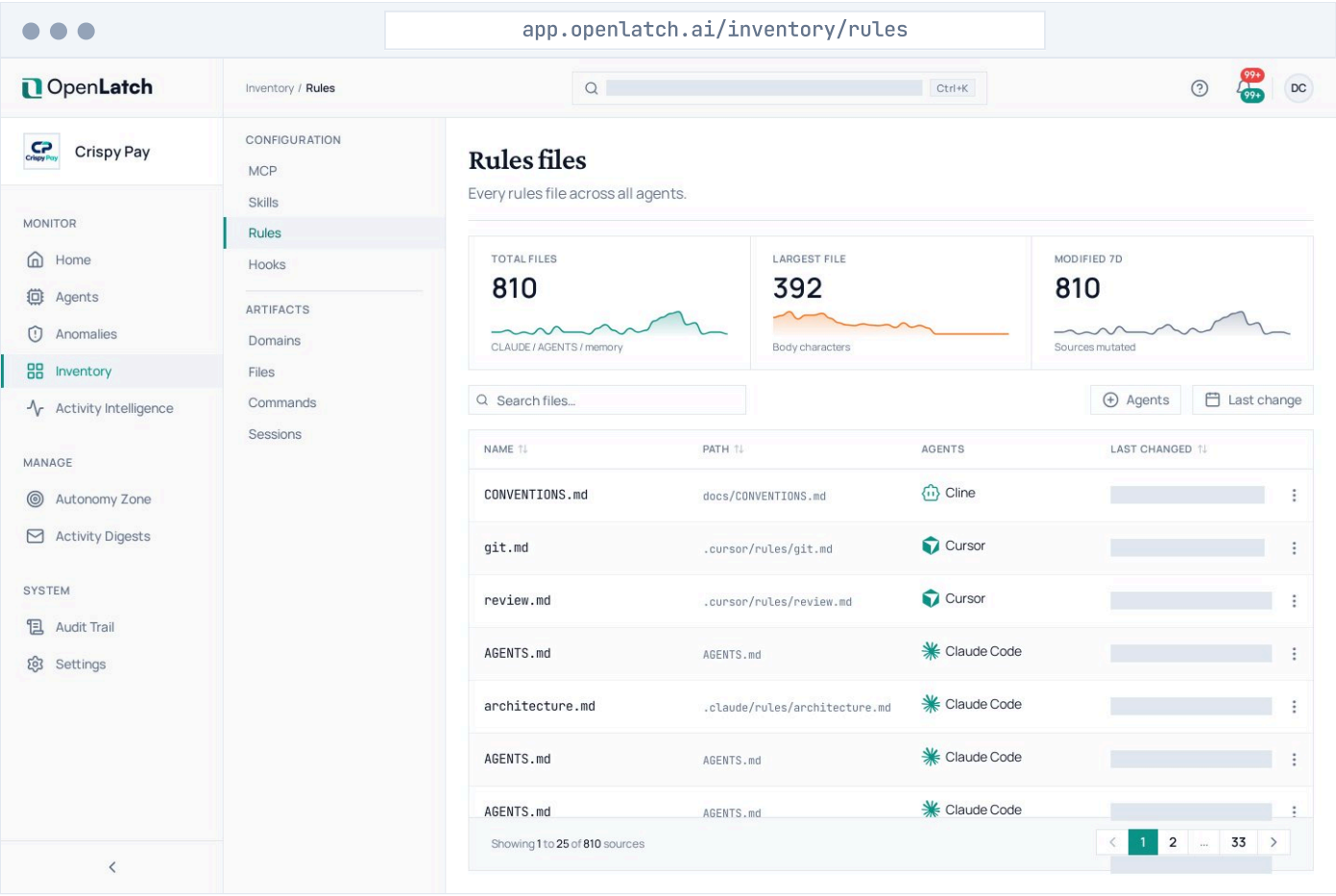


L'inventaire Domains : chaque site web atteint par les agents.



L'inventaire Sessions : chaque session de travail, avec l'agent auquel elle appartient.





app.openlatch.ai/inventory/skills

OpenLatch

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

Inventory / Skills

CONFIGURATION

MCP

Skills

Rules

Hooks

ARTIFACTS

Domains

Files

Commands

Sessions

Skills

Every skill across all agents.

TOTAL SKILLS

720

Across all agents

CHANGED 7D

720

Sources mutated

Search skills...

Agents

Last change

NAME [↑]	AGENTS	LAST CHANGED [↑]
ci-pipeline	Claude Code	⋮
refactor-helper	Claude Code	⋮
migration-author	Claude Code	⋮
refactor-helper	Claude Code	⋮
test-author	Claude Code	⋮
log-analyzer	Claude Code	⋮
log-analyzer	Claude Code	⋮

Showing 1 to 25 of 720 sources

<

1

2

...

29

>

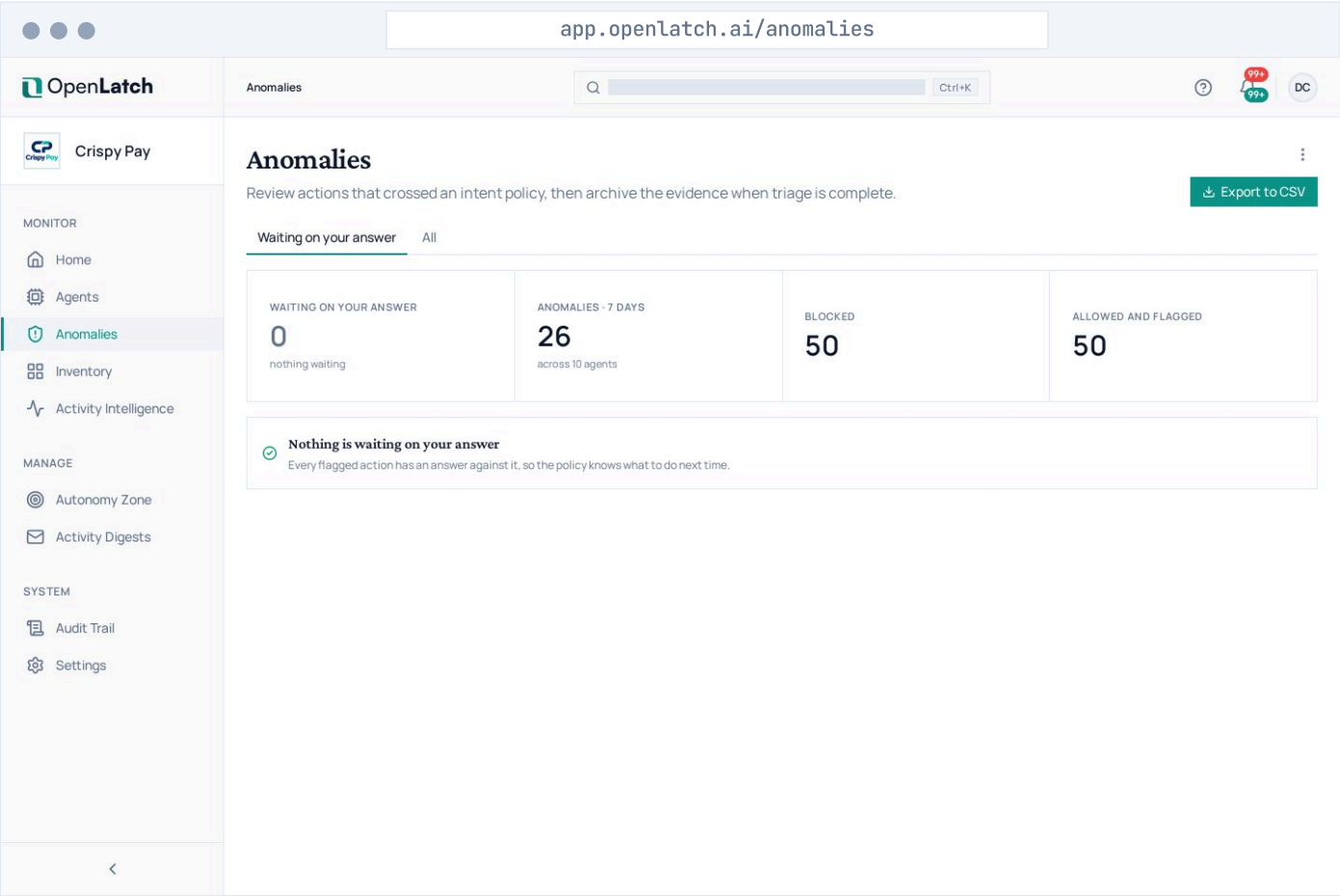
L'inventaire Skills : chaque skill que les agents peuvent lancer.

Un onglet vide signifie qu'aucun agent n'a encore déclaré ni fait quoi que ce soit de ce type.

Examiner les anomalies

Utilisez la page **Anomalies** pour traiter chaque action d'agent de votre organisation qui a enfreint une politique d'intention. Tous les rôles peuvent la lire ; répondre à un Ask demande le rôle editor ou supérieur.

Une anomalie est une action d'agent qui enfreint une politique d'intention de l'une des dimensions de l'Autonomy Zone de l'agent. Chacune nomme la politique, sa dimension, la zone d'où vient la politique et le verdict reçu par l'action. [Anomalies](#) explique le modèle.



La page Anomalies : chaque action qui a enfreint une politique d'intention, avec la politique, sa dimension et le verdict.

Traiter la liste

1. Réglez la fenêtre de temps et utilisez les filtres pour restreindre la liste à ce que vous examinez.

2. Ouvrez une anomalie. Son détail montre de quelle Autonomy Zone et de quelle politique elle vient, quelle intention a correspondu, l'action qui l'a levée, sa fréquence, et l'agent.
3. Ouvrez l'agent depuis l'anomalie pour lire l'action dans sa session ; voir [Enquêter sur ce qu'a fait un agent](#).

Ce que signifie une anomalie dépend du mode de la politique :

Mode de la politique	Ce qui est arrivé à l'action
Monitor	Elle s'est poursuivie, et l'anomalie enregistre ce que la politique aurait décidé
Enforce	Elle a reçu le verdict de la politique avant de s'exécuter, et l'anomalie l'enregistre

Répondre à un Ask

Quand l'ASK delivery d'une zone inclut la plateforme, les actions signalées par un Ask s'affichent en haut de cette page pour qu'une personne y réponde. Par défaut, ces actions se sont déjà exécutées : votre réponse ne les annule pas, elle décide de ce que la politique applique la prochaine fois qu'elle rencontre une action semblable. Ask empêche la récurrence, pas la première occurrence.

Réponse	Effet
Allow next time	La politique laisse s'exécuter les actions semblables à partir de maintenant
Block next time	La politique arrête les actions semblables à partir de maintenant
Not now – 24 h	La question revient dans un jour. Tant que vous ne répondez pas, elle se redéclenche sur le même motif

Là où une politique a activé, par atome, l'arrêt d'une action irréversible jusqu'à ce que quelqu'un réponde, l'action est listée comme bloquée jusqu'à la réponse. Chaque réponse est inscrite dans le journal d'audit avec votre nom. [Gérer les Autonomy Zones](#) couvre l'ASK delivery.

Depuis le détail d'une anomalie, **Promote to Enforce** passe sa politique en Enforce en une étape, quand la politique a des intentions applicables.

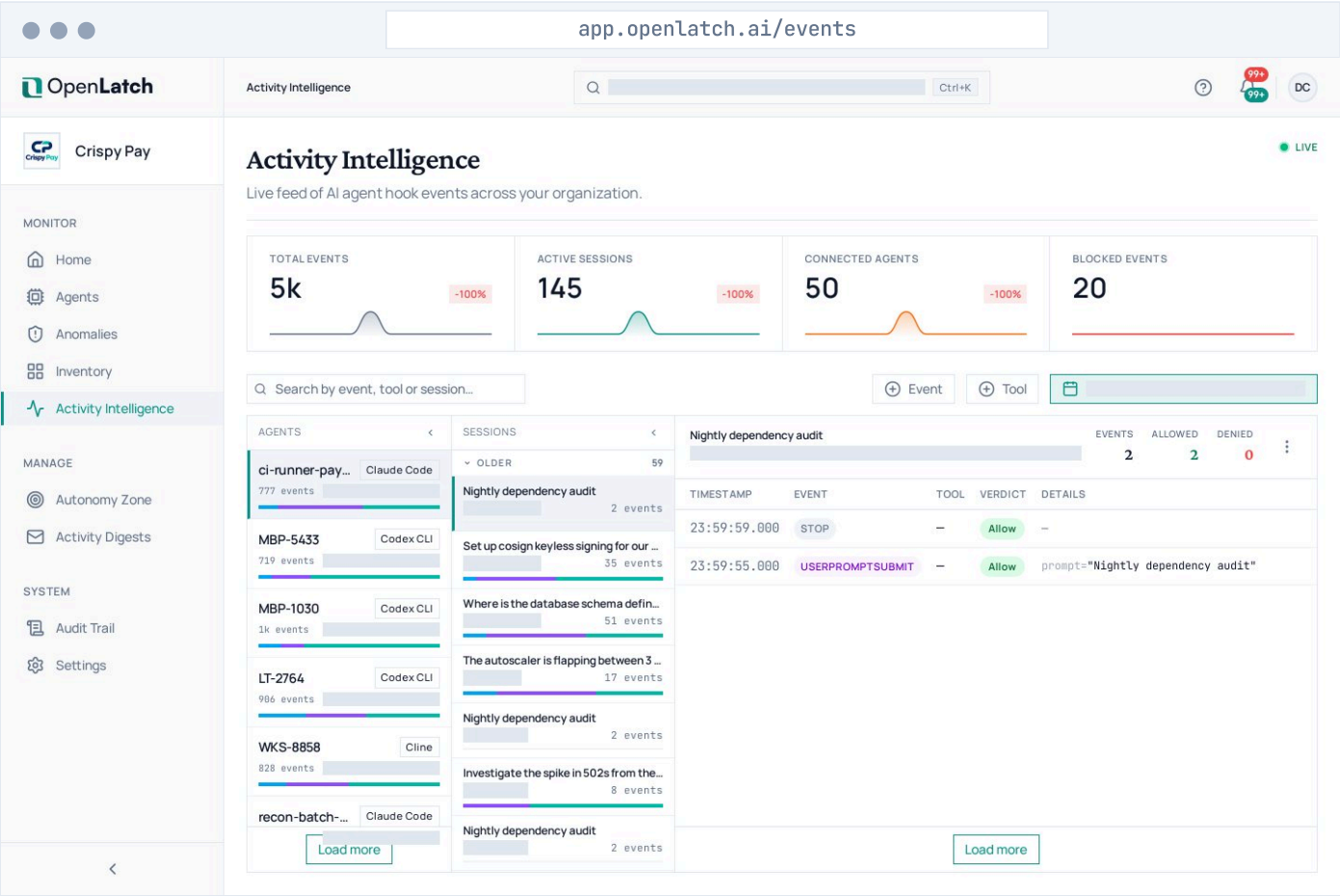
Ajuster la politique derrière une anomalie

Une anomalie sur un travail que vous jugez normal signifie que la politique en dit plus que ce que vous vouliez. Ouvrez la politique depuis l'anomalie, répondez Non à sa phrase en disant ce que vous vouliez dire, et ratifiez la nouvelle version ; voir [Rédiger une Agent Intent Policy](#). Une politique dont toutes les anomalies sont des actions sur lesquelles vous voulez agir est prête à [passer en Enforce](#).

Pour ne voir que les anomalies d'un agent, ouvrez son onglet **Anomalies** depuis la page **Agents**.

Suivre les événements en direct

Utilisez **Activity Intelligence**, la page des événements en direct de la console à /events, pour voir ce que font les agents en ce moment : chaque événement de hook de votre organisation, dès son arrivée. Tous les rôles peuvent la lire.



Activity Intelligence : le flux en direct des événements de hooks des agents, regroupés par session.

Lire le flux

Les chiffres en haut comptent, sur la plage de temps sélectionnée, les événements reçus, les sessions actives, les agents connectés et les événements bloqués, chacun comparé à la première moitié de la fenêtre. Un indicateur en direct montre que le flux est connecté.

En dessous, les sessions sont listées à gauche, regroupées en **Active now**, **Recent** et **Older**. Sélectionnez une session pour lire ses événements à droite, dans l'ordre. Ouvrez un événement pour son détail.

Pour	Faites ceci
Trouver un événement	Cherchez par événement, outil ou session
Restreindre le flux	Filtrez par type d'événement, outil ou plage de temps
Repartir de zéro	Choisissez Clear filters

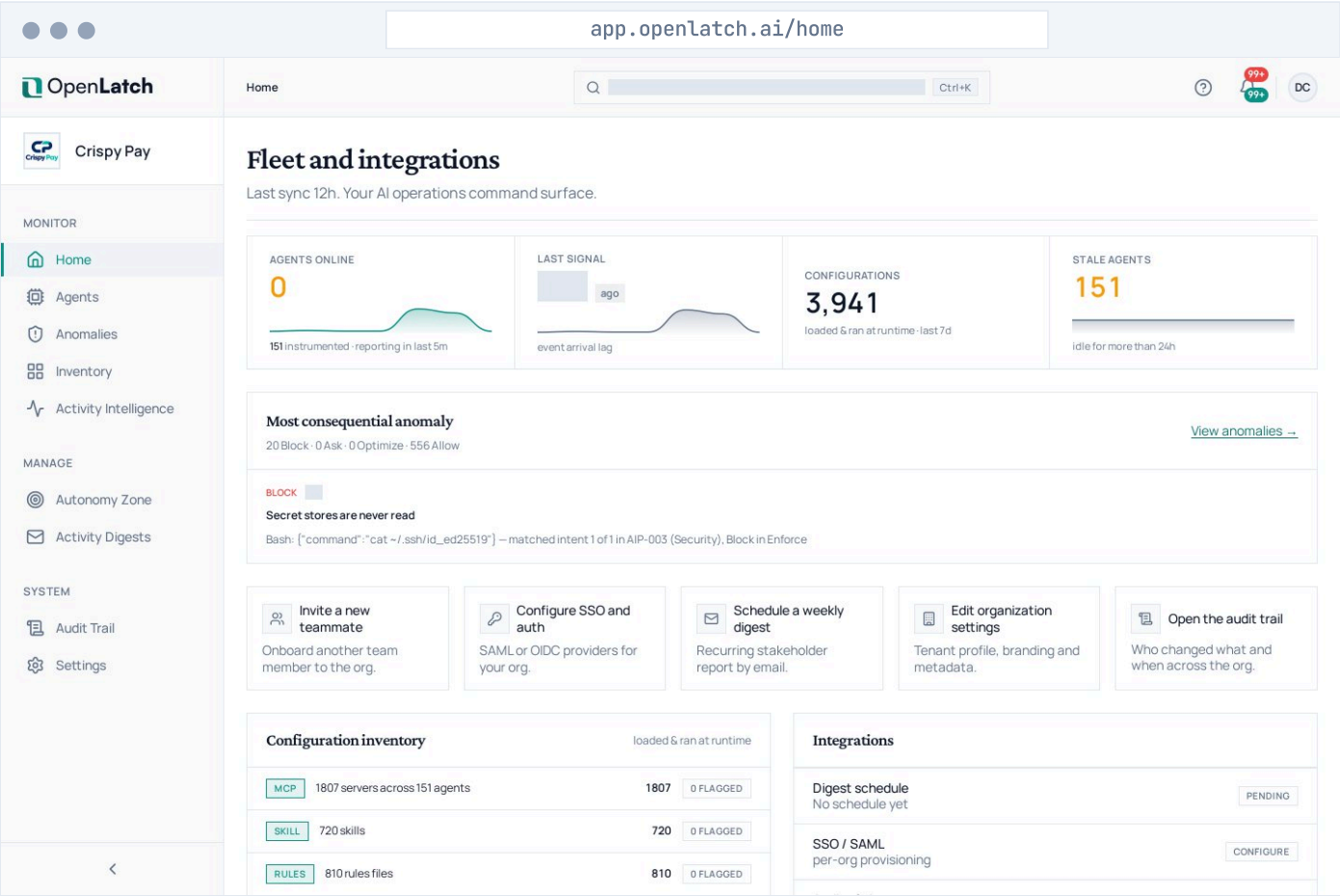
Avant le premier événement

Juste après la connexion d'un client, la page affiche **Awaiting first agent activity** : le client est connecté et rien ne s'est encore produit. Le premier événement apparaît dès qu'un agent invoque un outil. Si rien n'apparaît après que l'agent a travaillé un moment, lancez `openLatch doctor` sur la machine ; voir [Vérifier et déployer](#).

Pour l'historique complet d'un agent plutôt que le flux en direct, utilisez ses onglets ; voir [Enquêter sur ce qu'a fait un agent](#).

Lire la page d'accueil

La page **Home** est celle sur laquelle la console s'ouvre après la connexion. Elle résume l'état de votre organisation et indique ce qui mérite un coup d'œil. Ouvrir `app.openlatch.ai` sans chemin vous amène ici.



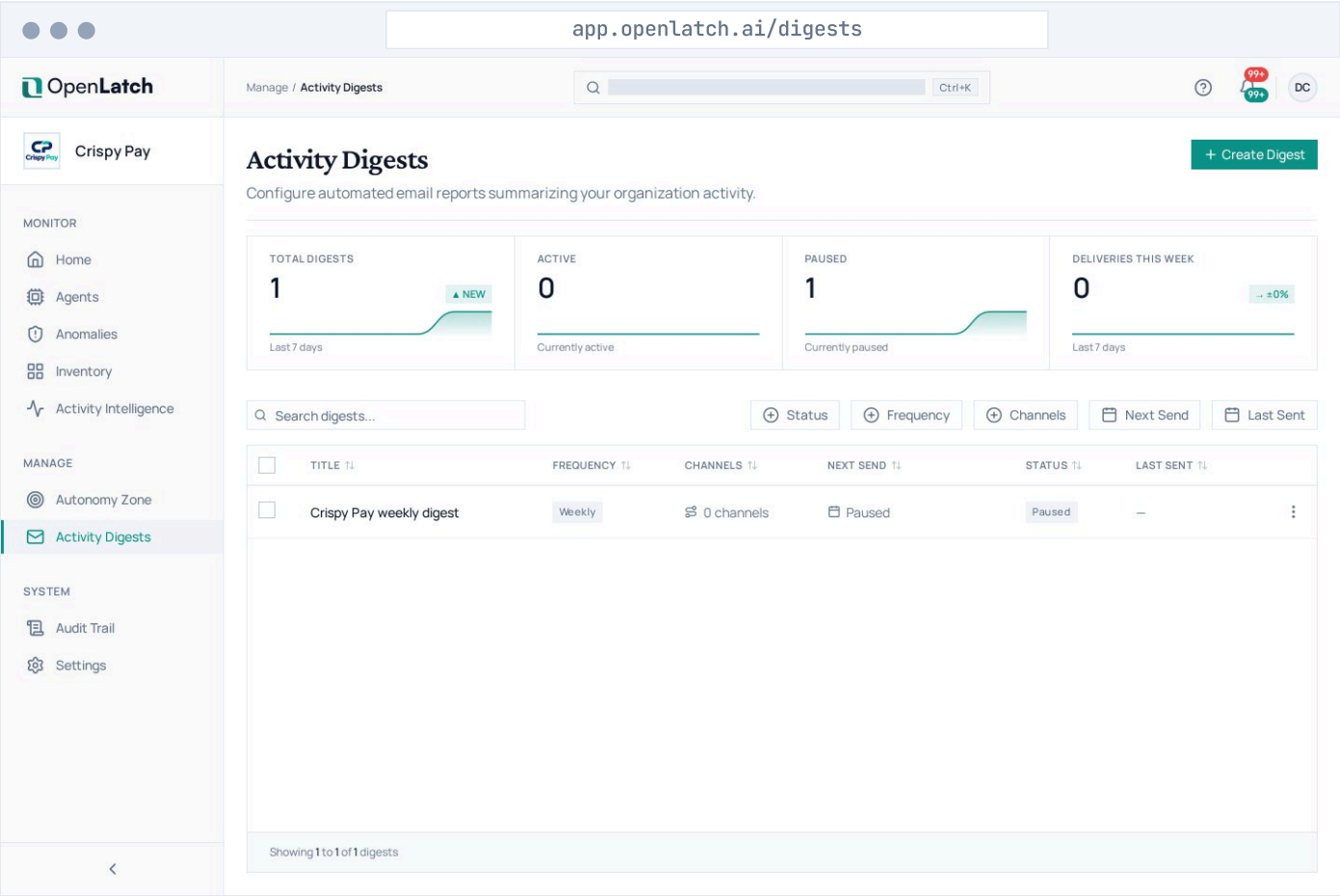
La page Home : les agents qui remontent des données, ce qui a changé depuis votre dernière visite, et ce qu'il faut regarder ensuite.

Panneau	Ce qu'il vous dit	Où il mène
Fleet health	Combien de vos agents remontent des données	Trouver et lire un agent
Configuration inventory	Combien de sources de configuration (hooks, serveurs MCP, fichiers de règles, skills) ont été ajoutées, modifiées ou supprimées depuis votre dernière visite	Rechercher dans l'inventaire
Integrations	Vos plannings de synthèses actifs	Planifier des synthèses d'activité
Single sign-on	L'état de vos méthodes de connexion. Le lire demande un rôle d'administrateur	Configurer l'authentification unique
Recent changes	Les dernières entrées du journal d'audit	Consulter le journal d'audit
Team	Combien de membres compte l'organisation	Gérer les utilisateurs et les équipes
Most active users	Les personnes dont les agents ont fait le plus d'actions sur les sept derniers jours	Gérer les utilisateurs et les équipes
If you have 5 minutes	Une courte liste de choses à faire, comme examiner une source de configuration modifiée, connecter un autre agent ou mettre en place une synthèse, ou une mention que tout est à jour	La page de chaque étape

La liste **Since you last visited** nomme chaque changement de configuration depuis votre visite précédente. Ouvrez-en un pour le lire dans l'inventaire. **Reopen setup** vous ramène à l'assistant de configuration de l'organisation.

Planifier des synthèses d'activité

Utilisez **Digests** pour envoyer par e-mail, selon un calendrier, un rapport sur l'activité des agents de votre organisation, à des personnes qui ont ou non un compte dans la console. Il vous faut le rôle editor ou supérieur pour créer ou modifier une synthèse.



Digests : chaque planning avec sa fréquence, ses canaux, son prochain envoi et son état.

Créer une synthèse

- 1 Ouvrez **Digests** et choisissez **Create Digest**.
- 2 Donnez-lui un titre et un objet. L'objet peut utiliser {period} et {organization_name}.
- 3 Sous **Notification Routing**, sélectionnez votre canal Email. Le champ des destinataires n'apparaît qu'une fois un canal Email sélectionné.

- 4 Ajoutez les destinataires, jusqu'à 100 adresses e-mail. Les destinataires n'ont pas besoin d'un compte dans la console.
- 5 Choisissez la fréquence, quotidienne, hebdomadaire, mensuelle ou trimestrielle, puis le jour, l'heure d'envoi et le fuseau horaire.
- 6 Choisissez les sections de contenu que porte le rapport, et enregistrez.

Les synthèses sont envoyées par le canal e-mail ; configurez-le d'abord dans [Configurer les canaux de notification](#).

Gérer les synthèses

La liste montre la fréquence, les canaux, le prochain envoi, le dernier envoi et l'état de chaque synthèse. Cherchez-y, ou filtrez par état, fréquence, canal ou date d'envoi.

Pour	Faites ceci
Arrêter une synthèse un temps	Mettez-la en pause. Son état indique Paused jusqu'à ce que vous la réactiviez
La modifier	Ouvrez-la et modifiez ses informations
Lire ce qu'elle a envoyé	Ouvrez-la, puis son onglet Generated Reports

Lire un rapport

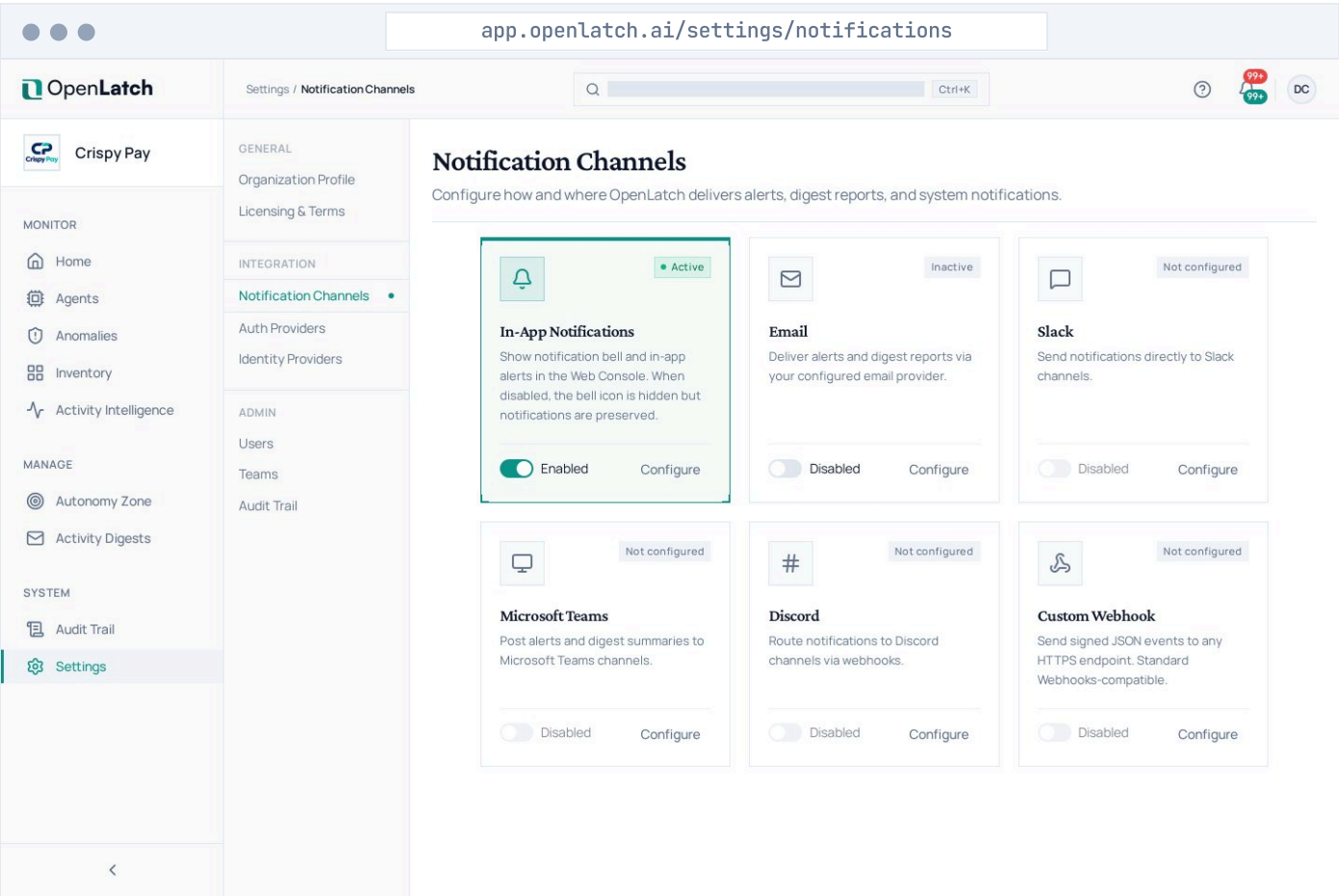
Un rapport généré s'ouvre sur sa propre page, mis en forme comme il est envoyé : un résumé en langage clair de la période couverte, les chiffres qui le sous-tendent, et une section qui explique comment chaque chiffre a été obtenu et ce que signifie chaque terme.



Un rapport de synthèse : le résumé de la période, les chiffres, et comment chacun a été obtenu.

Configurer les canaux de notification

Utilisez **Settings > Notification Channels** pour choisir comment et où OpenLatch délivre les notifications et les rapports de synthèse. Il vous faut le rôle admin ou supérieur.



Notification Channels : chaque canal vers lequel OpenLatch peut délivrer, et s'il est configuré.

Canaux

Canal	Ce qu'il fait
Notifications dans l'application	Affiche la cloche de notifications dans la console. La désactiver masque la cloche ; les notifications sont conservées
E-mail	Délivre les notifications et les rapports de synthèse par le fournisseur d'e-mail configuré pour votre site OpenLatch
Slack	Envoie les notifications vers des canaux Slack
Microsoft Teams	Publie les notifications et les résumés de synthèse dans des canaux Microsoft Teams
Discord	Achemine les notifications vers des canaux Discord par des webhooks
Webhook personnalisé	Envoie des événements JSON signés vers n'importe quel point d'accès HTTPS, compatible avec la spécification Standard Webhooks

Les canaux Slack, Microsoft Teams, Discord et webhook personnalisé peuvent chacun être ajoutés plusieurs fois, par exemple un par équipe.

Ajouter un canal

- 1 Sur la carte du canal, choisissez **Configure**.
- 2 Suivez le guide de configuration du panneau, puis saisissez la destination : une adresse de webhook Slack, Microsoft Teams ou Discord, ou votre propre point d'accès HTTPS.
- 3 Choisissez les notifications que reçoit le canal, et enregistrez.
- 4 Envoyez un test, et vérifiez qu'il arrive.

Vérifier les signatures de webhook

Un webhook personnalisé signe chaque événement avec un secret de signature, pour que votre point d'accès puisse vérifier qu'un événement vient d'OpenLatch. Le secret est affiché une seule fois, quand vous le révélez pour le copier dans la configuration de votre point d'accès. **Rotate signing secret** en émet un nouveau ; mettez votre point d'accès à jour en même temps.

Pour envoyer un rapport récurrent plutôt que des notifications individuelles, voir [Planifier des synthèses d'activité](#).

Consulter le journal d'audit

Utilisez la page **Audit Trail** pour consulter chaque changement et chaque action de votre organisation : qui a fait quoi, sur quoi, et quand. Le journal est en ajout seul et chaîné par hachage : une entrée modifiée ou supprimée rompt la chaîne. Ses réglages, sous **Settings > Audit Trail**, demandent le rôle admin ou supérieur.

OpenLatch

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

app.openlatch.ai/audit-trail

Manage / Audit Trail

Q

Ctrl+K

?

99+

99+

DC

Audit Trail

Review all changes and actions across your organization.

Q Search by actor, target, or changes..

Source

Action

Target Type

Status

Date Range

	TIMESTAMP	ACTOR	SOURCE	ACTION	TARGET	STATUS	CHANGES
☐		User Documentation user-doc@openlatch.ai	Auth	Login	Session user-doc@openlatch.ai	Success	
☐		Docs Capture Owner docs-capture@crispypayex...	Auth	Login	Session docs-capture@crispyp...	Success	
☐		Docs Capture Owner docs-capture@crispypayex...	Auth	Login	Session docs-capture@crispyp...	Success	
☐		Luc Delsalle luc.delsalle@openlatch.ai	API	Directory Provid	Directory Provider Corporate Dir...	Success	
☐		Luc Delsalle luc.delsalle@openlatch.ai	API	Directory Provid	Directory Provider Corporate Dir...	Success	
☐		User Documentation user-doc@openlatch.ai	Auth	Login	Session user-doc@openlatch.ai	Success	
☐		Docs Capture Owner docs-capture@crispypayex...	Auth	Login	Session docs-capture@crispyp...	Success	
☐		Docs Capture Owner docs-capture@crispypayex...	Auth	Login	Session docs-capture@crispyp...	Success	
☐		Luc Delsalle					

Showing 1 to 25 of 815 events

1 2 ... 33

La page Audit Trail : chaque changement et chaque action, avec son auteur, sa cible et son état.

Trouver un événement

- 1
- Ouvrez **Audit Trail**.
- 2
- Cherchez par auteur, cible ou changement, ou filtrez par source, action, type de cible, état et période.

- 3 Ouvrez un événement. Son détail montre l'auteur, le contexte, l'état, chaque champ modifié avec son ancienne et sa nouvelle valeur, et la place de l'événement dans la chaîne d'intégrité.

Chaque changement de politique y figure avec son auteur et l'état avant et après, de même que chaque réponse donnée à un Ask.

L'exporter

Choisissez **Export** pour télécharger les événements qui correspondent à votre recherche et à vos filtres, en CSV ou JSON. L'export lui-même est enregistré comme événement d'audit.

Définir la conservation et vérifier l'intégrité

Ouvrez **Settings > Audit Trail**.

The screenshot shows the OpenLatch web interface at the URL `app.openlatch.ai/settings/audit-trail`. The left sidebar contains a navigation menu with sections: MONITOR (Home, Agents, Anomalies, Inventory, Activity Intelligence), MANAGE (Autonomy Zone, Activity Digests), and SYSTEM (Audit Trail, Settings). The 'Settings' option is currently selected. The main content area is titled 'Audit Trail' and includes a subtitle: 'Configure retention policies, data management, and compliance settings for the audit trail.' The settings are organized into three sections: 1. 'Retention Policy' with a 'Retention Period' section showing options for 90 days, 180 days, 365 days (recommended), 730 days, and Custom. Below this is an 'Auto-Purge' toggle switch which is turned on. 2. 'Export Preferences' with a section to 'Choose the default format for audit trail exports', offering 'CSV - Spreadsheet-compatible, human-readable' (selected) and 'JSON - Machine-readable, preserves nested data'. 3. 'Hash Chain Integrity' with a 'Verify Integrity' button and a description: 'Verify the cryptographic hash chain that ensures audit events have not been tampered with.' At the bottom right of the settings area are 'Cancel' and 'Save Configuration' buttons.

Les réglages Audit Trail : conservation, purge automatique, format d'export et vérification d'intégrité.

Réglage	Ce qu'il fait
Durée de conservation	Combien de temps les événements sont conservés : 90, 180, 365 ou 730 jours, ou une durée personnalisée. La console recommande 365 jours
Purge automatique	Activée, elle supprime toutes les 24 heures les événements plus anciens que la durée de conservation
Format d'export	Le format par défaut d'un export : CSV, lisible dans un tableur, ou JSON, qui conserve les données imbriquées
Verify integrity	Vérifie la chaîne de hachage de chaque événement. Une purge rompt la chaîne là où elle a supprimé des événements, et la vérification signale cette rupture

La même page porte deux actions irréversibles : l'anonymisation du journal, qui remplace l'e-mail, le nom et l'identifiant utilisateur de l'acteur de chaque événement par des valeurs anonymisées tout en conservant les enregistrements et la chaîne (les adresses IP et les autres détails de chaque événement restent tels qu'enregistrés), et la suppression de tous les événements d'audit, qui rompt la chaîne. Chacune vous demande de saisir un mot de confirmation.

Le journal d'audit est le seul enregistrement dont vous définissez la conservation. [Traitement des données et sécurité](#) liste la conservation fixe de tous les autres.

Gérer les utilisateurs et les équipes

Utilisez **Settings** > **Users** pour décider qui peut utiliser la console et ce que chacun peut faire, et **Settings** > **Teams** pour regrouper les personnes. Il vous faut le rôle admin ou supérieur.

Rôles

Quatre rôles se succèdent dans un ordre strict de privilèges, chacun incluant tout ce qui est en dessous :

Rôle	Ce qu'il permet
owner	La propriété complète : toutes les permissions, plus la gestion des équipes et de l'organisation
admin	L'accès complet : utilisateurs, équipes, réglages et tout le contenu
editor	Gère les ressources et la configuration, comme les politiques d'intention, les zones et la propriété des agents
viewer	Accès en lecture seule aux ressources et à la configuration

Chaque personne invitée commence en viewer. Accordez les rôles supérieurs délibérément, aux quelques personnes qui en ont besoin. La **Role Permissions Reference** de la page liste chaque permission par rôle.

Inviter quelqu'un

- 1 Ouvrez **Settings** > **Users** et choisissez **Invite User**.
- 2 Saisissez son adresse e-mail et le rôle avec lequel la personne commence, puis envoyez l'invitation.
- 3 La personne reçoit un e-mail avec un lien d'inscription lié à cette adresse. Vous pouvez aussi copier le lien d'invitation et le partager vous-même.

Une invitation peut expirer ou être révoquée, et chacune ne sert qu'une fois. Toute personne dont l'adresse appartient à l'un des domaines de messagerie autorisés de votre organisation peut aussi rejoindre par l'authentification unique sans invitation ; voir [Configurer l'authentification unique](#).

The screenshot shows the OpenLATCH application interface. The top navigation bar displays the URL `app.openlatch.ai/settings/users`. The left sidebar contains a menu with sections: MONITOR (Home, Agents, Anomalies, Inventory, Activity Intelligence), MANAGE (Autonomy Zone, Activity Digests), and SYSTEM (Audit Trail, Settings). The 'Settings' option is highlighted. The main content area is titled 'Users' and includes a search bar and filters for Status, Team, Email, Created, and Last Login. A table lists 7 users:

	NAME	EMAIL	TEAMS	ROLE	STATUS	LAST LOGIN	CREATED
☐	DO Docs Capture Owner	docs-cap...	Administr...	Owner	Active		
☐	RK Rohan Kapoor	rohan.ka...	Policy Edi...	Editor	Active		
☐	CL Camille Lefèvre	camille.le...	Policy Edi...	Editor	Active		
☐	MW Marcus Washington	marcus...	Policy Edi...	Editor	Active		
☐	IS Ingrid Solberg	ingrid.sol...	Administr...	Admin	Active		
☐	UD User Documentation	user-doc...	Administr...	Admin	Active		
☐	Luc Delsalle	luc.delsal...	Administr...	Owner	Active		

Showing 1 to 7 of 7 users

Settings > Users : chaque membre avec son rôle et son état.

Modifier, désactiver ou retirer quelqu'un

Ouvrez un utilisateur pour changer son rôle ou ses informations, ou pour réinitialiser son mot de passe. Sélectionnez plusieurs utilisateurs pour les activer, les désactiver ou les supprimer ensemble. Un utilisateur désactivé garde son historique et ne peut plus se connecter. Chaque changement de rôle ou d'appartenance est inscrit dans le journal d'audit.

Organiser les personnes en équipes

Les équipes regroupent des utilisateurs, par exemple par service ou par agents dont ils s'occupent.

- 1 Ouvrez **Settings > Teams** et choisissez **Create Team**.
- 2 Nommez-la, décrivez ce dont elle est responsable, et ajoutez ses membres.

app.openlatch.ai/settings/teams

OpenLatch

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

Settings / Teams

GENERAL

Organization Profile

Licensing & Terms

INTEGRATION

Notification Channels

Auth Providers

Identity Providers

ADMIN

Users

Teams

Audit Trail

Teams

+ Create Team

Organize users into teams with shared permissions.

Q Search by name or description...

+ Role

+ Members

Created

☐	NAME ↑	DESCRIPTION ↑↓	ROLE ↑↓	MEMBERS ↑↓	CREATED ↑↓	
☐	Administrators System	Organization administrators	Admin	4		
☐	Policy Editors	—	Editor	3		
☐	Users System	Default user team	Viewer	0		

Showing 1 to 3 of 3 teams

ROLE PERMISSIONS REFERENCE

Settings > Teams : chaque équipe, sa description et ses membres.

Supprimer une équipe supprime ses appartenances ; les utilisateurs eux-mêmes restent dans l'organisation.

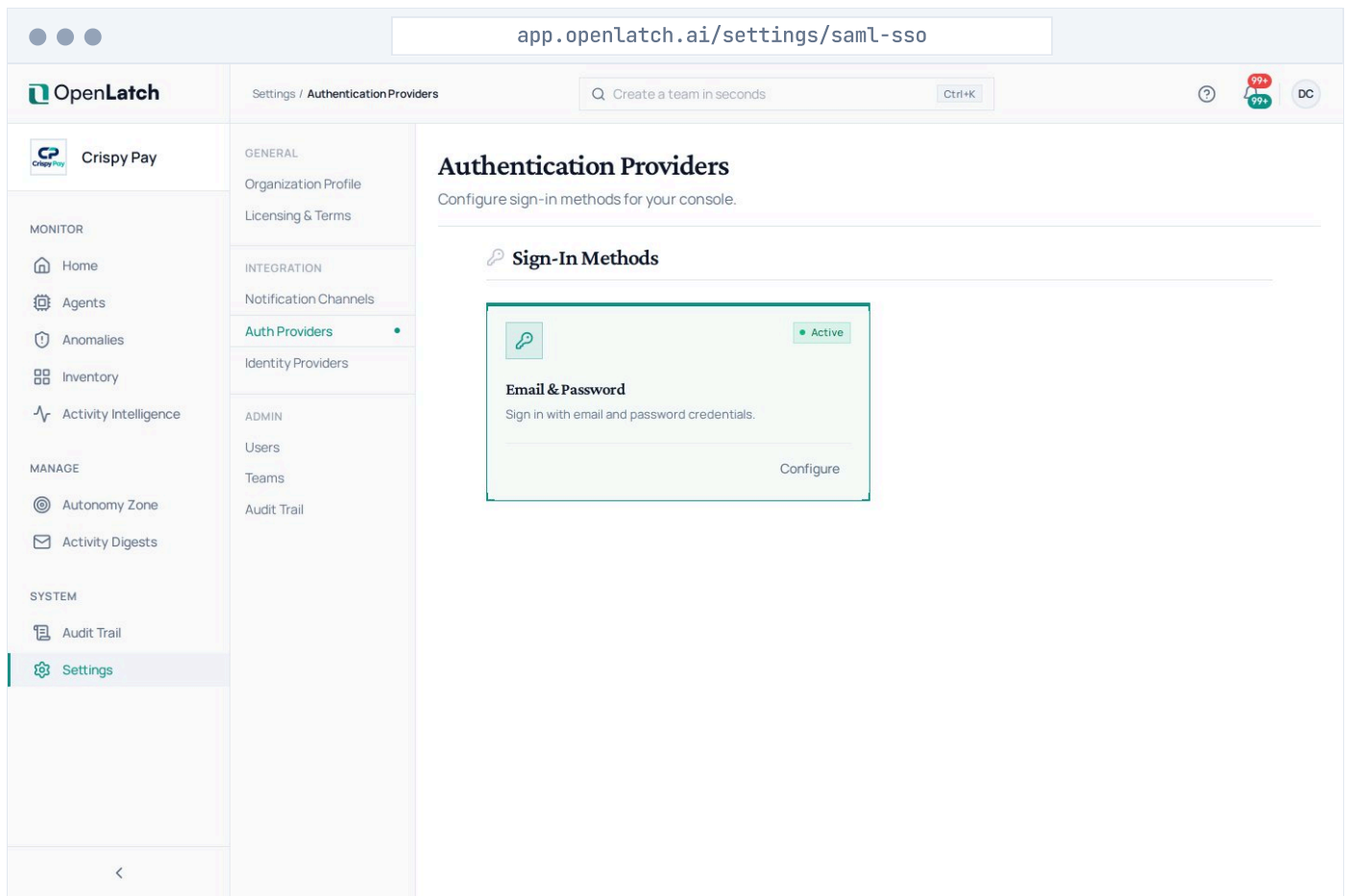
Configurer l'authentification unique

Utilisez **Settings** > **SAML SSO** pour connecter le fournisseur d'identité de votre entreprise, et **Settings** > **Auth Providers** pour consulter les méthodes de connexion de votre organisation et exiger l'authentification à deux facteurs. Il vous faut le rôle admin ou supérieur.

Connecter l'authentification unique SAML

La console prend en charge SAML 2.0, configuré par organisation. Microsoft Entra ID, Okta, PingOne et Active Directory Federation Services se connectent ainsi.

- 1 Ouvrez **Settings** > **SAML SSO**. Copiez les informations du fournisseur de services dans votre fournisseur d'identité quand vous y créez l'application SAML : l'URL de l'Assertion Consumer Service (ACS), l'identifiant d'entité SP et l'URL de métadonnées SP.
- 2 Sélectionnez votre fournisseur d'identité pour obtenir des instructions écrites pour lui, puis saisissez ses informations : son identifiant d'entité, son URL d'authentification, et son certificat de signature ou son URL de métadonnées. L'assertion de votre fournisseur d'identité doit porter l'adresse e-mail et le nom d'affichage de l'utilisateur.
- 3 Ajoutez les domaines de messagerie autorisés. Les personnes dont l'adresse appartient à l'un d'eux peuvent se connecter par SAML. Un domaine ne peut appartenir qu'à une organisation ; la console vous prévient si une autre organisation l'a déjà revendiqué.
- 4 Enregistrez, puis lancez le test de connexion. Il vérifie, sur la configuration enregistrée, que l'URL de métadonnées ou l'URL de connexion de votre fournisseur d'identité répond ; il n'effectue pas de connexion SAML. Connectez-vous ensuite vous-même par SAML, dans une fenêtre de navigation privée, avant de demander à qui que ce soit d'autre de le faire.

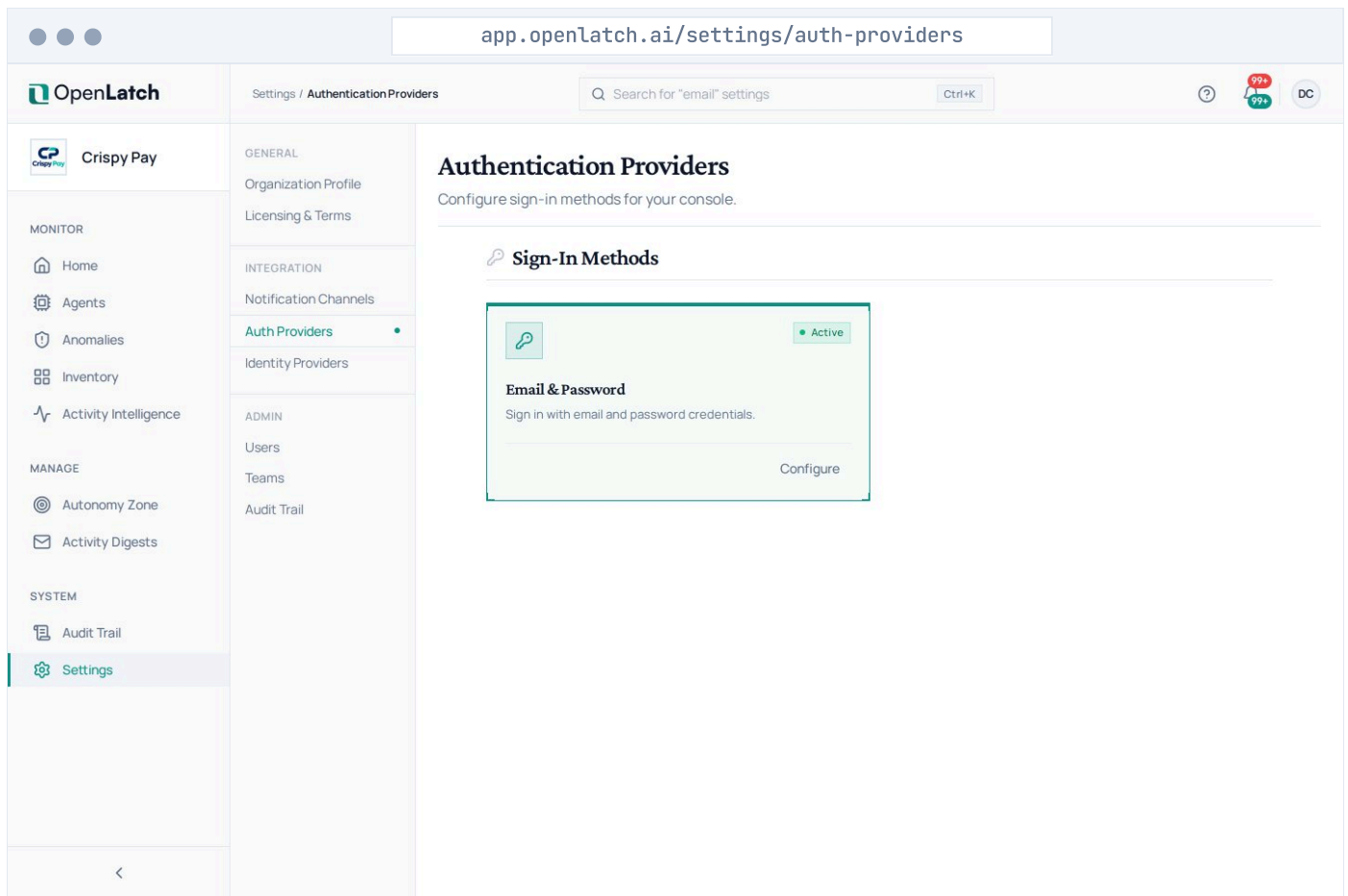


Settings > SAMLSSO : les informations du fournisseur de services à copier, et celles de votre fournisseur d'identité.

Une connexion SAML propre à l'organisation est la seule méthode de connexion qui admet un nouveau membre sans invitation, et uniquement pour les domaines autorisés.

Consulter les méthodes de connexion et exiger l'authentification à deux facteurs

Settings > Auth Providers présente les méthodes de connexion de votre organisation sous forme de cartes, chacune avec son état : **Email & Password**, et **SAML SSO** quand SAML est disponible sur votre site OpenLatch. **Configure** sur une carte ouvre ses réglages.



Settings > Auth Providers : les méthodes de connexion qu'autorise votre organisation.

Carte Configurer ouvre

Email & Password La politique de double authentification. Exigez l'authentification à deux facteurs pour inviter chaque utilisateur à mot de passe qui ne l'a pas à la configurer. Après le délai de grâce que vous choisissez, il est renvoyé vers ses réglages de sécurité à chaque connexion tant qu'il ne l'a pas fait. Les utilisateurs qui se connectent uniquement par un fournisseur externe en sont exemptés. La carte affiche **MFA Required** tant que la politique est active

SAMLSSO La connexion SAML, configurée comme ci-dessus

Chaque personne configure sa propre authentification à deux facteurs depuis son compte ; voir [Gérer votre compte](#).

Connecter un annuaire pour la recherche des propriétaires

Utilisez **Settings > Identity Providers** pour lire votre annuaire du personnel, afin qu'OpenLatch détermine qui est propriétaire de chaque agent et où cette personne se trouve dans votre organisation. La connexion est en lecture seule et ne sert pas à la connexion des utilisateurs, qui relève de [Configurer l'authentification unique](#). Il vous faut le rôle admin ou supérieur pour la modifier.

The screenshot shows the OpenLatch web interface. The top navigation bar includes the OpenLatch logo, the current page path 'Settings / Identity Providers', a search bar, and user avatars. The left sidebar contains a menu with categories: MONITOR (Home, Agents, Anomalies, Inventory, Activity Intelligence), MANAGE (Autonomy Zone, Activity Digests), and SYSTEM (Audit Trail, Settings). The 'Settings' option is highlighted. The main content area is titled 'Identity Providers' and includes a description: 'Read your employee directory to work out who owns each agent. Not used for sign-in.' Below this is a 'Signal capture' section with a toggle switch for 'Collect identity signals from endpoints', which is currently turned 'On'. The 'Directories' section displays five cards for different identity providers: Microsoft Entra ID, Google Workspace, Active Directory, Okta, and PingOne. Each card shows a 'Not configured' status, a brief description of how it reads user data, and 'Enable' and 'Connect' buttons.

Settings > Identity Providers : l'interrupteur de capture des signaux et les annuaires connectés.

La propriété se rapproche par deux côtés, et les deux doivent être présents :

Côté	Ce qu'il apporte	Où
Capture des signaux	Les machines remontent les signaux qui identifient qui travaille : l'utilisateur du système d'exploitation, l'e-mail git et le compte du fournisseur d'IA	L'interrupteur Collect identity signals from endpoints , activé par défaut
Annuaire	L'annuaire transforme ces signaux en une personne nommée, avec son intitulé de poste et sa place dans l'organisation	Les cartes d'annuaire

Si la capture des signaux est désactivée, aucun agent ne peut être rattaché à un propriétaire, aussi bien connecté que soit l'annuaire.

Connecter un annuaire

- 1 Ouvrez **Settings > Identity Providers**.
- 2 Choisissez la carte de votre annuaire : Microsoft Entra ID, Google Workspace, Active Directory, Okta ou PingOne. Chacun est une source en lecture seule : aucun ne crée d'utilisateurs dans OpenLatch, et aucun n'y copie l'annuaire.
- 3 Suivez les étapes du panneau pour accorder à OpenLatch un accès en lecture dans votre annuaire, puis enregistrez. S'il manque à l'annuaire une permission dont OpenLatch a besoin, la carte indique laquelle.
- 4 Vérifiez le résultat sur un agent que vous connaissez : son onglet **Identity** doit nommer son propriétaire. Voir [Trouver et lire un agent](#).

Une organisation sans annuaire connecté est un état pris en charge. Un agent dont le propriétaire ne peut pas être déterminé est enregistré comme non attribué plutôt que deviné, et un editor peut définir son propriétaire à la main depuis son onglet **Identity**.

Ce que l'annuaire apporte aussi

Les unités de l'annuaire forment l'arbre de votre organisation sur la page **Autonomy Zone**, et chaque agent se trouve par défaut dans l'unité de son propriétaire. Une unité issue de l'annuaire garde le nom et la place que lui donne l'annuaire, et est mise à jour à chaque synchronisation ; l'Autonomy Zone que vous y placez reste la vôtre. Voir [Gérer les Autonomy Zones](#).

Déconnecter un annuaire

Déconnecter un annuaire supprime sa connexion, ses identifiants enregistrés et les personnes qu'il avait déterminées. Les agents rattachés par lui perdent leur propriétaire jusqu'à ce qu'un autre annuaire les détermine.

L'adresse `/settings/directory-sync` est un ancien nom de cette page et l'ouvre.

Modifier le profil de l'organisation

Utilisez **Settings** > **Organization Profile** pour gérer l'apparence de votre organisation et les valeurs par défaut qui s'appliquent à toute l'organisation. Ouvrir **Settings** vous amène sur cette page. Il vous faut le rôle admin ou supérieur pour la modifier.

app.openlatch.ai/settings/organization-profile

OpenLatch

Settings / Organization Profile

Q Go to Organization Profile

Ctrl+K

?

99+

99+

DC

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

GENERAL

Organization Profile

Licensing & Terms

INTEGRATION

Notification Channels

Auth Providers

Identity Providers

ADMIN

Users

Teams

Audit Trail

Organization Profile

Manage your organization's public identity and primary domain.

Identity

Organization Logo

Recommended size 400x400px

Crispy Pay

Change Logo

Remove

Organization Name

Crispy Pay

The legal name of the entity operating this property.

System Metadata

ORGANIZATION ID

208bdd63-67fd-49e5-9bdf-9c6d55dd7f72

ORGANIZATION SLUG

crispy-pay

CREATED AT

SITE

us-east-1 (United States)

Your organization's data is stored in this region. OpenLatch's residency statement is available on request, in the trust pack and the DPA.

Agent Budgets

Reset to Defaults

Save Changes

Settings > Organization Profile : l'identité de l'organisation, ses métadonnées système et sa configuration.

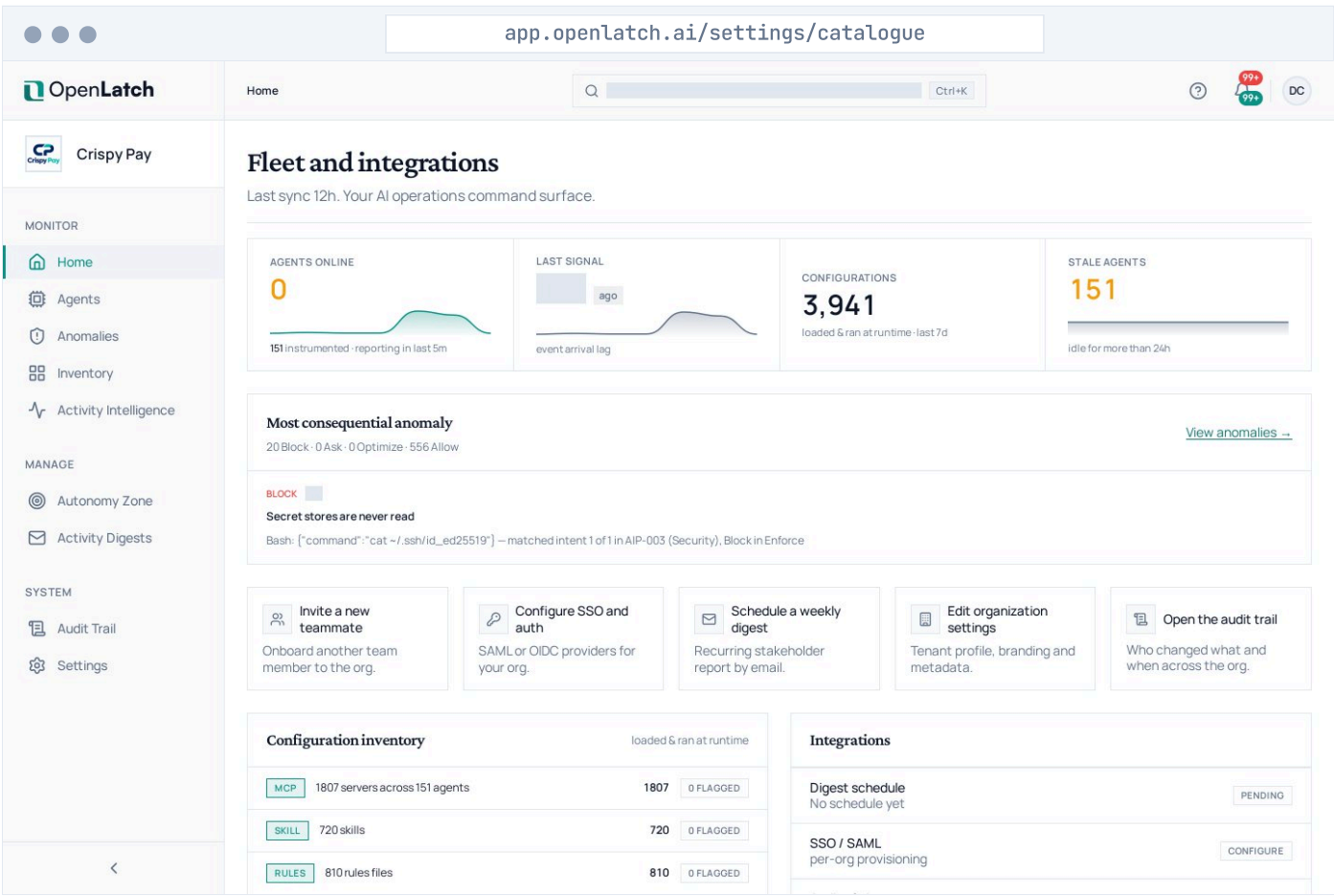
Section	Ce que vous pouvez faire
Identité	Changer le nom de l'organisation, la raison sociale de l'entité qu'elle représente, et téléverser ou retirer son logo, idéalement de 400×400 pixels. Ce nom est celui que porte votre journal d'audit
Métadonnées système	Lire l'identifiant de l'organisation, son slug, sa date de création, et son site : la région où sont stockées les données de votre organisation. Indiquez l'identifiant quand vous contactez le support
Budgets des agents	Définir un objectif de dépense mensuel par défaut pour chaque agent qui n'a pas de budget propre. Ce défaut s'applique à chacun de ces agents séparément : un défaut de 500 \$ donne à chacun 500 \$ par mois, pas 500 \$ partagés. L'effacer n'écrase jamais un budget défini sur un agent
Configuration	Relancer l'assistant de configuration pour revoir la connexion et le profil de l'organisation

Enregistrez vos changements avec **Save Changes**. Le budget propre d'un agent se définit depuis son onglet **Economics** ; voir [Suivre les dépenses d'un agent](#).

Les autres pages de réglages ont chacune leur guide : [utilisateurs et équipes](#), [authentification unique](#), [fournisseurs d'identité](#), [canaux de notification](#), le [journal d'audit](#), le [catalogue](#) et les [licences](#).

Ajuster le catalogue

Utilisez **Settings > Catalogue** pour voir et ajuster la façon dont OpenLatch reconnaît ce que fait une action d’agent et ce qu’elle touche. Les politiques d’intention s’écrivent en termes de ce que font les actions, comme lire un identifiant ou pousser du code ; le catalogue est ce qui rattache un outil, un programme ou une cible concrets à ces termes. Tous les rôles peuvent le lire ; seul un admin peut ajouter ou modifier des entrées.



Settings > Catalogue : les outils, programmes et cibles qu’OpenLatch reconnaît, avec l’effet et la source de chaque entrée.

Lire le catalogue

Le catalogue a trois onglets :

Onglet	Ce que reconnaissent ses entrées
Tools	Les outils qu'appelle un agent
Programs	Les programmes qu'un agent lance dans un shell
Targets	Ce que touche une action, comme un fichier, un hôte ou un dépôt

Chaque entrée montre sa clé, son effet, sa source et sa date d'ajout. Les entrées viennent de plusieurs sources : la base d'OpenLatch, l'apprentissage à partir des actions de vos agents, la rédaction de politiques, ou un admin. **Needs review** filtre les entrées qui attendent une décision, et **Show OpenLatch seed** ajoute à la liste les entrées de base, en lecture seule.

Les entrées apprises s'appliquent à une politique d'intention selon le mode de couverture de cette politique ; voir [Rédiger une Agent Intent Policy](#). Un catalogue vide se remplit à mesure qu'OpenLatch apprend les outils, programmes et cibles qu'utilisent vos agents.

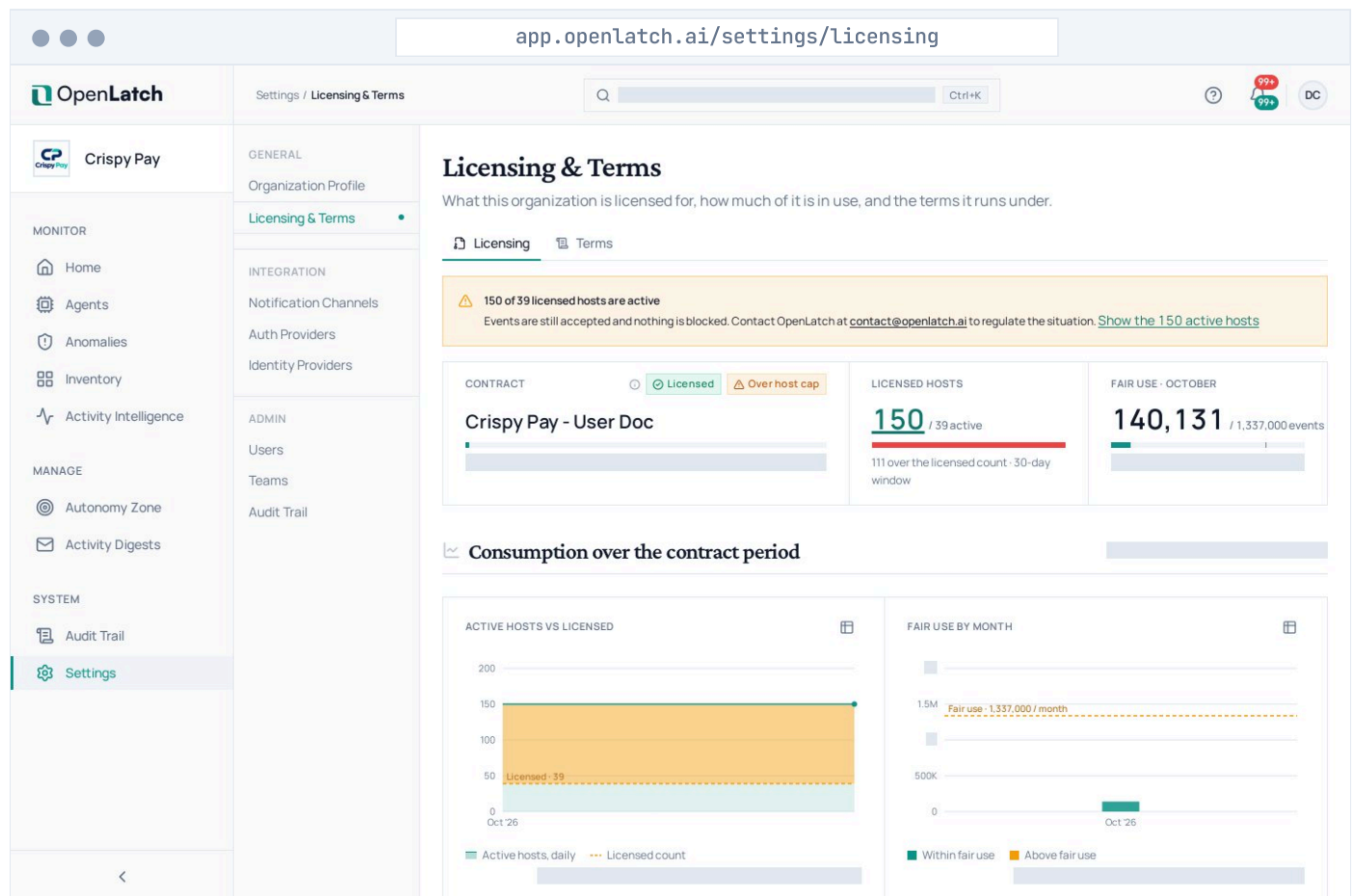
Ajuster une entrée

Pour	Faites ceci
Ajouter ce qu'OpenLatch ne reconnaît pas encore	Choisissez Add entry , et indiquez la clé et son effet
Changer le sens d'une entrée apprise	Choisissez Override sur l'entrée
Rendre une entrée de base plus stricte pour votre organisation	Choisissez Narrow dessus. La base elle-même est en lecture seule ; votre entrée restreint sa clé

Lire les licences et l'usage

Utilisez **Settings** > **Licensing** pour voir ce pour quoi votre organisation dispose d'une licence, la part utilisée, et les conditions qui s'appliquent. Retirer un hôte demande le rôle admin ou supérieur.

Votre licence compte des hôtes : les machines qui exécutent le client. Chaque licence porte aussi une allocation mensuelle d'usage raisonnable, en événements.



Settings > Licensing : le contrat, les hôtes sous licence utilisés et l'usage raisonnable du mois.

Lire les compteurs

L'onglet **Licensing** s'ouvre sur trois compteurs :

Compteur	Ce qu'il montre
Contrat	La période de vos conditions, et la part écoulée
Hôtes sous licence	Combien d'hôtes sont actifs, c'est-à-dire vus dans les 30 derniers jours, par rapport au nombre sous licence
Usage raisonnable	Les événements utilisés ce mois calendaire, par rapport à l'allocation mensuelle. Il repart à zéro au début de chaque mois, en UTC

Un bandeau n'apparaît que lorsque quelque chose demande votre attention : les conditions n'ont pas encore commencé, le contrat se termine bientôt, plus d'hôtes sont actifs que de licences, ou l'usage raisonnable du mois atteint 100 %. Dépasser le nombre d'hôtes sous licence ou l'allocation d'usage raisonnable ne bloque rien ; les événements restent acceptés, et le bandeau vous invite à contacter OpenLatch à contact@openlatch.ai. Les dates du contrat sont différentes. Avant le début de vos conditions, la console est disponible mais les événements du client sont refusés. Une fois le contrat terminé, les événements du client sont refusés et la console est restreinte jusqu'au renouvellement des conditions ; contactez OpenLatch à contact@openlatch.ai avant la date de fin pour renouveler.

Sous les compteurs, les graphiques montrent les hôtes actifs par rapport au nombre sous licence et l'usage raisonnable mois par mois sur la période du contrat, et l'onglet **Terms** contient les conditions elles-mêmes.

Trouver un hôte

Le tableau des hôtes sous licence liste chaque hôte avec son état, son système d'exploitation, son propriétaire et sa dernière activité. Filtrez-le par état, système d'exploitation, propriétaire ou dernière activité, ou cherchez-y. Ouvrez un hôte pour voir les agents qui y tournent.

Retirer un hôte

Retirez une machine qui n'exécute plus le client, pour qu'elle cesse de compter parmi vos hôtes sous licence.

- 1 Sélectionnez l'hôte, ou plusieurs hôtes, dans le tableau.
- 2 Choisissez **Retire host** et confirmez.

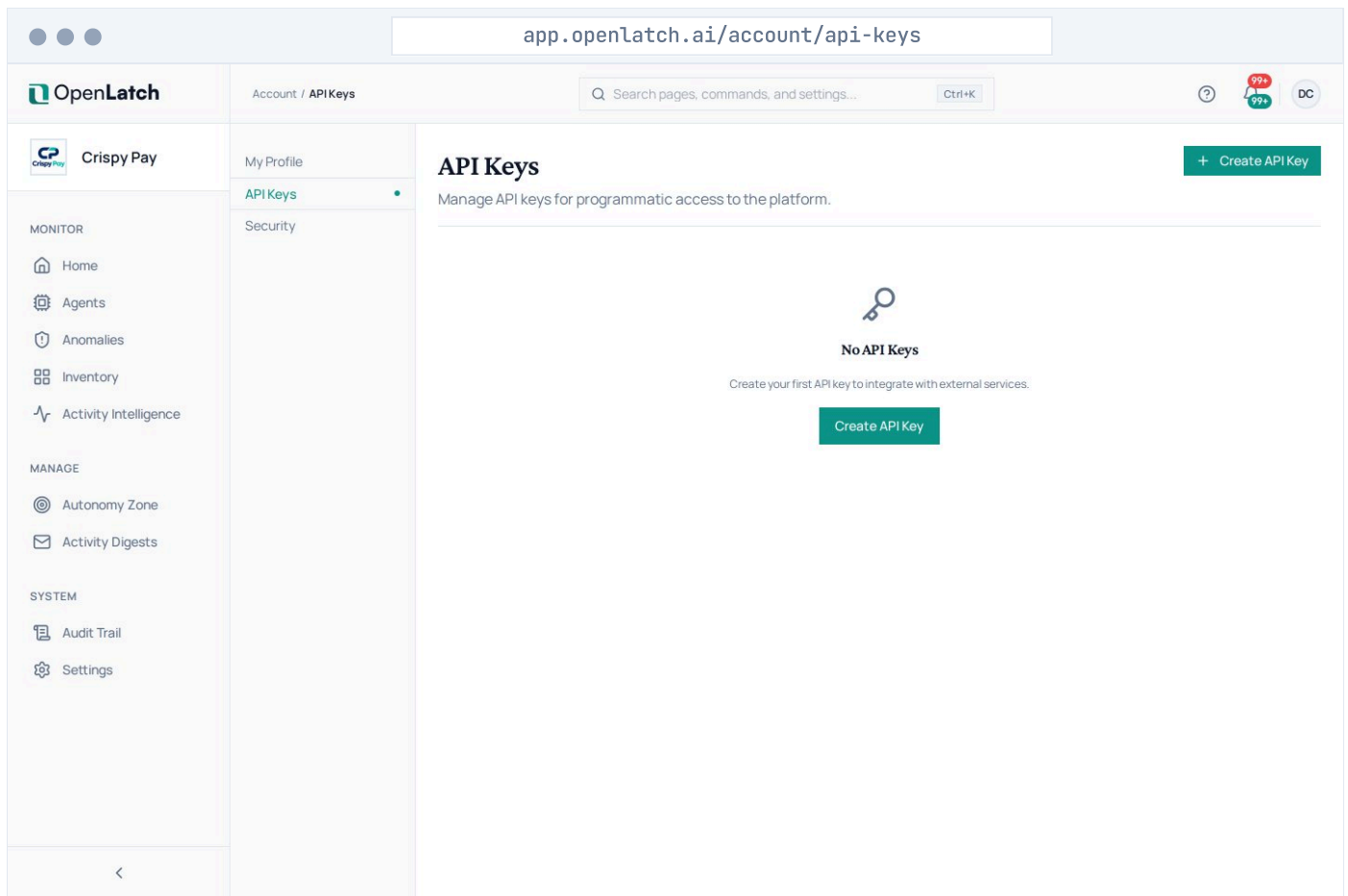
Un hôte retiré cesse de compter immédiatement. Son historique d'événements est conservé, et le retrait est inscrit dans le journal d'audit avec votre nom. Si l'hôte est vu de nouveau, il compte de nouveau automatiquement. Pour retirer le client d'une machine, voir [Exploiter](#).

Gérer les clés d'API

Utilisez **Account > API Keys** pour créer les clés qui donnent à un programme l'accès à OpenLatch en votre nom. Une clé sert à deux usages :

Usage	Comment la clé est fournie
Inscrire une machine sans connexion par navigateur, pour une installation gérée ou sans surveillance	Sous le nom OPENLATCH_API_KEY dans l'environnement de l'installation. Voir Déployer le client
Appeler l'API REST publique d'OpenLatch, sous <code>/api/public/v1</code> sur <code>app.openlatch.ai</code>	Dans l'en-tête <code>Authorization: Bearer</code> de chaque requête

Une clé agit avec votre rôle dans l'organisation. Une clé créée par un editor ne peut pas modifier un réglage d'identité, et révoquer la clé ou abaisser votre rôle ferment la même porte.



Account > API Keys : chaque clé avec son nom, sa valeur masquée, son état, sa date de création et sa dernière utilisation.

Créer une clé

- 1 Ouvrez la page **API Keys** de votre compte et choisissez **Create API Key**.
- 2 Nommez la clé d'après ce qui l'utilisera, par exemple la vague de déploiement ou l'intégration, et choisissez **Create Key**.
- 3 Copiez la valeur de la clé maintenant. Elle est affichée une seule fois, en entier ; après fermeture, la liste ne montre qu'une valeur masquée.
- 4 Conservez-la dans le coffre de secrets de votre organisation. Ne l'écrivez jamais dans un script, un paquet, une ligne de commande ou un document partagé.

Utilisez une clé par usage : une par vague de déploiement, une par intégration. Une clé que vous pouvez révoquer seule est une clé que vous pouvez révoquer sans rien perturber d'autre.

Révoquer une clé

Choisissez **Revoke** sur la clé et confirmez. La révocation est définitive, et tout ce qui utilise la clé perd l'accès immédiatement. Une machine inscrite avec une clé de vague enregistre cette clé et s'authentifie avec elle : révoquer la clé déconnecte la machine. Donnez d'abord une nouvelle clé à ces machines : définissez-la comme `OPENLATCH_API_KEY` et relancez `openlatch init --yes` sur chacune.

La colonne **Last used** vous indique si une clé est encore présentée avant de la révoquer. Une clé arrivée à expiration apparaît comme **Expired**.

 Attention

Une clé passée en ligne de commande atterrit dans l'historique du shell, la liste des processus et les journaux de gestion de configuration, et le client refuse un identifiant passé ainsi. Fournissez une clé par l'environnement, depuis votre coffre de secrets.

Gérer votre compte

Les réglages de votre compte vous appartiennent : tous les rôles peuvent les modifier. Ouvrez-les depuis votre avatar dans la console ; **Account** s'ouvre sur votre profil.

Modifier votre profil

Account > Profile contient vos informations personnelles : votre nom d'affichage et votre adresse e-mail. Changer d'adresse e-mail demande votre mot de passe actuel.

The screenshot shows the 'My Profile' page in the OpenLatch application. The browser address bar displays 'app.openlatch.ai/account/profile'. The page layout includes a sidebar on the left with the OpenLatch logo and navigation links under 'MONITOR' (Home, Agents, Anomalies, Inventory, Activity Intelligence), 'MANAGE' (Autonomy Zone, Activity Digests), and 'SYSTEM' (Audit Trail, Settings). The main content area is titled 'My Profile' and contains the following sections:

- Personal Information:** A section with a profile picture placeholder labeled 'DC' and an 'UPLOAD NEW' button. It includes input fields for 'Display Name' (containing 'Docs Capture Owner') and 'Email Address' (containing 'docs-capture@crispypay.example', marked as 'Verified').
- Change Email Address:** A section with a 'Change Email' button and a note: 'docs-capture@crispypay.example Your verified email address.'
- My Teams:** A section showing the user's 'Overall Role' as 'Owner' and a link to 'View Team Settings'. Below this, there is a list of teams, including 'Administrators' with an 'Admin' role.

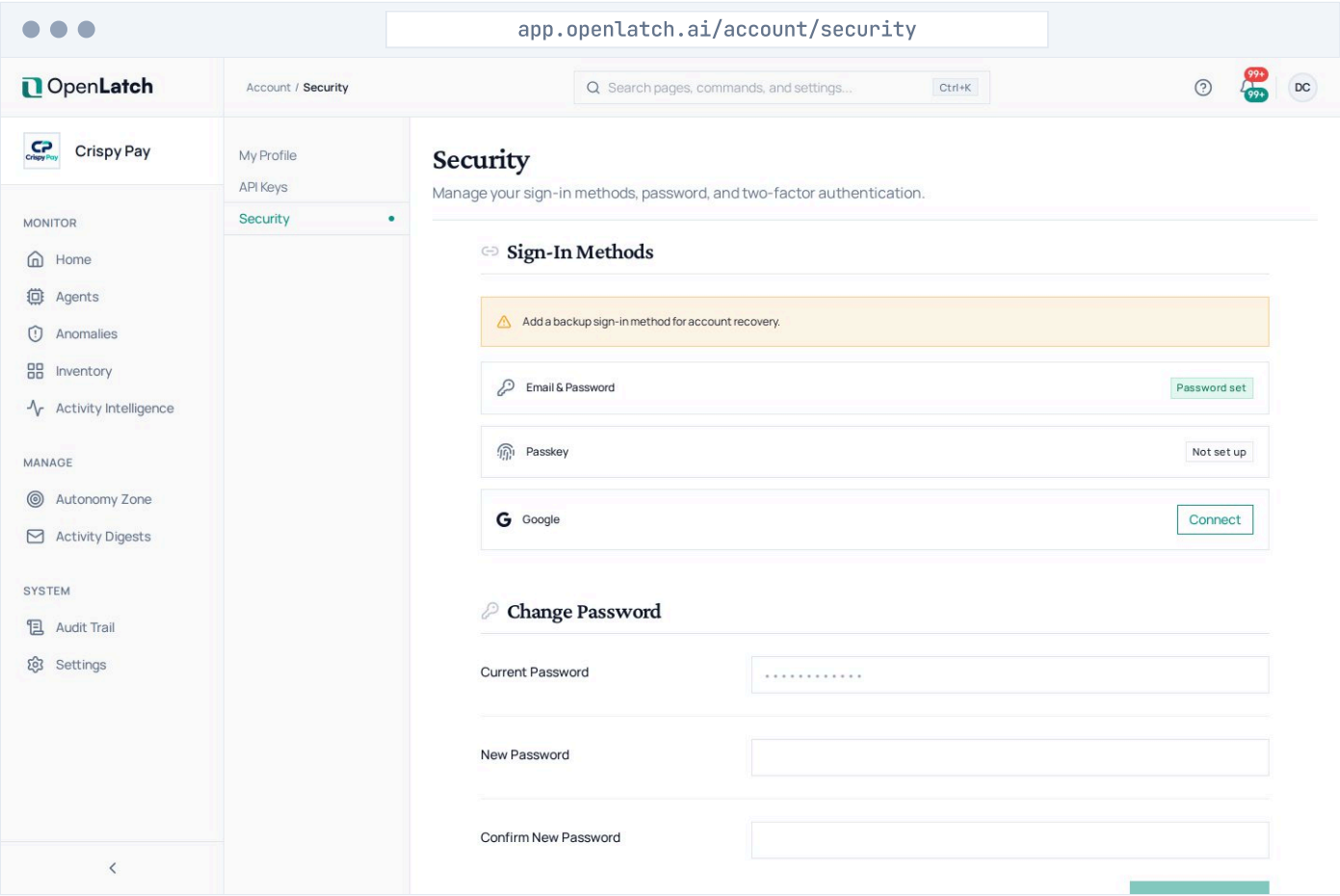
At the bottom right of the page, there are 'Cancel' and 'Save Changes' buttons.

Account > Profile : votre nom d'affichage et votre adresse e-mail.

Votre compte liste aussi vos sessions de connexion actives sur vos appareils. Déconnectez toute session que vous ne reconnaissez pas.

Sécuriser votre connexion

Account > Security gère vos méthodes de connexion, votre mot de passe et l'authentification à deux facteurs.



Account > Security : vos méthodes de connexion, votre mot de passe, l'authentification à deux facteurs et les passkeys.

Réglage	Ce qu'il fait
Mot de passe	Changer votre mot de passe ; la console demande d'abord l'actuel
Authentification à deux facteurs	Ajoute un code d'une application d'authentification à chaque connexion par mot de passe
Passkeys	Se connecter avec une passkey conservée sur votre appareil ou dans votre gestionnaire de mots de passe. Nommez chaque passkey pour les distinguer
Supprimer le compte	Supprime votre compte. La console demande votre mot de passe pour confirmer

Pour configurer l'authentification à deux facteurs :

- 1 Dans **Account > Security**, lancez la configuration de l'application d'authentification et confirmez votre mot de passe.
- 2 Scannez le QR code avec votre application d'authentification.
- 3 Conservez les codes de récupération en lieu sûr. Chaque code ne sert qu'une fois, et c'est ainsi que vous vous connectez si vous perdez l'appareil.
- 4 Saisissez un code à six chiffres de l'application pour terminer.

Si votre organisation exige l'authentification à deux facteurs, la console vous renvoie vers cette page tant que vous ne l'avez pas configurée ; voir [Configurer l'authentification unique](#). Les méthodes de connexion que vous pouvez utiliser relèvent du choix de votre organisation.

Les clés destinées aux programmes et à l'inscription de machines se trouvent dans **Account > API Keys** ; voir [Gérer les clés d'API](#).

Déployer OpenLatch

Cette section vous mène de zéro à un déploiement opérationnel : créer l'organisation, connecter votre fournisseur d'identité, déployer le client sur chaque machine qui exécute un agent, et préparer la console pour les personnes qui l'utiliseront. Elle est écrite pour que vous puissiez faire le travail sans appel.

Ce que couvre la section

Vous voulez

Lisez

Comprendre ce qu'est OpenLatch et ce qui tourne où

[Introduction](#) · [Architecture](#)

Préparer : ce qu'il faut avoir sous la main, et l'unique destination sortante à autoriser

[Prérequis](#) · [Exigences réseau](#)

Installer : l'organisation, le client sur chaque machine, la vérification et le déploiement progressif

[Configurer l'organisation](#) · [Déployer le client](#) · [Vérifier et déployer](#)

Le faire fonctionner dans la durée

[Exploiter](#)

Répondre à une revue de sécurité, et aux questions qui suivent

[Traitement des données et sécurité](#) · [FAQ du déploiement](#)

Retrouver une information

[Termes et flux sortants](#)

Comment utiliser cette section

Lisez-la dans l'ordre la première fois. L'introduction et la page d'architecture expliquent ce qu'est OpenLatch et ce qui tourne où ; elles suffisent pour répondre à une première revue avant toute installation. Les prérequis et les exigences réseau sont ce que vous préparez à l'avance : les valeurs à avoir sous la main, et l'unique destination sortante à autoriser. Les trois pages suivantes couvrent l'installation elle-même, de la première connexion à un déploiement vérifié, et [Exploiter](#) couvre la suite.

[Traitement des données et sécurité](#) décrit ce qu'OpenLatch traite, conserve et ne collecte jamais. C'est la page à remettre à une revue de sécurité. La [FAQ du déploiement](#) répond aux questions qui suivent, et

[Termes et flux sortants](#) réunit les termes et le tableau des flux sortants.

Chaque page de cette section s'imprime seule, sans la navigation du site, pour que vous puissiez la joindre à une demande de changement ou à une revue.

Retours et signalements de sécurité

Envoyez corrections et manques à hello@openlatch.ai, en nommant la page. Les signalements de sécurité suivent une autre voie : envoyez-les à security@openlatch.ai. OpenLatch accuse réception d'un signalement de sécurité sous deux jours ouvrés.

Introduction

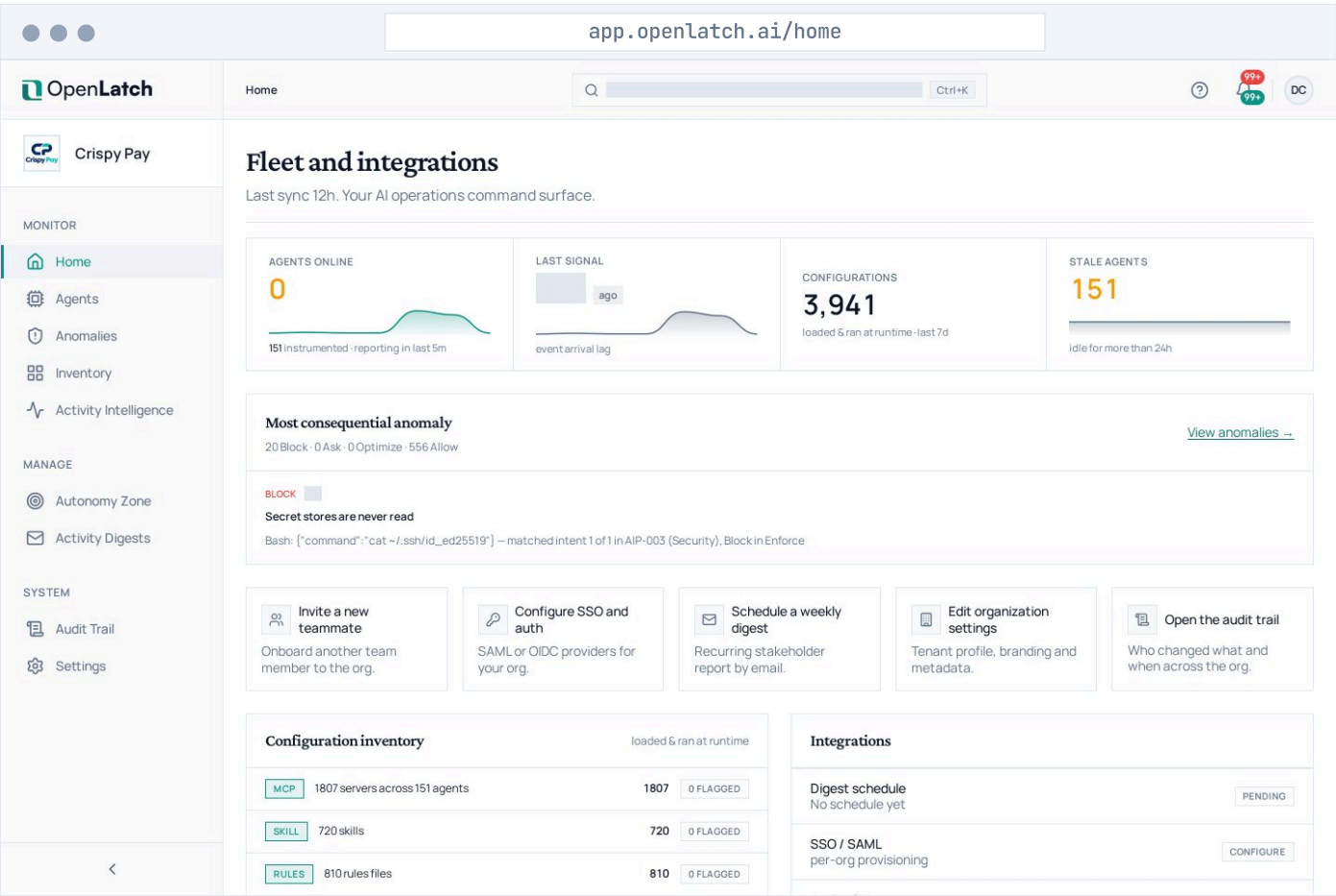
Lisez cette page pour savoir ce que vous déployez avant de préparer quoi que ce soit.

Ce qu'est OpenLatch

OpenLatch est la couche de contrôle d'exécution des agents IA d'entreprise. Il s'exécute dans le cycle de vie de l'agent lui-même, évalue chaque action couverte au regard des Agent Intent Policies de votre organisation, et rend un verdict déterministe avant que l'action ne s'exécute : Allow, Ask, Block ou Optimize.

Déterministe a ici un sens précis. Chaque action couverte atteint le point de décision des politiques, et le verdict produit par une politique explicite est celui qui est appliqué. [Architecture](#) montre le chemin qu'emprunte une action pour y parvenir.

Vous obtenez un seul endroit qui réunit chaque agent de l'organisation, les règles sous lesquelles chacun travaille, et les décisions déjà prises en son nom.



La console une fois les agents connectés et ce qu'ils peuvent faire défini par l'organisation.

Les quatre verdicts

Verdict	Ce qui se passe
Allow	L'action se poursuit
Ask	L'action se poursuit et est signalée pour une revue asynchrone. Elle ne patiente jusqu'à une réponse que là où une politique active l'option par atome pour les actions irréversibles
Block	L'action est arrêtée avant de s'exécuter
Optimize	Le chemin est modifié. Appliqué en Enforce, mesuré en Monitor

[Verdicts](#) explique chacun en détail.

Ce qui tourne où

OpenLatch a deux parties : une console dans laquelle vous travaillez, et un client que vous installez sur les machines qui exécutent des agents.

Partie	Ce que c'est	Où elle tourne
La console	Elle porte l'organisation, ses utilisateurs et leurs rôles, le registre des agents et le journal d'audit, et c'est là que les politiques sont rédigées. Elle n'applique jamais de verdict elle-même	Fournie en tant que service à <code>app.openlatch.ai</code> . L'hébergement et le traitement des données sont décrits dans Traitement des données et sécurité
Le client	Une seule unité installée. Il évalue chaque action couverte sur la machine, rend le verdict avant que l'action ne s'exécute, et envoie l'historique de ce qui s'est passé à la console. L'installation détecte les agents présents sur la machine et raccorde chacun d'eux	Chaque machine qui exécute un agent, une fois par machine. Il s'installe et s'exécute au niveau de l'utilisateur, sans droits d'administrateur

Ce que le déploiement ne demande pas à votre réseau :

Point	Ce dont OpenLatch a besoin
Ports entrants	Aucun. Le client n'écoute que sur l'interface de bouclage, et rien sur la machine n'est joignable depuis le réseau
Certificats	Aucun. OpenLatch n'émet aucun certificat et n'en exige aucun
Interception de trafic	Aucune. OpenLatch ne termine aucun trafic et n'en re-signé aucun
Changement réseau	Une seule entrée sortante à autoriser : <code>app.openlatch.ai</code> , sur TCP 443
Votre proxy existant	Le client fonctionne derrière lui, y compris sous inspection TLS lorsque l'autorité de certification d'interception est approuvée. Voir Exigences réseau

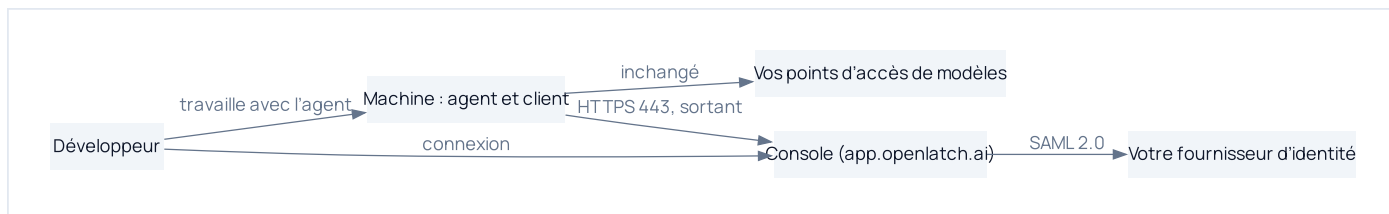


Figure 1. OpenLatch dans l'organisation : le client sur chaque machine, la console en tant que service, et le fournisseur d'identité.

Ce que vous allez mettre en place

Étape	Ce qu'elle recouvre	Où
1 · Créer l'organisation	Se connecter, nommer l'organisation, inviter les administrateurs et définir leurs rôles	Configurer l'organisation
2 · Connecter le fournisseur d'identité	Configurer l'authentification unique en SAML 2.0, et tester l'échange avant d'inviter qui que ce soit d'autre	Configurer l'organisation
3 · Connecter les agents	Déployer le client sur chaque machine, par le parcours développeur ou par vos outils de déploiement gérés	Déployer le client
4 · Vérifier et déployer	Prouver que chaque machine est connectée, puis élargir le déploiement en trois vagues	Vérifier et déployer

Ce qu'il faut pour commencer

Prérequis	Détail
Un compte dans la console	OpenLatch crée l'organisation et invite son propriétaire. Ce compte invite tous les autres
HTTPS sortant vers <code>app.openlatch.ai</code> sur TCP 443	L'unique destination qu'exige OpenLatch, à travers votre proxy. Aucun port entrant n'est ouvert. Autorisez-la par son nom, car il n'existe aucune plage d'adresses statique

[Prérequis](#) transforme cela en inventaire de préparation, avec la source de chaque valeur.

En bref

- Le verdict est produit sur la machine, avant que l'action ne s'exécute. La console n'est pas sur le chemin de décision.
- Les politiques restent en vigueur sur la machine quand la console est injoignable. L'historique est conservé localement et envoyé quand elle redevient joignable.
- Une seule destination sortante est requise. Aucun port entrant, aucun certificat à installer, aucune interception de trafic.
- La mise en place compte quatre étapes : l'organisation, le fournisseur d'identité, le client sur chaque machine, puis la vérification et le déploiement.

Architecture

Lisez cette page pour planifier en fonction de ce que le client installe sur une machine et de ce qui franchit la frontière de la machine.

Ce qui tourne sur la machine

Ici, une machine désigne le système d'exploitation sur lequel tournent les agents. Dans la plupart des parcs, c'est un poste de développeur, et c'est tout autant un bureau virtuel, un hôte de build ou une image de conteneur, partout où un agent est lancé. Si des agents y tournent, le client y est installé.

OpenLatch installe une seule chose sur une telle machine : le client OpenLatch. Il est installé une fois et couvre chaque agent de cette machine. Rien à placer devant l'agent, aucun service à monter à côté, aucun second produit à tenir synchronisé avec le premier.

Point	Détail
Ce qui est installé	Le client OpenLatch, une fois par système d'exploitation qui exécute des agents
Privilèges	S'installe et s'exécute au niveau de l'utilisateur. Aucun droit d'administrateur n'est requis, ni à l'installation ni ensuite
Posture réseau	N'écoute que sur l'interface de bouclage. Aucun port entrant n'est ouvert, et rien sur la machine n'est joignable depuis le réseau
Fichiers	Le client conserve sa configuration et ses enregistrements locaux dans un seul répertoire : <code>~/ .openlatch</code> sur macOS et Linux, <code>%APPDATA%\openlatch</code> sur Windows. <code>OPENLATCH_DIR</code> le déplace
Démarrage	Démarre avec la session de l'utilisateur, sous le gestionnaire de services du système d'exploitation
Où les décisions sont prises	Sur la machine, avant que l'action ne s'exécute. La console n'est pas sur ce chemin
Perte de connectivité	Continue de fonctionner. Les politiques restent en vigueur, et les enregistrements sont conservés jusqu'à ce que la console soit de nouveau joignable

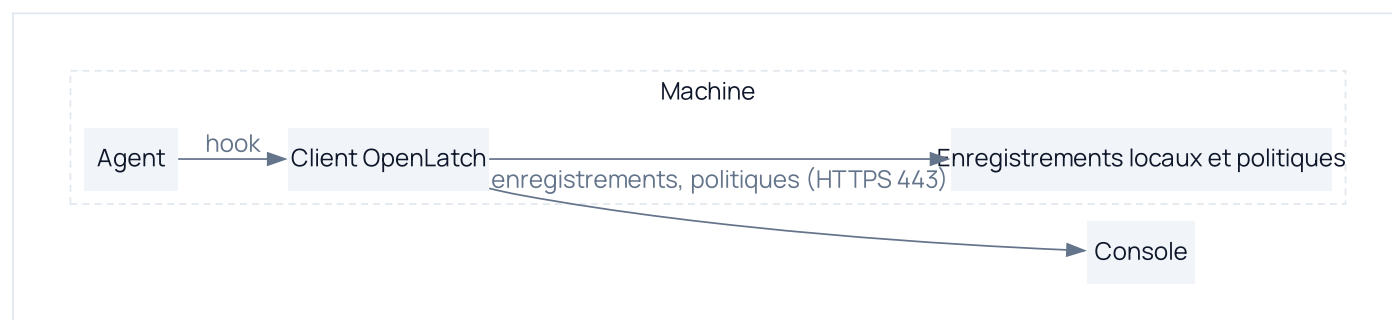


Figure 1. Vue de déploiement : le client sur chaque machine, et la console en tant que service.

La console

La console est fournie en tant que service à `app.openlatch.ai`. Rien à installer, rien à héberger. Les données de votre organisation sont stockées séparément de celles de toutes les autres organisations.

La console porte quatre éléments avec lesquels vous travaillez : l'organisation, ses utilisateurs et leurs rôles, le registre des agents et le journal d'audit. Le registre des agents liste chaque machine connectée et

les agents qui s’y trouvent, chacun avec un propriétaire et un [niveau de contrôle](#) : Enforced, Monitored ou Known. Le journal d’audit est en ajout seul et exportable, avec une durée de conservation que vous définissez. [Configurer l’organisation](#) décrit la procédure pour tout cela.

Les politiques sont rédigées dans la console, et de là elles atteignent chaque machine. [Rédiger une Agent Intent Policy](#) couvre la rédaction.

Comment une action est décidée

Une action produit un verdict, sur la machine, avant que l’action ne s’exécute. La console n’est pas sur ce chemin.

Étape	Ce qui se passe	Ce que vous pouvez observer
1 · L’agent propose l’action	L’agent s’apprête à lancer une commande shell ou à appeler un outil. Il remet d’abord l’action proposée à OpenLatch, et patiente	—
2 · OpenLatch décide sur la machine	La décision est prise au regard des politiques que la machine détient déjà. Aucun appel réseau et aucun modèle sur ce chemin	Un verdict est rendu que la console soit joignable ou non
3 · Le verdict revient à l’agent	Le verdict revient sous la forme que l’agent comprend, avec sa raison. Quand plusieurs politiques correspondent, la plus restrictive l’emporte	Sur un Block, le développeur voit le refus et sa raison dans l’agent, et l’action ne s’exécute pas
4 · Le résultat est enregistré	L’action et son verdict sont consignés sur la machine puis envoyés à la console. L’envoi est hors du chemin de décision : une console lente change le moment où une action est vue, jamais ce qui a été décidé	La machine, ses agents et leurs actions décidées apparaissent dans le registre des agents de la console

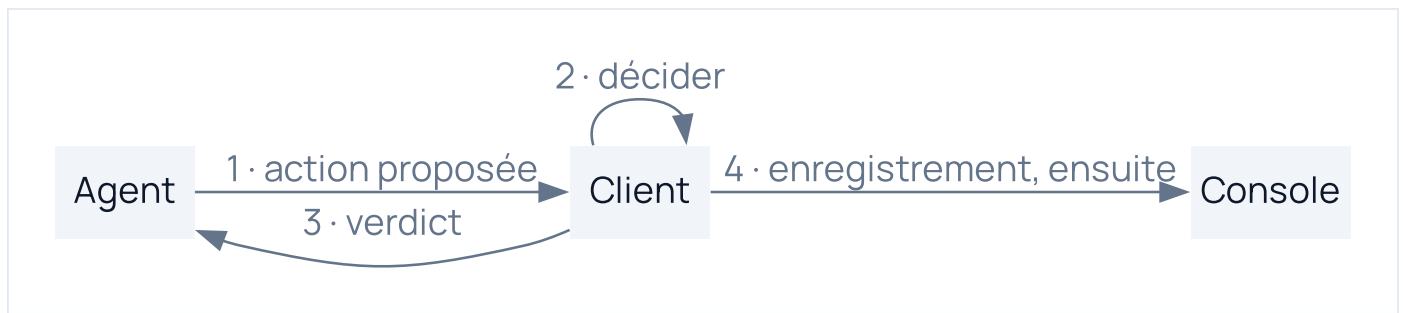


Figure 2. Une action, un verdict : la décision est prise sur la machine avant que l'action ne s'exécute, et l'enregistrement suit.

Ce qui franchit la frontière de la machine

Quatre flux franchissent la frontière d'une machine qui exécute le client. Deux seulement concernent OpenLatch.

Flux	Sens et contenu
Politiques	Descendant, de la console vers la machine. Le client vérifie au démarrage puis à intervalles réguliers s'il existe des politiques plus récentes que celles qu'il détient
Enregistrements	Montant, de la machine vers la console, en HTTPS et à travers votre proxy, une fois l'action déjà décidée
Le trafic de modèles de vos agents	Inchangé. Vos agents atteignent leurs points d'accès de modèles exactement comme avant, avec leurs propres identifiants, et OpenLatch ne modifie pas ce trafic
Connexion à la console	Vers la console depuis un navigateur, à travers votre fournisseur d'identité

[Exigences réseau](#) liste chaque destination et couvre les proxys et l'inspection TLS.

Quand le réseau est indisponible

Une console injoignable est un problème d'acheminement, pas un problème d'application. Les politiques déjà présentes sur la machine restent en vigueur, et la machine continue de décider avec elles aussi longtemps que dure la coupure. Les enregistrements qui ne peuvent pas être envoyés sont conservés localement, jusqu'à une limite de taille, et remontent quand la console redevient joignable. Une coupure

assez longue pour dépasser la limite abandonne les enregistrements non envoyés les plus anciens, et le client le consigne.

Les politiques n'expirent pas d'elles-mêmes. Une politique qui se désactiverait après une coupure permettrait à n'importe qui de désactiver l'application en débranchant un câble : la machine continue donc d'appliquer, et la console montre que les politiques de la machine sont périmées. Une machine qui n'a jamais reçu de politique autorise l'action et consigne qu'elle n'avait pas de politique : OpenLatch ne bloque jamais sans politique. [Exigences réseau](#) donne le tableau complet des états dégradés.

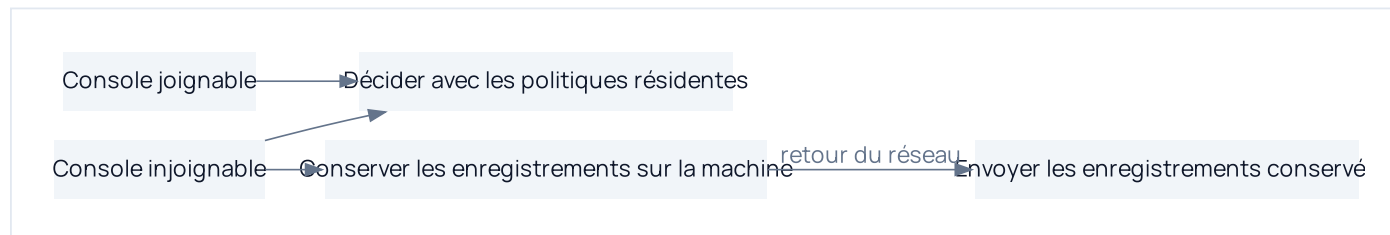


Figure 3. Décider hors ligne : la machine continue de décider avec les politiques qu'elle détient, et les enregistrements attendent le réseau.

En bref

- Un client par machine qui exécute un agent, au niveau de l'utilisateur, sans droits d'administrateur. Il n'écoute que sur le bouclage et n'ouvre aucun port entrant.
- Il conserve sa configuration et ses enregistrements locaux dans un seul répertoire, `~/ .openlatch` ou `%APPDATA%\openlatch`, que `OPENLATCH_DIR` déplace.
- Les actions sont décidées sur la machine, avant de s'exécuter, au regard des politiques que la machine détient déjà. Aucun appel réseau sur ce chemin.
- La console est un service, rien à installer ni à héberger. Elle porte l'organisation, ses utilisateurs et leurs rôles, le registre des agents et le journal d'audit, et les politiques atteignent chaque machine depuis elle.
- Une console injoignable n'arrête pas l'application. Les politiques restent en vigueur, les enregistrements sont conservés et envoyés plus tard, et une machine sans politique autorise l'action et consigne pourquoi.

Prérequis

Remplissez cette page avant de planifier la première vague de déploiement. Rien dans le déploiement n'est difficile à obtenir. Ce qui bloque un déploiement, c'est de découvrir, le matin de la première vague, qu'une décision se trouve ailleurs dans l'organisation, ou que la personne qui peut approuver une entrée de flux sortant est absente. L'inventaire ci-dessous transforme cette découverte en un formulaire que quelqu'un remplit la semaine précédente.

Remplissez l'inventaire de préparation en une seule séance de travail, avec dans la salle toutes les personnes qui détiennent une réponse. La liste de contrôle transforme ensuite ces réponses en lignes regroupées par phase de travail. La dernière partie indique ce que coûte le déploiement en heures.

Inventaire de préparation

Chaque réponse change quelque chose en aval. Les réponses sur les agents décident quelles machines appartiennent à quelle vague. Les réponses sur les postes décident du canal d'installation. Les réponses réseau décident si la route est découverte ou fournie. Les réponses d'identité décident qui administre l'organisation. Quand la réponse est réellement « aucun », écrivez « aucun » plutôt que rien, pour qu'un lecteur ultérieur distingue une question répondue d'une question jamais posée.

Agents

Élément	Pourquoi c'est important
Quels agents sont utilisés, et sur quelles machines	L'installation détecte les agents présents sur une machine et raccorde chacun d'eux. Connaître l'ensemble à l'avance indique quelles machines appartiennent à quelle vague. Dès qu'un agent remonte des données, la console affiche son niveau de contrôle
Si un agent tourne ailleurs que sur un poste personnel	Le client est installé une fois par machine qui exécute un agent, où que se trouve cette machine. Un hôte de build ou un poste partagé qui exécute un agent est concerné au même titre qu'un portable

Postes

Élément	Pourquoi c'est important
Répartition des systèmes d'exploitation et nombre de machines par système	Le client s'installe et s'exécute au niveau de l'utilisateur sur chaque plateforme prise en charge, sans droits d'administrateur. Les versions minimales sont macOS 13, Windows 10 1809 ou Windows Server 2019, Ubuntu 20.04 et Debian 10
Architectures de processeur du parc	Des paquets existent pour macOS sur Apple silicon et Intel, Linux x64 et arm64, et Windows x64. Windows sur ARM64 et les distributions Linux basées sur musl, comme Alpine, ne sont pas pris en charge
Le canal de distribution ou de configuration que votre organisation exploite déjà	Les vagues 2 et 3 livrent l'installation par un canal qui atteint déjà ces machines. Ce canal transporte aussi l'identifiant de la vague depuis votre coffre de secrets
Si des machines d'une vague sont partagées par plusieurs personnes	L'installation et l'état local sont propres à chaque utilisateur. Chaque utilisateur d'une machine partagée fait sa propre installation

Réseau

Élément	Pourquoi c'est important
Proxy sur le chemin : oui ou non	Décide si la route est découverte à l'installation ou lui est fournie. Voir Exigences réseau
Type de proxy : HTTP CONNECT, SOCKS5 ou configuration automatique de proxy	Les trois sont pris en charge. Un fichier de configuration automatique de proxy est évalué par le système d'exploitation sur Windows et macOS. Sur Linux, définissez la route explicitement
Authentification du proxy : aucune, Basic, Kerberos (Negotiate) ou NTLM	Aucune, Basic et Negotiate sont prises en charge. NTLM ne l'est pas, par choix de conception : un parc uniquement NTLM prévoit un chemin Negotiate ou non authentifié pour le client avant la première vague
Inspection TLS : oui ou non, et où se trouve l'autorité de certification d'interception	L'inspection ne demande aucune exception. La racine d'interception va dans le magasin de confiance du système d'exploitation, ou est fournie sous forme de fichier PEM à l'installation
Si les points d'accès de modèles que vos agents atteignent déjà restent joignables, et sont exemptés de la mise en mémoire tampon des réponses et de l'inspection des flux	Vos agents atteignent ces points d'accès exactement comme avant. Un équipement qui met en tampon une réponse en flux fait arriver la sortie de l'agent d'un seul bloc

Identité

Élément	Pourquoi c'est important
Le fournisseur d'identité SAML 2.0 de votre organisation	La console authentifie une organisation en SAML 2.0. Microsoft Entra ID, Okta, PingOne et Active Directory Federation Services se connectent ainsi. La connexion demande un administrateur de chaque côté
Domaines de messagerie autorisés	Les domaines autorisés et les invitations contrôlent ensemble qui peut rejoindre l'organisation
Annuaire pour la recherche des propriétaires : Microsoft Entra ID, Okta, Google Workspace, PingOne ou Active Directory	Une recherche en lecture seule rattache le propriétaire d'un agent à une personne. Elle ne crée ni ne supprime jamais d'utilisateurs. Cette connexion est facultative
Qui détient les rôles owner et admin dans la console	Les rôles de la console sont owner, admin, editor, viewer. Les personnes invitées commencent en viewer, et les réglages d'identité demandent admin

Secrets

Élément	Pourquoi c'est important
Où est conservée la <code>OPENLATCH_API_KEY</code> de chaque vague, et comment elle atteint une machine sans apparaître dans un script	Une clé par vague de déploiement, créée dans la console et révoquée quand la vague se termine. Les installations gérées la lisent depuis votre coffre de secrets comme variable d'environnement. Elle n'est jamais écrite dans un script, un paquet ou une ligne de commande

Pilote

Élément	Pourquoi c'est important
Les dix machines de la vague 1 et leurs propriétaires	La vague 1 suit le parcours développeur. Des propriétaires nommés permettent de rapprocher chaque machine du registre des agents de la console
Comment vous joindrez OpenLatch pendant le déploiement	Convenez de la voie de support avant la première vague plutôt que pendant. Voir Exploiter

 Note

Un parc dont le proxy est uniquement NTLM est la seule réponse de cet inventaire qui peut bloquer un déploiement d'emblée. Interrogez tôt l'équipe qui gère votre proxy, et lisez la matrice de prise en charge des proxys dans [Exigences réseau](#) avant de planifier la première vague.

Liste de contrôle consolidée

Chaque ligne indique d'où vient la valeur et la page qu'elle bloque. Les quatre phases sont indépendantes : les lignes console et les lignes réseau avancent en parallèle la même semaine.

Préparer la console

#	Élément	D'où il vient	Bloque
1	Organisation créée dans la console	Votre contact OpenLatch crée l'organisation et invite son propriétaire. Acceptez l'invitation depuis l'adresse invitée. Les connexions suivantes commencent sur <code>app.openlatch.ai</code> par une étape e-mail dont la vérification dans le navigateur demande <code>challenges.cloudflare.com</code>	Configurer l'organisation
2	SAML connecté et testé	La console d'administration de votre fournisseur d'identité, pour l'identifiant d'entité, l'URL d'authentification, et le certificat ou l'URL de métadonnées. Saisis dans la console sous Settings > SAML SSO	Configurer l'organisation
3	Domaines de messagerie autorisés définis	Les mêmes réglages SAML que la ligne 2	Configurer l'organisation
4	Rôles owner, admin, editor et viewer attribués	La console, sous Settings > Users . Les personnes invitées commencent en viewer	Configurer l'organisation
5	Annuaire choisi pour la recherche des propriétaires, facultatif	La console d'administration de votre annuaire. La recherche est en lecture seule	Configurer l'organisation
6	Conservation du journal d'audit définie	La console, sous Settings > Audit Trail : 90, 180, 365 ou 730 jours, ou une durée personnalisée, avec purge automatique et export	Configurer l'organisation
7	Canal de notification webhook ou Slack, facultatif	La console, sous Settings > Notification Channels	Configurer l'organisation
8	Identifiant de vague créé et	La console, sous API Keys dans votre compte, depuis une session de navigateur connectée. Stocké dans le coffre	Configurer l'organisation

#	Élément	D'où il vient	Bloque
	stocké	de secrets de votre organisation	

Préparer le réseau

#	Élément	D'où il vient	Bloque
9	Entrée de flux sortant pour <code>app.openlatch.ai</code> sur TCP 443 approuvée	Votre processus de changement, d'après le tableau de Exigences réseau	Déployer le client
10	Les points d'accès de modèles que vos agents atteignent déjà restent joignables, et sont exemptés de la mise en tampon des réponses et de l'inspection des flux	La configuration de votre proxy ou de votre passerelle web sécurisée	Déployer le client
11	Type de proxy et chemin d'authentification confirmés	La configuration de votre proxy, lue au regard de la matrice de prise en charge	Déployer le client
12	Autorité de certification d'interception disponible en fichier PEM, si le système d'exploitation ne l'approuve pas déjà	L'export de votre propre autorité de certification	Déployer le client

Préparer les machines

#	Élément	D'où il vient	Bloque
13	Les agents à couvrir sont installés et connectés sur ces machines	Les machines elles-mêmes, confirmées avec la personne qui utilise chacune	Déployer le client
14	Installation revue pour la voie gérée	La voie gérée dans Déployer le client . Elle lit <code>OPENLATCH_API_KEY</code> depuis votre coffre de secrets et ne la porte jamais en clair	Déployer le client

Préparer le déploiement

#	Élément	D'où il vient	Bloque
15	Liste des machines de la vague 1 avec leurs propriétaires	Votre inventaire de postes, recoupé avec les propriétaires des agents	Vérifier et déployer
16	Développeurs informés que certains agents leur demandent de confirmer la connexion une fois	Votre propre communication de déploiement, envoyée avant le début de la vague	Vérifier et déployer
17	Voie de support convenue pour le déploiement	Exploiter	Vérifier et déployer
18	Responsable du retour arrière désigné	Votre propre décision : la personne qui peut suspendre une vague et retirer le client d'une machine	Vérifier et déployer

Effort et calendrier

Le déploiement du client se fait en trois vagues de deux semaines. La vague 1 installe environ dix machines par le parcours développeur. La vague 2 passe environ cinquante machines par le canal que vous exploitez déjà. La vague 3 s'étend au périmètre complet et se termine par la décision de passer en exploitation courante.

L'effort se concentre sur la première semaine, quand l'organisation est mise en place et la route prouvée sur une machine de chaque système d'exploitation du parc. Ensuite, chaque vague répète une procédure qui a déjà fonctionné.

Phase de travail	Vague 1	Vague 2	Vague 3	Effort habituel
Préparer la console	Connecter SAML, définir les domaines autorisés et la conservation de l'audit, inviter les personnes, créer l'identifiant de vague	Renouveler l'identifiant de vague	Transmettre la routine d'exploitation	Une journée la première semaine, puis quelques heures par semaine
Préparer le réseau	Approuver l'entrée de flux sortant et confirmer la route du proxy	Revoir la voie gérée	Aucun autre changement	Une demi-journée la première semaine
Préparer les machines	Prouver l'installation sur une machine par système d'exploitation, puis passer dix machines par le parcours développeur	Porter l'installation dans le canal que vous exploitez déjà et lancer la vague	Étendre le même canal aux machines restantes	Quelques minutes par machine après la première
Confirmer et exploiter	Confirmer que chaque machine installée apparaît dans le registre des agents avec un propriétaire	Point hebdomadaire et revue de la voie gérée	Point hebdomadaire, puis décision de passer en exploitation courante	Un point hebdomadaire pendant tout le déploiement

En bref

- Remplissez l'inventaire de préparation en une seule séance avant de planifier la première vague. Chaque case vide est une décision que quelqu'un prendra plus tard sous pression.
- Une seule entrée de flux sortant est obligatoire : `app.openlatch.ai` sur TCP 443, en sortie. Tout le reste sur le chemin est le trafic propre de vos agents, qui ne change pas.

- Un parc dont le proxy est uniquement NTLM bloque le déploiement. Prévoyez un chemin Negotiate ou non authentifié pour le client.
- Aucun droit d'administrateur n'est nécessaire sur une machine. L'installation et l'état local sont propres à chaque utilisateur : une machine partagée par plusieurs personnes demande une installation par utilisateur.
- Une OPENLATCH_API_KEY par vague de déploiement, conservée dans votre coffre de secrets et révoquée dans la console quand la vague se termine. Elle n'apparaît jamais dans un script, un paquet ou une ligne de commande.

Exigences réseau

Utilisez cette page comme demande de changement pour votre équipe réseau. Le client a besoin d'une seule destination sortante, atteinte à travers votre proxy, et n'ouvre rien en entrée.

Une seule dépendance sortante

Le client tourne sur la machine qui exécute l'agent, dans le cycle de vie de l'agent lui-même. Il contacte OpenLatch pour trois choses : récupérer les politiques avec lesquelles il décide, transmettre l'enregistrement de ce qu'il a décidé, et s'authentifier. Les trois vont vers une seule destination sur un seul port. La télémétrie produit et les rapports de plantage, facultatifs, vont vers cette même destination : OpenLatch n'ajoute aucune autre destination, que vous les laissiez actifs ou que vous les coupiez.

Propriété	Valeur
Sens	Sortant uniquement. Le client n'ouvre aucun port d'écoute sur une interface réseau, et rien sur la machine ne devient joignable depuis le réseau
Destination	<code>app.openlatch.ai</code> , sur TCP 443
Transport	HTTPS. TLS est vérifié avec le magasin de confiance du système d'exploitation, plus un paquet de certificats facultatif fusionné par-dessus. La vérification ne peut pas être désactivée
À travers votre proxy	Oui : HTTP et HTTPS CONNECT, SOCKS5, et configuration automatique de proxy là où le système d'exploitation la prend en charge, avec authentification Basic et Kerberos (Negotiate)
Liste d'adresses IP autorisées	Non prise en charge. Il n'existe aucune plage d'adresses statique : autorisez par nom de domaine complet et par SNI
Certificats	Aucun émis par OpenLatch, et aucun exigé de votre part

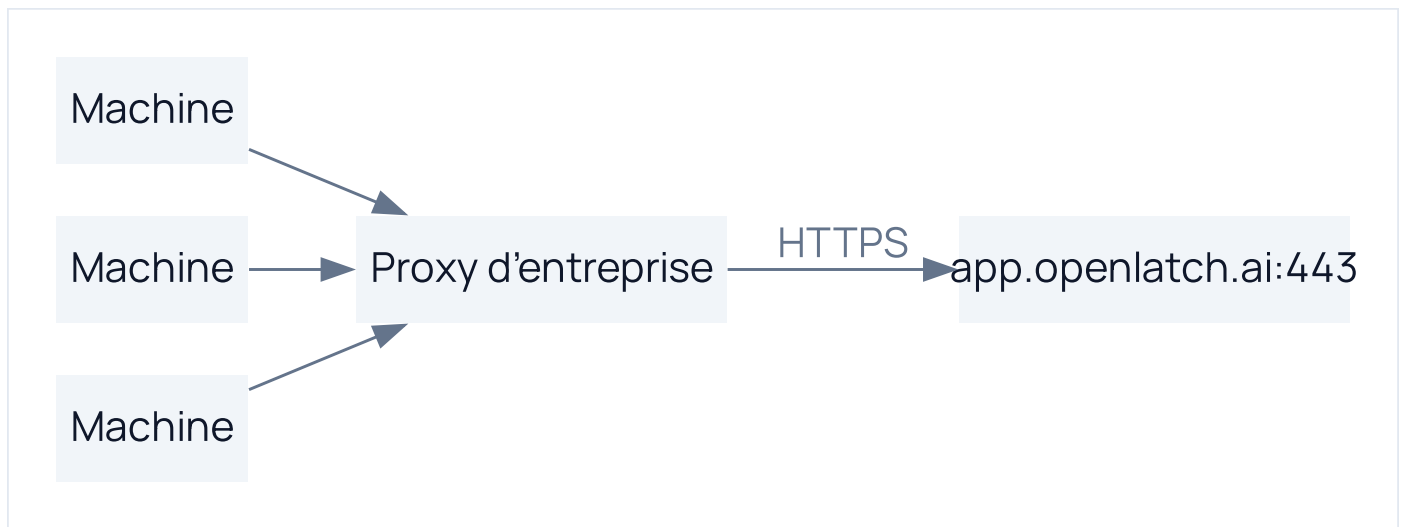


Figure 1. Un parc de machines derrière un proxy d'entreprise : une seule destination sortante, rien en entrée.

Flux sortants requis

C'est le tableau à coller dans une demande de changement. La première ligne est ce dont OpenLatch a besoin pour fonctionner, et c'est la seule destination qu'une machine qui exécute le client demande à OpenLatch. La deuxième n'est nécessaire que pendant l'installation ou la mise à jour du client ; un parc qui dispose d'un miroir interne de paquets la pointe sur ce miroir. La troisième n'est nécessaire que pour installer avec l'un des deux scripts. La quatrième ne concerne que les navigateurs.

Destination	Port	Sens	Protocole	Usage	Autorisation
<code>app.openlatch.ai</code>	TCP 443	Sortant	HTTPS	Politiques, enregistrements de décisions, authentification, et la télémétrie produit et les rapports de plantage facultatifs	Par nom et SNI. Il n'existe aucune plage d'adresses statique
Votre registre de paquets : <code>registry.npmjs.org</code> , sauf si vous pointez le client sur un miroir	TCP 443	Sortant	HTTPS	Installer le client, et télécharger une mise à jour quand vous l'autorisez	Par nom, ou par le nom de votre miroir interne. Nécessaire uniquement pendant une installation ou une mise à jour
<code>openlatch.ai</code>	TCP 443	Sortant	HTTPS	Télécharger le script d'installation, <code>install.sh</code> ou <code>install.ps1</code> , et le rapport d'installation qu'il envoie	Par nom. Nécessaire uniquement pour une installation par script
<code>challenges.cloudflare.com</code> , depuis le navigateur de toute personne qui se connecte à la console	TCP 443	Sortant	HTTPS	La vérification que l'étape e-mail effectuée à la connexion	Par nom, pour les navigateurs uniquement. Aucune

Destination	Port	Sens	Protocole	Usage	Autorisation
					machine qui exécute le client ne le contacte

Note

Sur un parc qui refuse les flux sortants par défaut, ce sont la deuxième et la troisième lignes qu'on oublie. L'installation échoue alors avant que le client n'ait jamais atteint la console, et l'échec ressemble à un produit défaillant plutôt qu'à une route fermée.

Le navigateur de l'administrateur

La connexion à la console commence par une étape e-mail, et cette étape effectuée dans le navigateur une courte vérification que le visiteur est une personne. La vérification est servie depuis `challenges.cloudflare.com` : les navigateurs avec lesquels vos administrateurs se connectent doivent atteindre ce nom sur TCP 443. Un navigateur qui ne peut pas l'atteindre échoue à chaque fois. Cette exigence ne concerne que les navigateurs : `app.openlatch.ai` reste la seule destination dont une machine qui exécute le client a besoin.

Le trafic propre de vos agents

Vos agents atteignent leurs points d'accès de modèles exactement comme avant l'installation d'OpenLatch. OpenLatch ne termine ni ne re-signe ce trafic, et n'y ajoute aucune destination. Ces points d'accès doivent rester joignables et être exemptés de la mise en tampon des réponses et de l'inspection des flux, pour qu'une réponse en flux ne soit pas retenue en chemin vers la personne qui l'attend.

Un équipement qui met la réponse en tampon, ou l'analyse comme un corps entier, retient le flux jusqu'à la fin puis le libère d'un bloc. La requête aboutit quand même ; le développeur voit un agent qui semble se figer puis affiche tout d'un coup.

Pour les produits de proxy qui orientent le trafic par application plutôt que par destination, le nom du processus est `openlatch` sur macOS et Linux et `openlatch.exe` sur Windows. Utilisez ce nom dans une définition d'application personnalisée.

Matrice de prise en charge des proxys

Répondez contrôle par contrôle. Une ligne « Non » est un choix de conception plutôt qu'un manque, et chacune indique la voie à suivre à la place.

Contrôle réseau	Pris en charge	Traitement	Remarques
Proxy HTTP ou HTTPS CONNECT non authentifié	Oui	Découvert à l'installation, ou fourni à l'installation	La forme courante en entreprise. La route qui fonctionne est enregistrée sur la machine
Authentification Basic	Oui	L'identifiant est demandé lors d'un défi et conservé dans le magasin d'identifiants du système d'exploitation	Jamais écrit en clair dans un fichier
Kerberos (Negotiate)	Oui	Utilise l'identité de la session ouverte, par le mécanisme du système d'exploitation	Le principal de service vaut par défaut le service HTTP de l'hôte du proxy et peut être remplacé à l'installation. Un ticket manquant ou expiré est signalé avec son remède
NTLM	Non	Non pris en charge, par choix de conception	Microsoft a déprécié NTLM, et un client qui y redescend discrètement va à l'encontre de ce pour quoi il a été déployé. Utilisez Negotiate, ou un chemin non authentifié pour le client
Configuration automatique de proxy (PAC ou WPAD)	Oui, sur Windows et macOS	L'URL est confiée au résolveur du système d'exploitation	Refusée sur Linux, où aucun résolveur de configuration automatique n'est disponible pour le client. Définissez-y la route explicitement
SOCKS5	Oui	Une route socks5 : // ou socks5h : //	La résolution de noms reste locale avec socks5 et passe au proxy avec socks5h
Proxy transparent	Oui	Aucune configuration sur la machine ; le trafic	Combinez-le avec la ligne d'inspection TLS si l'équipement

Contrôle réseau	Pris en charge	Traitement	Remarques
		est intercepté sur le chemin	re-signe la connexion
Inspection TLS avec une autorité de certification d'entreprise	Oui	La racine d'interception va dans le magasin de confiance du système d'exploitation, ou est fournie en paquet PEM à l'installation	Le paquet est fusionné par-dessus les racines du système, jamais en remplacement, et la vérification n'est jamais désactivée
Certificats clients mTLS vers le proxy	Non	—	Le client ne présente aucun certificat client. Utilisez l'un des modes d'authentification pris en charge
Liste d'adresses IP autorisées pour les flux sortants d'OpenLatch	Non	—	Il n'existe aucune plage d'adresses statique. Autorisez <code>app.openlatch.ai</code> par nom et SNI

Configurer la route

Comment la route est trouvée

Sur la plupart des machines d'entreprise, l'installation trouve la route sans qu'on la lui indique. Elle sonde ce qui est déjà configuré, puis parcourt les réglages du système d'exploitation, y compris la configuration automatique de proxy sur Windows et macOS. Elle ne pose une question que si rien de cela n'atteint la console, une fois, dans un terminal. Une installation sans surveillance qui ne trouve pas de route s'arrête avec un rapport qui nomme chaque candidat essayé, plutôt que de passer en direct sans le dire.

La route qui a permis l'installation est enregistrée sur la machine. C'est important : le client démarre avec la session de l'utilisateur et n'hérite pas des variables du shell d'installation, si bien qu'une route qui n'existait que dans une variable d'environnement serait perdue au redémarrage suivant.

Fournir la route à l'installation

Quand la découverte ne peut pas fonctionner, ou qu'une installation gérée tourne sans terminal pour poser la question, fournissez la route directement à l'installation. Fournissez en même temps le paquet de certificats si le système d'exploitation n'approuve pas déjà la racine d'interception.

Retenez l'inscription avec `OPENLATCH_NO_INIT` et lancez-la dans une seconde étape, pour que les options atteignent la commande qui l'effectue. Exportez la variable avant le lancement du script, pour que le shell qui lit le script la porte aussi.

Sur macOS et Linux :



```
export OPENLATCH_NO_INIT=1
curl -LsSf https://openlatch.ai/install.sh | sh
~/local/bin/openlatch init --proxy http://proxy.corp:8080 --ca-bundle /etc/ssl/corp-root.pem
```

Sur Windows :



```
$env:OPENLATCH_NO_INIT = "1"
powershell -ExecutionPolicy Bypass -c "irm https://openlatch.ai/install.ps1 | iex"
& "$env:LOCALAPPDATA\openlatch\bin\openlatch.exe" init --proxy http://proxy.corp:8080 --ca-bundle C:\corp\corp-root.pem
```

Sur une machine qui dispose déjà de Node.js, `npx @openlatch/client init` accepte les mêmes options en une seule étape :



```
npx @openlatch/client init --proxy http://proxy.corp:8080 --ca-bundle /etc/ssl/corp-root.pem
```

Une liste d'exceptions peut être fournie avec la route, dans la syntaxe habituelle. Le trafic qui ne quitte pas la machine ne passe pas par le proxy et ne demande aucune exception.

Quand la politique interdit à une machine de sortir sans le proxy, le client peut être configuré pour refuser une connexion directe si aucune route n'est résolue. Il signale alors un échec de route au lieu de

contourner discrètement vos contrôles. [Configuration du client](#) liste les réglages.

Gestion des identifiants

Un identifiant de proxy est conservé dans le magasin d'identifiants du système d'exploitation : le Gestionnaire d'identifiants sur Windows, le Trousseau sur macOS, Secret Service sur Linux, et un fichier chiffré sur un hôte sans interface qui n'en dispose pas. Il n'est jamais écrit en clair dans un fichier de configuration. Partout où une URL de proxy est affichée ou journalisée, elle apparaît sous la forme `user:*****@host`. Une URL de proxy qui porte un mot de passe en ligne de commande est refusée, car une ligne de commande est lisible par n'importe quel processus de la machine.

Inspection TLS

La connexion à la console fonctionne sous inspection et ne demande aucune exception. Le client fait des appels HTTPS ordinaires de type requête-réponse, vérifie avec le magasin de confiance du système d'exploitation, et n'épingle aucun certificat. Placez la racine d'interception dans le magasin de confiance là où votre parc le fait déjà, ou fournissez-en une copie PEM à l'installation. Le paquet est fusionné par-dessus les racines du système : une machine continue de faire confiance à tout ce qu'elle approuvait déjà.

Les points d'accès de modèles de vos agents font exception, et la demande y est étroite : exemptez-les de la mise en tampon des réponses et de l'inspection des flux, et laissez le reste de votre politique d'inspection intact.

Attention

Ne désactivez jamais la vérification des certificats pour contourner un proxy qui inspecte. Le client n'offre aucun réglage pour cela. Un échec de négociation est signalé avec son remède : approuver la racine d'interception sur la machine, ou la fournir à l'installation.

Comportement hors ligne et dégradé

Le client actualise ses politiques depuis la console en arrière-plan. Chaque décision est prise sur la machine, avec les politiques déjà présentes : aucun appel réseau n'est sur le chemin de décision. C'est ce qui permet à un verdict de survivre à une route coupée.

Là où la machine détient des politiques, elle continue de les appliquer quoi que fasse le réseau. Là où elle n'en détient aucune, elle autorise l'action et marque l'enregistrement pour indiquer que la décision a été

prise sans politique. Rien n'échoue en silence.

Situation	Ce que fait la machine	Ce que vous voyez
La console est injoignable	Continue d'appliquer les politiques qu'elle détient et réessaie en arrière-plan. Les enregistrements sont conservés sur la machine	La machine cesse de se signaler. openlatch status la montre déconnectée, avec des politiques présentes
Les politiques de la machine n'ont pas été actualisées depuis un moment	Continue de les appliquer. Les politiques n'expirent pas	Un avertissement de péremption. La péremption est signalée, jamais suivie d'effet
L'identifiant de la vague est révoqué ou renouvelé, et la console le refuse	Cesse de récupérer les mises à jour, continue d'appliquer ce qu'elle détient, et reprend seule dès qu'un identifiant valide est en place	Un avertissement d'authentification sur la machine, et plus aucun signalement dans la console
Votre organisation n'a encore publié aucune politique	Maintient en vigueur ce qu'elle détient déjà	La machine se signale comme connectée, sans rien à récupérer
Une mise à jour arrive endommagée ou sous une forme que la machine ne reconnaît pas	La rejette et garde les politiques précédentes. Une mise à jour endommagée ne prend jamais effet	Un avertissement d'intégrité sur la machine, avec les politiques précédentes toujours en vigueur
La copie locale des politiques est modifiée sur la machine	La supprime. La machine fonctionne sans politique jusqu'à la prochaine actualisation réussie	Chaque verdict est marqué comme décidé sans politique, et la machine signale la perte
Aucune politique n'a jamais atteint la machine	Rend Allow : une machine ne bloque jamais sans politique	Chaque verdict est marqué comme décidé sans politique

Situation	Ce que fait la machine	Ce que vous voyez
Le client ne tourne pas	L'action se poursuit et n'est pas retenue	<code>openlatch status</code> signale que le client ne tourne pas, et la console ne montre aucune activité récente pour la machine
Les enregistrements ne peuvent pas être transmis	Réessaie, puis les conserve sur la machine et les envoie quand la console redevient joignable	Un trou dans la console qui se referme quand la machine se reconnecte. La réserve est bornée : au-delà de sa limite de taille, les enregistrements non envoyés les plus anciens sont abandonnés, et le journal du daemon consigne un avertissement <code>OL - 1204</code>
Vous passez toutes les politiques en Monitor depuis la console	Chaque machine applique le changement à sa prochaine actualisation, et consigne ce qu'elle aurait fait	Le changement, dans la console, sans accès nécessaire à une machine. Les décisions continuent d'être enregistrées

Vérifier la route

Lancez ces commandes sur la première machine avant la vague, puis sur toute machine qui signale un problème de connexion.

...

```
openlatch status
openlatch doctor
openlatch system proxy test
```

Commande	Ce qu'elle prouve
<code>openlatch status</code>	Si la machine est connectée à la console, si des politiques sont présentes, et quels agents sont raccordés. Elle répond même quand la route est coupée, c'est-à-dire exactement quand vous en avez besoin
<code>openlatch doctor</code>	La machine entière, avec une cause et un remède pour chaque point relevé. Elle signale le saut par votre proxy séparément de la connexion à la console : un proxy hors service et une console injoignable ne ressemblent jamais à la même panne
<code>openlatch system proxy test</code>	La route elle-même, sondée saut par saut. Elle s'arrête au saut qui échoue et le nomme

[Déployer le client](#) couvre son installation, et [Vérifier et déployer](#) la vérification complète d'une machine.

En bref

- Approuvez une seule entrée pour les machines : `app.openlatch.ai` sur TCP 443, en sortie. Aucun port entrant.
- Les navigateurs qui se connectent à la console ont aussi besoin de `challenges.cloudflare.com` sur TCP 443. Aucune machine qui exécute le client ne le contacte.
- Autorisez par nom et SNI, pas par adresse. Il n'existe aucune plage d'adresses statique.
- Vos agents atteignent leurs points d'accès de modèles comme avant. Gardez ces points d'accès joignables et exemptés de la mise en tampon des réponses et de l'inspection des flux.
- L'inspection TLS ne demande aucune exception pour la connexion à la console. Les proxys Basic, Kerberos, SOCKS5, transparents et à configuration automatique sont pris en charge. NTLM ne l'est pas : prévoyez un chemin Negotiate ou non authentifié.
- Une route coupée dégrade l'acheminement, pas l'application. La machine continue de décider avec les politiques qu'elle détient, les enregistrements attendent que la console soit joignable, et les deux états apparaissent dans `openlatch status`.

Configurer l'organisation

Préparez la console pour votre organisation : le compte, l'authentification unique, les utilisateurs et leurs rôles, la recherche des propriétaires, la conservation et les notifications, et l'identifiant avec lequel le client s'inscrit. Tout se passe dans le navigateur, prend environ une demi-heure et ne touche aucune machine. L'installation du client est décrite dans [Déployer le client](#).

Avant de commencer, ayez rempli l'[inventaire de préparation](#), et vérifiez que votre navigateur atteint `challenges.cloudflare.com`.

Se connecter et finaliser l'organisation

OpenLatch crée l'organisation pour vous, désigne comme propriétaire l'adresse que vous avez donnée à votre contact OpenLatch, et envoie une invitation à cette adresse. L'assistant de bienvenue est l'endroit où vous finalisez la configuration.

- 1 Acceptez l'invitation.** Ouvrez le lien de l'e-mail d'invitation. Il vous mène à la page de connexion propre à votre organisation et admet l'adresse invitée comme propriétaire. Terminez avec n'importe quelle méthode proposée par la page : une passkey, un lien magique, une adresse e-mail et un mot de passe, ou une connexion via un réseau social.

Résultat attendu : la console s'ouvre sur votre organisation, et votre rôle indique owner.

- 2 Connectez-vous depuis `app.openlatch.ai`.** Chaque connexion suivante commence par une étape : votre adresse e-mail. Cette étape effectuée dans votre navigateur une courte vérification que le visiteur est une personne ; elle ne demande généralement rien, et au plus un clic. La console trouve ensuite l'organisation à laquelle appartient votre adresse et ouvre sa page de connexion avec l'adresse déjà remplie.

Une adresse qu'aucune organisation ne connaît en est informée. Une vérification qui échoue à chaque fois signifie le plus souvent que le navigateur ne peut pas atteindre `challenges.cloudflare.com` ; voir [Exigences réseau](#).

- 3 Suivez l'assistant de bienvenue.** Il compte quatre étapes : nommer l'organisation, inviter des collègues, connecter un agent, et un récapitulatif de ce que vous avez fait et passé. Utilisez la

première pour définir la raison sociale ou le nom d'entité que vous voulez voir dans le journal d'audit. La connexion d'un agent est décrite dans [Déployer le client](#) ; la passer ici ne coûte rien.

Résultat attendu : l'assistant se ferme sur la console, sans agent inscrit pour l'instant.

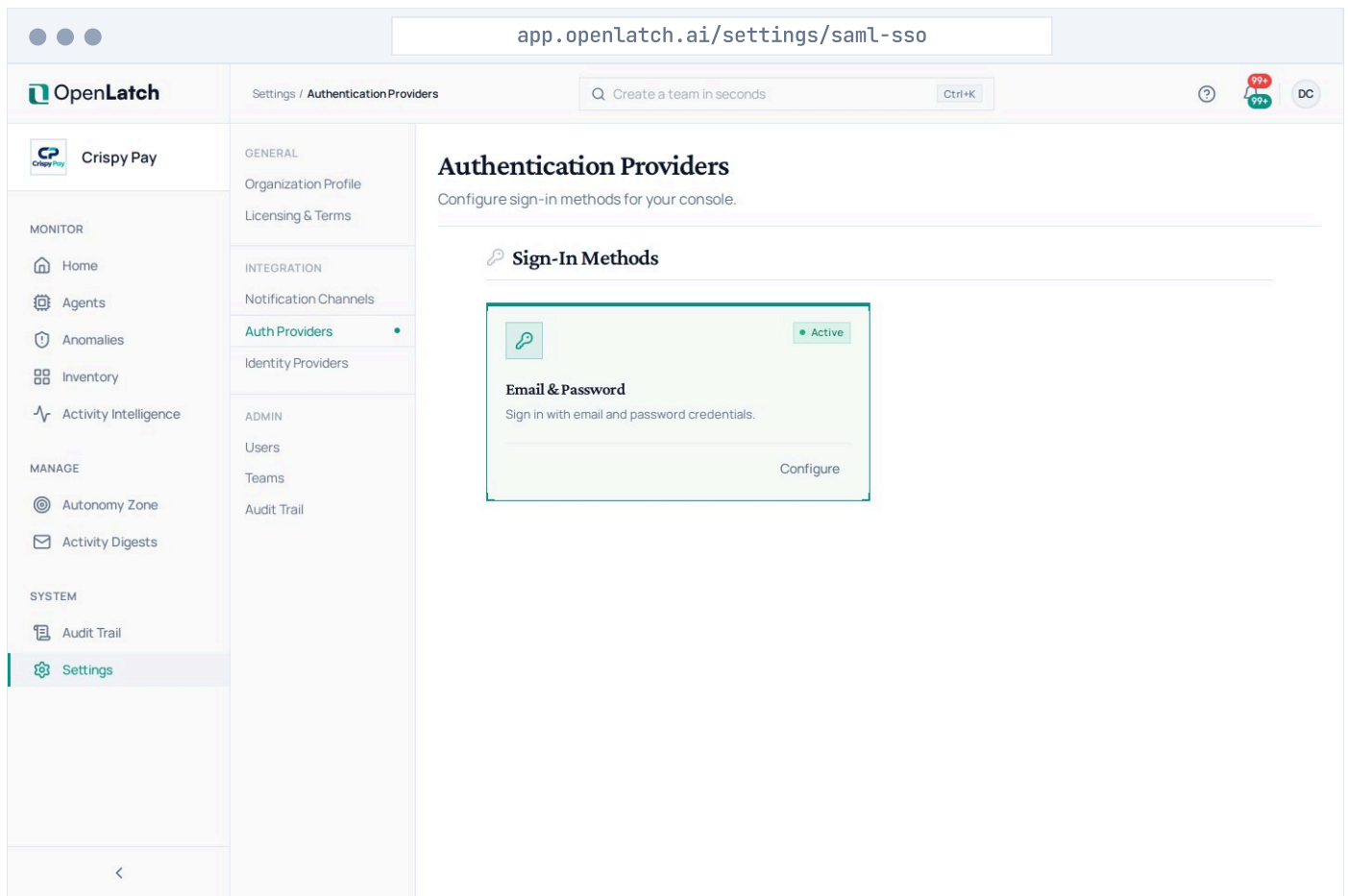
Connecter l'authentification unique

L'authentification unique utilise SAML 2.0, configuré par organisation. Microsoft Entra ID, Okta, PingOne et Active Directory Federation Services se connectent tous par le même écran. Les réglages d'identité demandent le rôle admin ou supérieur.

- 1 **Ouvrez Settings > SAML SSO et ajoutez un fournisseur.** La page montre les informations du fournisseur de services dont aura besoin la personne qui administre votre fournisseur d'identité. Ayez sous la main les métadonnées de votre fournisseur d'identité.
- 2 **Saisissez les informations du fournisseur d'identité.** La console demande l'identifiant d'entité du fournisseur d'identité, son URL d'authentification, son certificat de signature, et les domaines de messagerie autorisés à se connecter. Vous pouvez fournir une URL de métadonnées au lieu de coller le certificat. Choisissez d'abord votre fournisseur d'identité dans la liste : la page affiche un guide de configuration écrit pour ce produit.

L'assertion envoyée par votre fournisseur d'identité doit porter l'adresse e-mail et le nom d'affichage de l'utilisateur ; le guide de configuration nomme les champs exacts.
- 3 **Enregistrez, lancez le test de connexion, puis connectez-vous.** Le test porte sur la configuration enregistrée, pas sur ce qui est encore à l'écran. Il vérifie que l'URL de métadonnées ou l'URL de connexion du fournisseur d'identité répond ; il n'effectue pas de connexion SAML. Connectez-vous ensuite vous-même par SAML, dans une fenêtre de navigation privée, avant de demander à qui que ce soit d'autre de le faire.

Résultat attendu : le test réussit, et votre propre connexion SAML atteint la console.
- 4 **Décidez qui peut rejoindre.** Deux portes contrôlent l'appartenance. Un domaine de messagerie autorisé permet à toute personne ayant une adresse dans ce domaine de s'inscrire dans l'organisation. Une invitation admet une personne nommée. Gardez les domaines autorisés restreints, et appuyez-vous sur les invitations pour tous les autres.



Les réglages SAML SSO, avec les informations du fournisseur de services et les valeurs que fournit votre fournisseur d'identité.

[Configurer l'authentification unique](#) décrit plus en détail les réglages SAML et les autres méthodes de connexion.

Inviter l'équipe et attribuer les rôles

Quatre rôles se succèdent dans un ordre strict de privilèges. Chaque personne invitée commence en viewer : un rôle est accordé délibérément plutôt que retiré après coup.

Rôle	Ce qu'il permet
owner	Tout ce que permet admin, plus la propriété de l'organisation elle-même. Confiez-le à la personne responsable du déploiement
admin	Réglages d'identité, politiques, identifiants du client, conservation et canaux de notification. Accordez-le aux quelques personnes qui mènent le déploiement
editor	Politiques et remplacements de propriétaire d'agent
viewer	Lit les agents, les exécutions, les décisions et le journal d'audit. Suffisant pour tous les autres, y compris un développeur qui vérifie que sa machine est inscrite

- 1 **Invitez par adresse e-mail** sous **Settings > Users**. Chaque invitation produit un lien d'inscription lié à cette adresse. Invitez les personnes qui mènent le déploiement, pas toute l'organisation d'ingénierie.

Résultat attendu : chaque personne invitée apparaît dans la liste des utilisateurs, en attente, avec le rôle viewer.

- 2 **Relevez les rôles qui doivent l'être**. N'accordez admin que là où les réglages d'identité, les identifiants ou la conservation doivent changer. Accordez editor à qui maintient les politiques et le registre des agents. Laissez les autres en viewer.

app.openlatch.ai/settings/users

OpenLatch

Settings / Users

GENERAL

Organization Profile

Licensing & Terms

INTEGRATION

Notification Channels

Auth Providers

Identity Providers

ADMIN

Users

Teams

Audit Trail

Users

Manage team members and invitations.

Search by name or email...

+ Status + Team + Email Created Last Login

	NAME ↕	EMAIL ↕	TEAMS	ROLE	STATUS ↕	LAST LOGIN ↕	CREATED ↕
☐	DO Docs Capture Owner	docs-cap...	Administr...	Owner	Active		
☐	RK Rohan Kapoor	rohan.ka...	Policy Edi...	Editor	Active		
☐	CL Camille Lefèvre	camille.le...	Policy Edi...	Editor	Active		
☐	MW Marcus Washington	marcus...	Policy Edi...	Editor	Active		
☐	IS Ingrid Solberg	ingrid.sol...	Administr...	Admin	Active		
☐	UD User Documentation	user-doc...	Administr...	Admin	Active		
☐	Luc Delsalle	luc.delsal...	Administr...	Owner	Active		

Showing 1 to 7 of 7 users

Les réglages Users : chaque membre et le rôle qu'il détient.

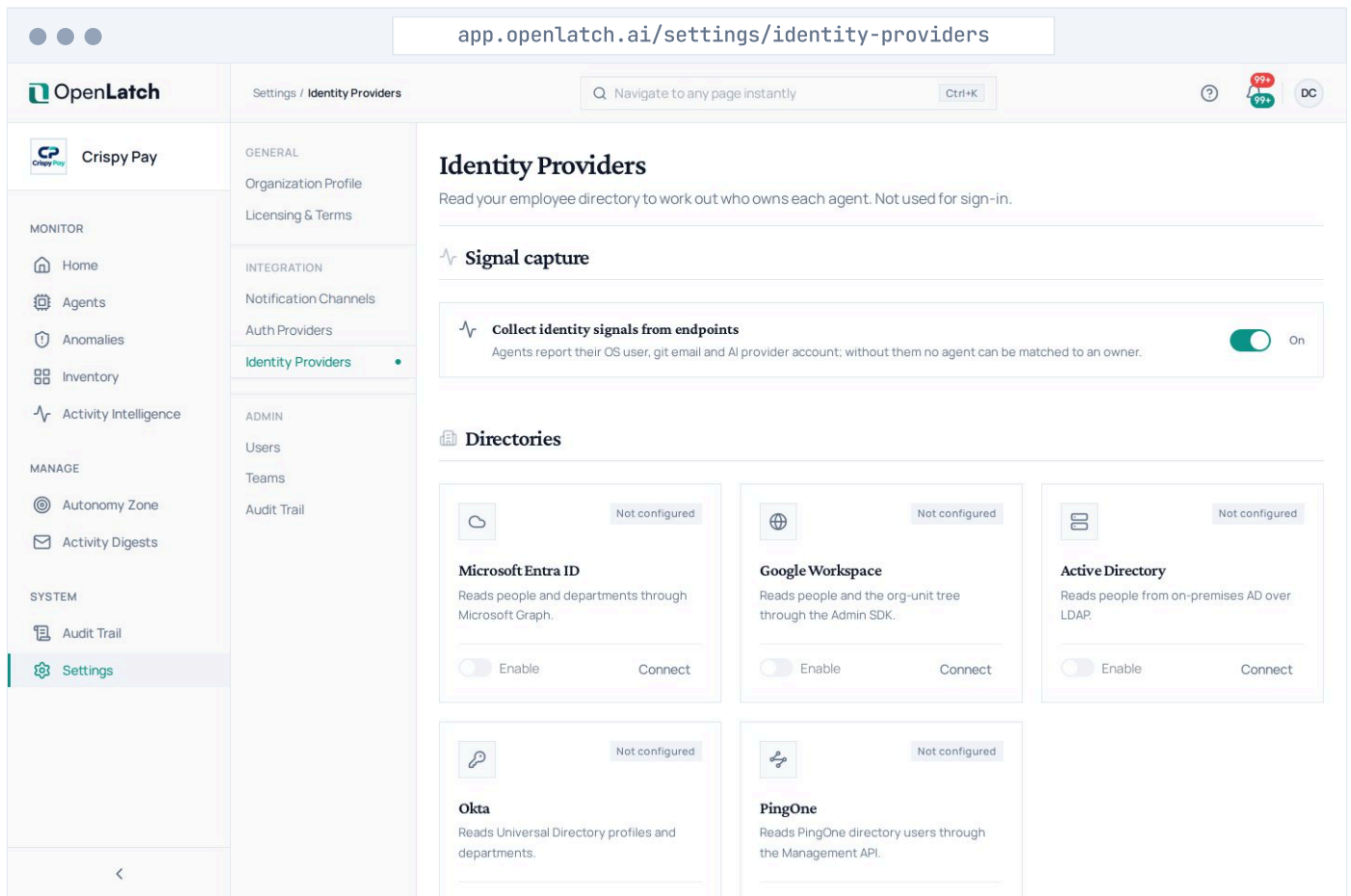
Connecter un annuaire pour la recherche des propriétaires

Cette étape est facultative, et vous pouvez la faire après l'inscription des premières machines. Une connexion d'annuaire rattache le propriétaire de chaque agent connecté, pour que le registre des agents montre une personne plutôt qu'un nom de compte du système d'exploitation. Elle est en lecture seule et ne sert pas à la connexion.

La propriété se rapproche par deux côtés, et les deux doivent être présents. Les machines remontent les signaux qui identifient qui travaille, et l'annuaire transforme ces signaux en une personne nommée. Le côté signaux est un interrupteur sur la même page, **Collect identity signals from endpoints**, activé par défaut. S'il est désactivé, aucun agent ne peut être rattaché à un propriétaire.

- 1 Ouvrez **Settings > Identity Providers** et connectez un annuaire : Microsoft Entra ID, Google Workspace, Active Directory, Okta ou PingOne. Chacun est une source en lecture seule ; aucun ne crée d'utilisateurs, et aucun ne copie l'annuaire dans OpenLatch.

- 2 Recherchez une personne que vous savez présente dans l'annuaire. Une organisation sans annuaire connecté est un état pris en charge, et un agent dont le propriétaire ne peut pas être déterminé est enregistré comme non attribué plutôt que deviné.



Identity Providers : l'interrupteur qui permet aux machines de signaler qui travaille, et les annuaires qui en font une personne nommée.

Définir la conservation et les notifications

Définissez ces deux réglages avant le début du déploiement ; une revue de sécurité vous interrogera sur les deux.

- 1 **Définissez la conservation du journal d'audit** sous **Settings > Audit Trail**. Le journal d'audit est en ajout seul et chaîné par hachage. Choisissez une durée de conservation, décidez de la purge automatique, et définissez le format d'export qu'attend votre processus de preuves.

Réglage	Valeurs
Conservation	90, 180, 365 ou 730 jours, ou une durée personnalisée. La console recommande 365
Purge automatique	Activée ou non. Activée, elle supprime toutes les 24 heures les enregistrements au-delà de la durée de conservation
Export	CSV ou JSON

- Ajoutez un canal de notification** sous **Settings > Notification Channels**. Pointez-le vers le canal dans lequel l'équipe de déploiement travaille déjà, comme un canal Slack ou un webhook HTTPS, et envoyez un test.

Résultat attendu : l'envoi de test arrive.

The screenshot shows the OpenLatch web interface at the URL `app.openlatch.ai/settings/audit-trail`. The left sidebar contains navigation menus for Monitor, Manage, and System. The main content area is titled "Audit Trail" and includes the following sections:

- Retention Policy:** A section with a subtitle "Configure retention policies, data management, and compliance settings for the audit trail." It features a "Retention Period" selector with options: 90 days, 180 days, 365 days (recommended), 730 days, and Custom.
- Auto-Purge:** A toggle switch that is currently turned on. The text below it states: "When enabled, audit events older than the retention period are automatically removed every 24 hours."
- Export Preferences:** A section with the subtitle "Choose the default format for audit trail exports." It has two radio button options: "CSV — Spreadsheet-compatible, human-readable" (which is selected) and "JSON — Machine-readable, preserves nested data".
- Hash Chain Integrity:** A section with a subtitle "Verify the cryptographic hash chain that ensures audit events have not been tampered with." It includes a "Verify Integrity" button.

At the bottom right of the page, there are "Cancel" and "Save Configuration" buttons.

Les réglages Audit Trail : durée de conservation, purge automatique et format d'export.

Créer l'identifiant du client

Le client s'authentifie auprès de la console avec une clé d'API. Vous la créez depuis une session de navigateur connectée, sur la page **API Keys** de votre compte, et vous la révoquez depuis la même page.

- 1 **Créez une clé par vague de déploiement**, nommée d'après sa vague. Une clé par vague signifie qu'une clé à révoquer ne concerne qu'une seule vague.

Résultat attendu : la valeur de la clé est affichée une seule fois, en entier, à la création. Ensuite, la liste montre son nom, une valeur masquée, sa date de création et sa dernière utilisation.

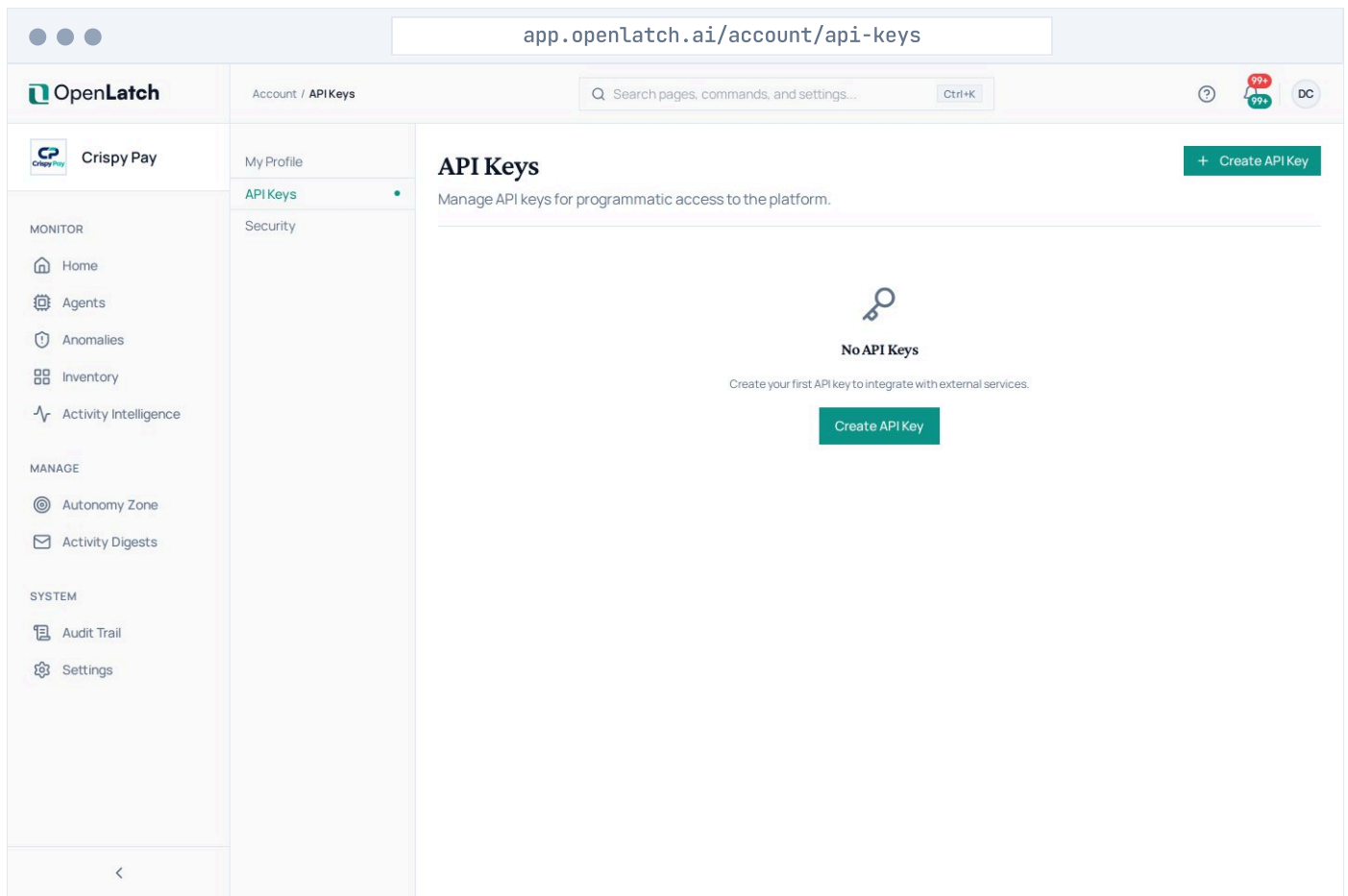
- 2 **Stockez-la dans le coffre de secrets de votre organisation**. L'installation gérée lit la clé dans l'environnement, sous le nom `OPENLATCH_API_KEY`. Elle n'est jamais écrite dans un script, un paquet ou une ligne de commande.

- 3 **Gardez-la jusqu'à ce que les machines de la vague en détiennent une autre**. Chaque machine enregistre la clé avec laquelle elle a été inscrite et s'authentifie avec elle : révoquer une clé de vague déconnecte toutes les machines de cette vague. Avant d'en révoquer une, donnez une nouvelle clé à ses machines : définissez-la comme `OPENLATCH_API_KEY` et relancez `openlatch init --yes` sur chaque machine. La date de dernière utilisation vous indique si une clé est encore présentée avant de la révoquer.

Résultat attendu : la clé apparaît comme révoquée, et tout ce qui la présente encore, une installation ou une machine, est refusé.

Attention

Une clé passée en ligne de commande atterrit dans l'historique du shell, la liste des processus et les journaux de gestion de configuration. Le client refuse un identifiant passé ainsi. Fournissez-le par l'environnement, depuis votre coffre de secrets.



API Keys : chaque clé listée avec une valeur masquée après sa création.

[Gérer les clés d'API](#) décrit les clés plus en détail.

En bref

- OpenLatch crée l'organisation et invite son propriétaire ; l'assistant de bienvenue est l'endroit où vous la nommez.
- SAML 2.0 se configure par organisation et se vérifie par un test qui nomme l'étape en échec. Les domaines de messagerie autorisés et les invitations sont les deux portes de l'appartenance.
- Les rôles sont owner, admin, editor, viewer, et chaque personne invitée commence en viewer. Une connexion d'annuaire est facultative, en lecture seule, et rattache le propriétaire d'un agent.
- Définissez la conservation du journal d'audit, avec purge automatique, avant le début du déploiement, et dirigez les notifications vers le canal déjà utilisé.
- Créez un identifiant client par vague de déploiement, fournissez-le par le coffre de secrets sous le nom OPENLATCH_API_KEY, et révoquez-le quand la vague se termine.

Déployer le client

Installez le client sur chaque machine qui exécute un agent. Le client est le point de décision sur le chemin d'exécution de l'agent : l'installer connecte chaque agent déjà présent sur la machine, et dès lors chaque action couverte est décidée sur la machine, avant de s'exécuter, même quand la console est injoignable.

Avant de commencer, vérifiez ces éléments :

#	Élément	D'où il vient	Bloque
1	L'organisation existe, et les personnes qui mènent le pilote peuvent se connecter	Configurer l'organisation	Toutes les voies
2	Une clé d'API par vague de déploiement, conservée dans votre coffre de secrets	La page API Keys de votre compte	Les voies gérées uniquement
3	Au moins un agent installé sur la machine cible et connecté à son propre compte	Le programme d'installation de l'agent	Toutes les voies
4	Flux sortants approuvés, avec les réponses de modèles en flux exemptées de mise en tampon et d'inspection	Exigences réseau	Toutes les voies
5	Le type de proxy en vigueur sur le sous-réseau cible : direct, Basic, Negotiate, PAC, ou inspection TLS avec son certificat racine	L'équipe qui gère votre proxy	Les voies gérées

Un agent absent au moment de l'installation n'est pas connecté. Installez d'abord les agents, ou relancez l'installation ensuite ; la relancer est idempotent, donc une répétition n'est jamais destructrice.

Choisir une voie d'installation

Quatre voies mènent au même état final. Choisissez selon la provenance de l'identifiant et le nombre de machines que la voie doit atteindre, pas selon le système d'exploitation. La plupart des parcs en utilisent

deux : le parcours développeur pour le pilote, et une voie gérée pour le reste.

Voie	Idéale pour	Portée	Identifiant
Parcours développeur	Vague 1 : une équipe pilote qui s'inscrit elle-même	Environ dix machines	Connexion par navigateur
Script de gestion des postes, contexte utilisateur	Parcs déjà gérés par Intune ou Jamf	Le parc entier	Clé de vague depuis le coffre de secrets
Gestion de configuration ou étape de pipeline	Parcs Linux et parcs mixtes dotés d'une automatisation existante, comme Ansible	Le parc entier	Clé de vague depuis une recherche dans un coffre
Image de conteneur	Machines de build et exécuteurs d'intégration continue	Une inscription par tâche	Secret d'exécution, monté par tâche

Trois commandes installent le client. Laquelle vous lancez dépend de ce que la machine possède déjà, pas de la taille du parc :

Machine	Commande
macOS ou Linux, aucun environnement d'exécution requis	Le script <code>install.sh</code> , ci-dessous
Windows, aucun environnement d'exécution requis	Le script <code>install.ps1</code> , ci-dessous
Toute machine qui dispose déjà de Node.js	<code>npx @openlatch/client init</code> . Node.js est ce dont npx a besoin ; le client est un binaire natif et n'a besoin d'aucun environnement d'exécution

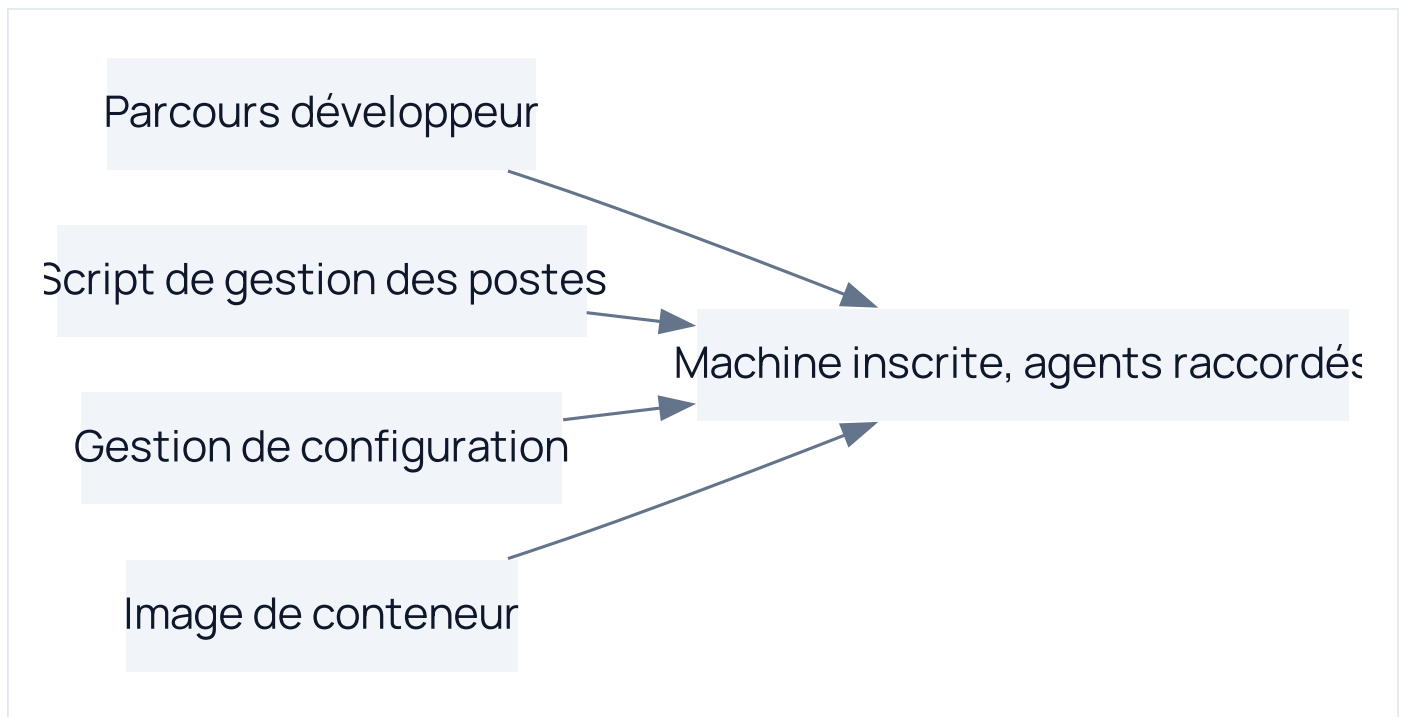


Figure 1. Quatre voies d'installation, un seul état final : une machine inscrite avec chaque agent raccordé.

Installer une machine

Une commande inscrit une machine. Les scripts d'installation embarquent le client : la machine n'a besoin d'aucun environnement d'exécution. Lancez la ligne de la plateforme devant vous.

Sur macOS et Linux :

```
curl -LsSf https://openlatch.ai/install.sh | sh
```

Sur Windows :

```
powershell -ExecutionPolicy Bypass -c "irm https://openlatch.ai/install.ps1 | iex"
```

Sur une machine qui dispose déjà de Node.js :

```
npx @openlatch/client init
```

Les trois mènent au même état final : une équipe qui utilise déjà l'une d'elles la garde. Les scripts installent le client dans le répertoire personnel de l'utilisateur, ajoutent ce répertoire au PATH, et préviennent que le shell en cours ne le verra qu'après redémarrage. Ils lancent ensuite `openlatch init` par son chemin complet, pour que l'inscription se termine dans la même exécution. Si votre parc n'atteint pas le registre public de paquets, `OPENLATCH_NPM_REGISTRY` pointe le script et le client vers votre miroir.

Ce que fait l'installation

Étape	Ce qui se passe	Résultat attendu
Vérifier la route de sortie	La vérification des flux sortants passe en premier : une route bloquée échoue avant toute modification de la machine. Rien n'est laissé à moitié installé	La route de sortie est indiquée, avec la source qui l'a définie
Connecter chaque agent	Chaque agent installé sur la machine est connecté, selon la convention de configuration propre à cet agent. Le contenu existant de ces fichiers est préservé	Une ligne par agent trouvé, et une ligne explicite quand aucun n'est trouvé
S'authentifier	L'ordre est <code>OPENLATCH_API_KEY</code> dans l'environnement, puis un identifiant déjà présent dans le magasin d'identifiants du système, puis la connexion par navigateur. Une erreur réseau conserve un identifiant enregistré ; seul un refus déclenche une nouvelle connexion. Avec plusieurs organisations, la connexion demande laquelle, sauf si <code>--org</code> ou <code>OPENLATCH_ORG</code> la nomme. Sans personne à qui demander et sans identifiant, <code>init</code> ne lance jamais de connexion dans le navigateur et s'arrête sur <code>0L-1600</code>	Le compte et l'organisation sont affichés, ou une fenêtre de navigateur s'ouvre dans un terminal interactif
Interroger sur la télémétrie produit	La télémétrie est activée par défaut et peut être refusée ; le choix est proposé avant toute écriture de configuration d'agent. Elle porte des noms de commandes, des durées et des informations d'erreur, jamais de contenu	Une seule question, et le choix dans le rapport final
Démarrer avec la session de l'utilisateur	Le client s'enregistre auprès du gestionnaire de services du système d'exploitation, au niveau de l'utilisateur. Un échec ici n'est pas fatal : une session sans interface et un planificateur restreint sont tous deux surmontables	Le rapport indique que le client redémarrera après une déconnexion ou un redémarrage
Prouver, puis rendre compte	L'installation confirme que le client tourne, que chaque agent connecté répond et que la console est joignable	Un rapport final qui nomme tout ce qui demande attention, avec le remède à côté

Note

Certains agents demandent au développeur de confirmer la connexion une fois, au premier lancement après l'installation. D'ici là, l'agent est connecté sur le papier mais pas dans les faits. `openlatch status` montre quels agents attendent encore, et [Niveaux de contrôle](#) explique comment chaque agent reçoit un verdict. Une machine n'est pas terminée tant que le développeur n'a pas ouvert l'agent une fois.

Confirmer l'installation

```
openlatch status
openlatch doctor
```

Une machine est installée quand `openlatch doctor` ne signale aucun problème et que la machine apparaît dans le registre des agents de la console avec ses agents et un propriétaire. [Vérifier et déployer](#) couvre cette vérification machine par machine et sur une vague entière.

Derrière un proxy d'entreprise

Chaque appel sortant du client emprunte une seule route de sortie. Le trafic de bouclage sur la machine ne l'emprunte jamais. La route est résolue réglage par réglage : une option peut fixer l'URL du proxy pendant que le paquet de certificats vient d'ailleurs.

Rang	Source	À utiliser pour
1	Les options de la commande <code>openlatch init</code>	Fixer la route au moment de l'inscription
2	<code>OPENLATCH_PROXY</code> et ses variables sœurs	Le seul canal autorisé à porter un identifiant de proxy
3	La configuration propre du client, écrite à l'installation	Ce que le client lit quand il démarre avec la session, là où aucune variable de shell n'existe
4	La famille standard de variables <code>https_proxy</code>	Un parc qui les exporte déjà pour tous les outils
5	La découverte par le système d'exploitation	Une inscription sans configuration sur un poste géré
6	Direct	Le dernier recours. Il peut être refusé, pour qu'une route non résolue échoue au lieu de passer en direct

Ce qui a permis l'installation est écrit dans la configuration propre du client, avec sa source. Le client démarre avec la session de l'utilisateur et n'hérite d'aucune variable du shell d'installation : une route qui n'existait que dans une variable d'environnement laisserait la machine passer en direct après la fin de la session.

```
openlatch init --proxy http://proxy.corp:8080 --ca-bundle /etc/ssl/corp-root.pem
openlatch init --proxy http://proxy.corp:8080 --proxy-auth negotiate --proxy-spn
HTTP/proxy.corp
```

Le paquet de certificats est fusionné par-dessus le magasin de confiance du système, sans jamais le remplacer, et la vérification n'est jamais désactivée. Negotiate utilise l'identité de la session ouverte, avec un nom de principal de service qui vaut par défaut HTTP/ suivi de l'hôte du proxy. NTLM est refusé par choix de conception : un parc uniquement NTLM demande un changement côté proxy avant l'inscription.

Un identifiant passé dans `--proxy` est refusé, car les arguments de ligne de commande sont lisibles par tous les processus de l'hôte. Fournissez-le par `OPENLATCH_PROXY` ou répondez à la question dans un terminal. Dans les deux cas, le mot de passe va dans le magasin d'identifiants du système, n'atterrit jamais dans un fichier sur disque, et est masqué dans chaque ligne de journal.

Attention

Sur un hôte derrière un proxy, les adresses de bouclage doivent figurer dans la liste d'exceptions du proxy du shell qui lance vos agents. Certains agents reçoivent cette entrée automatiquement à l'installation. Quand un agent n'a aucun endroit pour la recevoir, exportez `NO_PROXY=127.0.0.1,localhost` dans le shell qui le lance. Sans cela, le trafic de l'agent vers le client part vers le proxy d'entreprise au lieu de rester sur la machine, et le proxy répond par une erreur.

Installations gérées pour les parcs

Attention

Utilisez une clé d'API par vague de déploiement, lue depuis votre coffre de secrets et révoquée dans la console quand la vague se termine. N'écrivez jamais la clé dans un script, un paquet ou une ligne de commande. Chaque exemple ci-dessous place `OPENLATCH_API_KEY` dans l'environnement de l'installation. Quand la plateforme de gestion peut transmettre une valeur à un script, comme Jamf avec le paramètre 4, utilisez ce canal ; quand elle ne le peut pas, comme Intune, le script va chercher la clé dans le coffre de secrets en première action. Considérez le corps d'un script de déploiement comme lisible : une plateforme de gestion le conserve en clair et peut l'écrire dans les journaux de l'appareil.

Une installation gérée lance le même script avec trois ajouts :

Ajout	Effet
<code>OPENLATCH_INSTALL_YES</code>	Rend l'installation non interactive
<code>OPENLATCH_NO_INIT</code>	Retient l'inscription que le script lancerait sinon à la fin, pour que votre script lance lui-même <code>openlatch init --yes</code> . Avec <code>--yes</code> , l'inscription ne pose jamais de question : elle échoue et nomme ce qui lui manquait
<code>OPENLATCH_API_KEY</code> dans l'environnement	L'identifiant arrive depuis votre coffre de secrets plutôt que par un navigateur

Lancez `openlatch doctor` en dernière ligne de chaque script et collectez sa sortie : une machine installée mais pas entièrement connectée reste ainsi visible au lieu d'être comptée comme terminée.

Script de gestion des postes, contexte utilisateur (Windows)

```

# Platform script, user context. Runs silently and cannot prompt.
$ErrorActionPreference = "Stop"

# The platform has no field that sets an environment variable, so the script
# fetches the wave key itself, as its first action. Replace this call with the
# one your secret store provides.
$env:OPENLATCH_API_KEY = Get-WaveKeyFromSecretStore
if (-not $env:OPENLATCH_API_KEY) {
    Write-Output "the wave key could not be read from the secret store"; exit 1
}

$env:OPENLATCH_INSTALL_YES = "1"
$env:OPENLATCH_NO_INIT     = "1"

powershell -ExecutionPolicy Bypass -c "irm https://openlatch.ai/install.ps1 | iex"
if ($LASTEXITCODE -ne 0) { exit $LASTEXITCODE }

$openlatch = "$env:LOCALAPPDATA\openlatch\bin\openlatch.exe"
& $openlatch init --yes
if ($LASTEXITCODE -ne 0) { exit $LASTEXITCODE }
& $openlatch doctor
exit $LASTEXITCODE

```

Intune exécute les scripts en silence et ne prend pas en charge l'installation interactive : c'est pourquoi l'installation est non interactive et --yes est obligatoire. L'extension de gestion récupère les affectations selon son propre calendrier : prévoyez une vague en jours plutôt qu'en minutes. Un script en contexte utilisateur demande un utilisateur connecté, et les scripts de plateforme n'atteignent pas les appareils seulement inscrits auprès de Microsoft Entra ID ; pour atteindre les appareils personnels inscrits, utilisez la voie d'application Win32 avec un comportement d'installation User, que l'absence d'élévation rend possible.

Aucun réglage Intune ne définit de variable d'environnement pour un script de plateforme. C'est pourquoi l'appel au coffre de secrets est la première action du script. Passer la clé en argument n'est pas une option : le client refuse un identifiant dans ses propres arguments.

Script de gestion des postes, contexte utilisateur (macOS)

Jamf transmet sa première valeur personnalisée dans le paramètre 4, ce qui garde la clé hors du corps du script. La règle n'a besoin d'aucune étape de création de paquet, car l'installation s'enregistre dans le répertoire personnel de l'utilisateur. Associez la règle à un déclencheur au niveau de l'utilisateur pour que l'inscription s'exécute en tant que développeur et non en tant que root.

```
#!/bin/zsh
set -eu

# Parameter 4 carries the wave key, or replace this line with a secret-store call.
export OPENLATCH_API_KEY="${4:?wave key was not supplied}"

export OPENLATCH_INSTALL_YES=1
export OPENLATCH_NO_INIT=1

curl -LsSf https://openlatch.ai/install.sh | sh

"$HOME/.local/bin/openlatch" init --yes
"$HOME/.local/bin/openlatch" doctor
```

Gestion de configuration ou étape de pipeline

```
- name: Install the OpenLatch client
  ansible.builtin.shell: curl -LsSf https://openlatch.ai/install.sh | sh
  args:
    creates: "{{ ansible_env.HOME }}/.local/bin/openlatch"
  environment:
    OPENLATCH_INSTALL_YES: "1"
    OPENLATCH_NO_INIT: "1"

- name: Enroll the machine
  ansible.builtin.command: "{{ ansible_env.HOME }}/.local/bin/openlatch init --yes"
  environment:
    # Or a vault lookup, so the value never lands in the play or in a fact.
    OPENLATCH_API_KEY: "{{ lookup('env', 'OPENLATCH_API_KEY') }}"

- name: Report what the machine looks like
  ansible.builtin.command: "{{ ansible_env.HOME }}/.local/bin/openlatch doctor"
  changed_when: false
```

Lancez l'inscription sur un échantillon de la vague avant l'exécution complète, et lisez la sortie de la tâche de rapport sur ces machines. Un type de proxy que l'installation ne sait pas découvrir apparaît là, plutôt que sur toutes les machines à la fois.

Image de conteneur et intégration continue

```
FROM debian:bookworm-slim
RUN apt-get update \
    && apt-get install -y --no-install-recommends ca-certificates curl \
    && rm -rf /var/lib/apt/lists/*

ENV OPENLATCH_INSTALL_YES=1 \
    OPENLATCH_NO_INIT=1 \
    OPENLATCH_NO_MODIFY_PATH=1 \
    OPENLATCH_INSTALL_DIR=/usr/local/bin

RUN curl -LsSf https://openlatch.ai/install.sh | sh
COPY entrypoint.sh /usr/local/bin/entrypoint.sh
ENTRYPOINT ["/usr/local/bin/entrypoint.sh"]
```

entrypoint.sh

```
#!/bin/sh
set -eu
: "${OPENLATCH_API_KEY:?the runtime secret was not mounted}"
openlatch init --yes --no-persistence
exec "$@"
```

```
- name: Enroll the runner
  env:
    OPENLATCH_API_KEY: ${ secrets.OPENLATCH_API_KEY }
    OPENLATCH_INSTALL_YES: "1"
    OPENLATCH_NO_INIT: "1"
  run: |
    curl -LsSf https://openlatch.ai/install.sh | sh
    "$HOME/.local/bin/openlatch" init --yes --no-persistence
    "$HOME/.local/bin/openlatch" status
```

L'image de base n'embarque rien pour le client : le script apporte le client avec lui. L'identifiant est monté à l'exécution, donc aucune couche de l'image ne le contient. `OPENLATCH_NO_MODIFY_PATH` empêche l'installation d'écrire une ligne `PATH` dans un profil de shell qu'aucun shell interactif ne lira, et `OPENLATCH_INSTALL_DIR` place le client là où le `PATH` de l'image cherche déjà. `--no-persistence` ignore le gestionnaire de services, car un conteneur n'a pas de session utilisateur dans laquelle démarrer. La télémétrie produit est désactivée en intégration continue sans configuration. Les exécuteurs éphémères s'inscrivent à chaque tâche, et comme une installation répétée est sans risque, une tâche relancée l'est aussi.

Ce que le client installe sur une machine

Le client s'installe et s'exécute au niveau de l'utilisateur. Aucun droit d'administrateur n'est requis, rien n'est écrit dans un répertoire système, et aucun service n'est enregistré au niveau de la machine.

Élément	Détail
Une seule unité installée	Le client OpenLatch, installé une fois par machine qui exécute un agent
Où il arrive	Les scripts placent le client dans <code>~/ .local/bin</code> sur macOS et Linux, et dans <code>%LOCALAPPDATA%\openlatch\bin</code> sur Windows, puis ajoutent ce répertoire au PATH. <code>OPENLATCH_INSTALL_DIR</code> le déplace. Ce n'est volontairement pas le répertoire d'état ci-dessous
Fichiers	Le client conserve sa configuration et ses enregistrements locaux dans un seul répertoire : <code>~/ .openlatch</code> sur macOS et Linux, <code>%APPDATA%\openlatch</code> sur Windows. <code>OPENLATCH_DIR</code> le déplace. Tout élément sensible qu'il contient n'est lisible que par son propriétaire
Configuration des agents	Le fichier de configuration de chaque agent connecté reçoit une entrée qui pointe vers le client. Le contenu existant est préservé, et l'état antérieur est enregistré pour pouvoir être restauré
Démarrage	Le client démarre avec la session de l'utilisateur, sous le gestionnaire de services du système d'exploitation
Identifiants	La clé d'API, et tout identifiant de proxy, sont conservés dans le magasin d'identifiants du système
Posture réseau	N'écoute que sur l'interface de bouclage. Aucun port entrant n'est ouvert

`openlatch uninstall` déconnecte chaque agent trouvé sur la machine, arrête le client, et restaure ce qu'il avait modifié dans la configuration de chaque agent. `openlatch uninstall --purge` supprime aussi le répertoire d'état et l'identifiant enregistré. Une purge ne supprime pas les fichiers installés : ils se trouvent hors du répertoire d'état, si bien qu'une purge les laisse, ainsi que la ligne PATH ajoutée par le script. [Exploiter](#) couvre la suppression complète.

En bref

- Quatre voies d'installation mènent à un seul état final. Choisissez selon la provenance de l'identifiant et le nombre de machines à atteindre, pas selon le système d'exploitation.

- Les scripts d'installation inscrivent une machine sur macOS, Windows et Linux sans aucun environnement d'exécution ; `npx @openlatch/client init` mène au même état final là où Node.js est présent. Dans tous les cas, l'installation détecte les agents présents et connecte chacun d'eux, et la relancer est sans risque.
- Certains agents demandent au développeur de confirmer la connexion une fois. Une machine n'est pas terminée tant que ce n'est pas fait.
- Une machine est installée quand `openlatch doctor` ne signale aucun problème et que la machine apparaît dans le registre des agents de la console avec ses agents et un propriétaire.
- Une clé d'API par vague, lue depuis votre coffre de secrets à l'exécution, jamais dans un script, un paquet ou une ligne de commande.

Vérifier et déployer

Une machine est déployée quand `openlatch doctor` ne signale aucun problème et que la console détient son enregistrement. Cette page prouve les deux, puis élargit le déploiement en trois vagues.

Vérifier une machine

Lancez les deux commandes sur chaque machine installée, dans cet ordre.

```
openlatch status
openlatch doctor
```

`openlatch status` répond à une question : cette machine est-elle couverte en ce moment ? Un bon résultat montre la machine connectée à la console, des politiques présentes sur la machine, et chaque agent de la machine raccordé.

`openlatch doctor` répond à la question plus difficile : ce compte rendu est-il vrai ? Il affiche une cause et un remède pour chaque point relevé. Un bon résultat, c'est qu'il ne relève aucun problème. C'est l'étape de recette d'une machine : comptez une machine avec un point non résolu comme une machine en échec dans le bilan de la vague. Quand il relève quelque chose, suivez le remède qu'il indique et relancez-le, jusqu'à ce qu'il soit propre.

Ce que vous confirmez	À quoi ressemble un bon résultat
La machine est connectée	Elle a atteint la console et y détient un enregistrement à jour
Les politiques sont présentes	Des politiques sont sur la machine. C'est ce qui lui permet de continuer à décider quand la console est injoignable
Les agents sont raccordés	Chaque agent installé sur la machine est raccordé, et aucun n'attend encore que son développeur confirme la connexion

Attention

Une machine peut tourner sans rien couvrir, et vue de l'extérieur elle ressemble exactement à une machine qui fonctionne : le client tourne, la console détient un enregistrement, et un agent de la machine n'a jamais été raccordé. Ne passez pas à l'étape suivante du déploiement derrière une installation que openlatch doctor n'a pas validée.

Constituer une archive de diagnostic

Quand une machine a un problème que le remède affiché ne permet pas de résoudre, constituez une archive de diagnostic.

```
openlatch doctor --rescue
```

La commande produit une archive ZIP expurgée qui contient le diagnostic, la configuration du client et une fenêtre récente des enregistrements locaux de la machine.

Option	Effet
<code>--since 24h</code>	Borne la fenêtre d'enregistrements locaux incluse dans l'archive
<code>--yes</code>	Saute la confirmation, pour une exécution sans surveillance

L'archive permet d'examiner un problème à partir des éléments de la machine elle-même plutôt que d'une description du symptôme. Elle est expurgée dans ce but : envoyez l'archive plutôt que le répertoire lui-même.

Note

Ouvrez une demande de support avec l'archive jointe, le système d'exploitation de la machine, et l'heure du problème avec son fuseau horaire. Contactez votre référent technique OpenLatch par le canal convenu avec votre organisation. Envoyez les signalements de sécurité à security@openlatch.ai.

Confirmer dans la console

Un compte rendu propre sur la machine est la moitié de la vérification. L'autre moitié, c'est la console qui détient un enregistrement pour la machine, avec les agents détectés et la personne à qui ses actions sont attribuées.

1 Ouvrez le registre des agents. Connectez-vous sur `app.openlatch.ai` et ouvrez **Agents**.

Résultat attendu : une ligne par agent qui a remonté des données.

2 Trouvez la machine. Cherchez par nom d'hôte ou par l'utilisateur du système d'exploitation sous lequel l'installation a tourné.

Résultat attendu : la machine apparaît dans les 24 heures suivant l'installation, et en quelques minutes sur un réseau sain.

3 Confirmez les agents de la machine. Ouvrez un agent et lisez ses détails, dont son niveau de contrôle.

Résultat attendu : chaque agent installé sur cette machine est listé, chacun avec son niveau de contrôle. Un agent manquant est un problème d'installation, pas un problème de console : revenez aux vérifications de la machine ci-dessus.

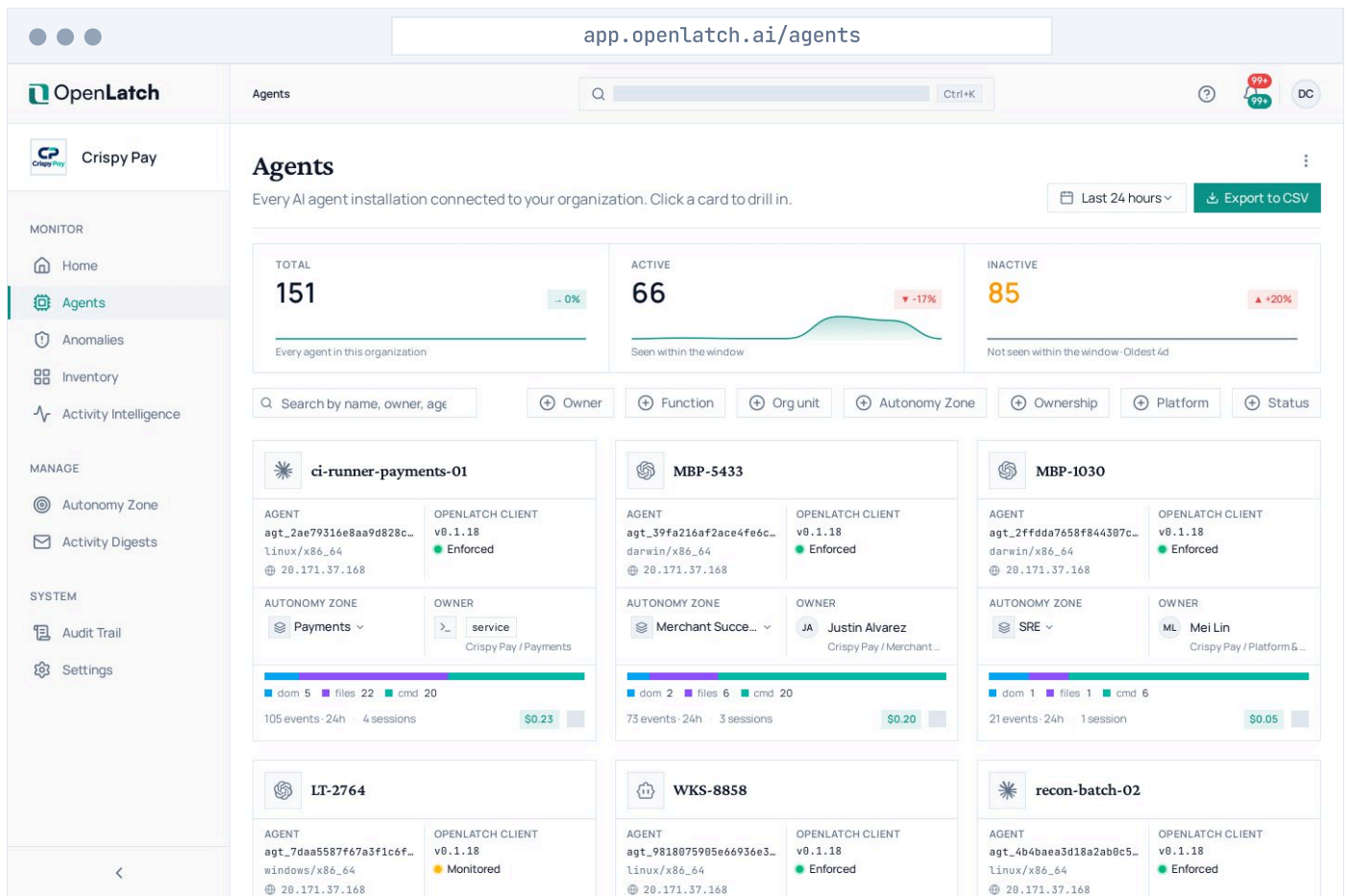
4 Confirmez le propriétaire déterminé. L'onglet **Identity** de l'agent nomme la personne à qui ses actions sont attribuées, avec son rôle et sa place dans l'organisation. Avec un annuaire connecté, le propriétaire est déterminé par une recherche en lecture seule ; sans annuaire, OpenLatch s'appuie sur ce que la machine remonte.

Deux commandes se trouvent à côté du nom : l'une explique comment ce propriétaire a été déterminé, l'autre permet de le remplacer à la main. Corriger la propriété est le travail du rôle editor.

Résultat attendu : une personne nommée plutôt qu'un champ vide.

5 Confirmez que les enregistrements arrivent. Lisez l'heure de dernière activité de l'agent pendant qu'un développeur travaille sur cette machine.

Résultat attendu : l'heure avance. Un enregistrement qui cesse d'avancer renvoie à la route de sortie de la machine.



Agents : une ligne par agent qui a remonté des données, avec son niveau de contrôle et son propriétaire.

The screenshot displays the OpenLatch web application interface. The top navigation bar shows the URL `app.openlatch.ai/agents/ci-runner-payments-01/iden...`. The left sidebar contains a menu with sections: MONITOR (Home, Agents, Anomalies, Inventory, Activity Intelligence), MANAGE (Autonomy Zone, Activity Digests), and SYSTEM (Audit Trail, Settings). The 'Agents' section is currently selected. The main content area is titled 'ci-runner-payments-01' and shows the 'Identity' tab. This tab displays the agent's ownership information, including the Jenkins CI service, the user Camille Lefèvre, and the agent itself. The 'Ownership' section shows a hierarchy: Crispy Pay (Payments) -> Camille Lefèvre (VP, Payments Engineering) -> ci-runner-payments-01 (agt_2ae79316...). The agent is currently in an 'IDLE' state.

L'onglet Identity d'un agent : la personne à qui ses actions sont attribuées, et sa place dans l'organisation.

Déployer par vagues

Trois vagues de deux semaines, chacune prouvant quelque chose dont la suivante dépend. La vague 1 prouve l'installation par le parcours développeur. La vague 2 prouve la voie gérée. La vague 3 prouve le périmètre. Émettez un identifiant client par vague et révoquez-le dans la console quand la vague se termine.

Attribut	Vague 1	Vague 2	Vague 3
Périmètre	Environ dix machines de développeurs	Environ cinquante machines	Le périmètre complet
Voie d'installation	Parcours développeur, dans le terminal du développeur	Voie gérée, sans surveillance	Voie gérée, sans surveillance
Durée	Deux semaines	Deux semaines	Deux semaines
Activité	Inscrire chaque machine, la vérifier, relancer les agents qui attendent encore leur développeur, et noter ce que les personnes rencontrent	Emballer l'installation, la prouver sur une machine de chaque système d'exploitation, puis la lancer sans surveillance et lire le rapport qu'elle collecte	Lancer le même paquet sur tout le périmètre, et relancer les machines qui ne remontent jamais
Critères d'entrée	L'organisation est configurée, et un identifiant est émis pour la vague	La vague 1 a rempli ses critères de sortie, et le script géré tourne proprement sur une machine de chaque système d'exploitation	La voie gérée est prouvée sur chaque système d'exploitation du parc, et le type de proxy est confirmé sur chaque sous-réseau concerné
Critères de sortie	Chaque machine inscrite remonte à la console dans les 24 heures suivant l'installation ; openlatch doctor ne signale aucun problème sur chacune ; aucun agent n'attend son développeur ; la route du proxy est confirmée sur chaque système d'exploitation de la vague	Les quatre mêmes, plus chaque machine installée sans qu'un développeur ouvre de terminal	Les quatre mêmes, plus la décision de passer en exploitation courante

Attribut	Vague 1	Vague 2	Vague 3
Retour arrière	openlatch uninstall sur la machine concernée	Le même, livré par l'outil qui a installé	Le même, livré par l'outil qui a installé

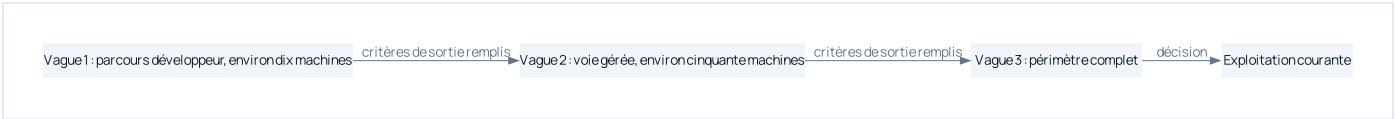


Figure 1. Trois vagues, ce que chacune prouve, et la porte entre elles.

Problèmes les plus probables

Symptôme	Cause probable	Que faire
La machine remonte, mais un de ses agents n'est pas raccordé	Cet agent n'était pas installé au moment de l'installation, ou sa configuration se trouve à un chemin personnalisé	Installez l'agent, puis relancez <code>openlatch init</code> sur la machine. Le client est déjà là : aucun téléchargement n'est nécessaire. Si le chemin est personnalisé, définissez-le dans l'environnement de l'inscription
Un agent est indiqué en attente	Certains agents demandent au développeur de confirmer la connexion une fois, au premier lancement après l'installation	Faites ouvrir l'agent une fois au développeur pour qu'il confirme. Si le parc verrouille la configuration des agents de façon centralisée, livrez la connexion par ce canal géré
Les politiques n'arrivent jamais sur la machine	La machine n'a pas atteint la console depuis son installation, ou son identifiant a été révoqué ou renouvelé	Confirmez d'abord la route de sortie, selon Exigences réseau . Si l'identifiant a été refusé, émettez-en un pour la vague et relancez l'installation
La machine est connectée, mais ses agents se comportent comme si rien n'était là	Les points d'accès de modèles des agents sont bloqués au proxy, ou leurs réponses en flux sont mises en tampon	Exemptez ces points d'accès de la mise en tampon des réponses et de l'inspection des flux. C'est le trafic propre des agents, qui doit rester joignable comme avant
La machine n'apparaît jamais dans la console	La route de sortie est bloquée, ou l'inscription n'est jamais allée au bout	Lancez <code>openlatch doctor</code> sur la machine. Il nomme ce qui a échoué, ce qui distingue un problème de proxy d'une installation interrompue

En bref

- Une installation est vérifiée quand `openlatch doctor` ne signale aucun problème et que la console montre les agents de la machine, chacun avec un niveau de contrôle et un propriétaire déterminé.
- `openlatch status` indique si la machine est couverte. `openlatch doctor` le prouve, avec une cause et un remède pour chaque point relevé.

- `openlatch doctor --rescue` constitue une archive expurgée avec le diagnostic, la configuration et une fenêtre récente des enregistrements locaux. Ouvrez chaque demande de support avec elle.
- Un agent qui attend encore que son développeur confirme la connexion est connecté sur le papier mais pas dans les faits. Relancez ces cas avant de clore une vague.
- Trois vagues de deux semaines : environ dix machines de développeurs, environ cinquante par la voie gérée, puis le périmètre. Un identifiant client par vague, révoqué quand la vague se termine.

Exploiter

Utilisez cette page une fois les machines inscrites : pour garder le client à jour, travailler avec les enregistrements qu'il conserve sur chaque machine, et le désinstaller.

Mises à jour

Le client peut vérifier l'existence d'une nouvelle version et se mettre à jour. Le téléchargement est vérifié en intégrité avant que quoi que ce soit ne soit remplacé sur la machine : une empreinte SHA-512 est comparée à celle que le registre de paquets publie pour cette version, si bien qu'un téléchargement corrompu ou tronqué n'est jamais appliqué. Cette vérification détecte la corruption et la troncature ; elle ne protège pas contre un registre compromis. Laissez la mise à jour automatique active pour les installations qu'un développeur lance lui-même : elle garde la machine à jour sans que personne ne la planifie.

Un parc géré veut en général garder la main sur la version. Désactivez la mise à jour automatique et gardez la vérification active, pour qu'une machine continue de signaler qu'une version plus récente existe pendant qu'elle attend votre fenêtre de maintenance. Livrez ensuite la nouvelle version comme le client a été livré la première fois, par les outils qui gèrent déjà la machine.

Variable	Valeur
OPENLATCH_AUTO_UPDATE	<code>false</code> empêche le client de se mettre à jour lui-même. Définissez-la partout où vous voulez figer la version
OPENLATCH_UPDATE_CHECK	Laissez-la à <code>true</code> , pour que <code>openlatch status</code> signale toujours qu'une version plus récente existe



Note

Remplacer les fichiers sur disque ne fait pas passer à lui seul le client en cours d'exécution à la nouvelle version. La machine signale l'écart jusqu'au redémarrage du client : terminez la fenêtre avec `openlatch restart`, ou avec la déconnexion et la reconnexion que votre parc planifie déjà.

Enregistrements locaux

Le client conserve sa configuration et ses enregistrements locaux dans un seul répertoire : `~/ .openlatch` sur macOS et Linux, `%APPDATA%\openlatch` sur Windows. `OPENLATCH_DIR` le déplace. Un parc qui le déplace doit pointer ses propres scripts de collecte et de suppression vers la même valeur.

Les enregistrements sont écrits avec les identifiants masqués, et les chemins de fichiers sont masqués de la même façon. [Traitement des données et sécurité](#) liste ce qui est capturé, ce qui est masqué et combien de temps chaque type est conservé. Les enregistrements qui n'ont pas encore atteint la console restent dans ce répertoire et sont envoyés quand la console redevient joignable. Cette réserve est bornée : au-delà de sa limite de taille, 100 Mo par défaut, les enregistrements non envoyés les plus anciens sont abandonnés et le journal du daemon consigne un avertissement 0L-1204. Voir les réglages de l'outbox et du journal de secours dans [Configuration](#).

Ouvrir une demande de support

Lancez d'abord `openlatch doctor` sur la machine concernée. Il décrit ce qu'il trouve en termes simples. Constituez ensuite l'archive expurgée avec `openlatch doctor --rescue` : elle contient le diagnostic, la configuration et une fenêtre récente des enregistrements locaux. Envoyez cette archive, jamais le répertoire lui-même, avec le système d'exploitation et l'heure du problème avec son fuseau horaire.

Contactez votre référent technique OpenLatch par le canal convenu avec votre organisation. Envoyez les signalements de sécurité à security@openlatch.ai.

Désinstaller le client

Pour retirer le client d'une machine :

```
openlatch uninstall
```

Pour retirer le client et les enregistrements qu'il a conservés localement :

```
openlatch uninstall --purge
```


`openlatch uninstall` détache le client de chaque agent trouvé sur la machine et l'arrête. Chaque agent est laissé tel qu'il était avant l'installation, et les réglages qu'OpenLatch n'a pas écrits ne sont pas touchés. La configuration et les enregistrements locaux restent sur disque, sauf si vous demandez leur suppression.

`--purge` supprime aussi ce répertoire et l'identifiant conservé dans le magasin d'identifiants du système. Il refuse tout répertoire qui n'est pas celui du client, et il résout les liens symboliques avant de supprimer, pour qu'un lien ne puisse pas rediriger la suppression ailleurs. Un parc qui déplace le répertoire doit garder la même vérification dans son propre script de suppression.

Aucune des deux commandes ne supprime les fichiers du client lui-même. Ils ont été installés hors du répertoire d'état : une purge les laisse sur la machine, ainsi que la ligne PATH ajoutée par l'installation. Pour retirer complètement le client d'une machine, supprimez les deux fichiers placés par l'installation, `openlatch` et `openlatch-hook` dans `~/ .local/bin` sous macOS et Linux, ou `openlatch.exe` et `openlatch-hook.exe` dans `%LOCALAPPDATA%\openlatch\bin` sous Windows. Laissez le répertoire et tout ce qu'il contient d'autre : d'autres logiciels peuvent s'en servir. Sous Windows, le répertoire appartient au client : vous pouvez aussi le supprimer, ainsi que l'entrée PATH ajoutée par l'installation ; sous macOS et Linux, gardez la ligne PATH si d'autres outils dépendent de `~/ .local/bin`. Il n'existe pas de script de désinstallation : un parc qui veut que ce soit fait pour lui le fait par les outils qui ont installé le client.

La suppression se fait machine par machine. Rien n'est à défaire dans la console pour une machine désinstallée, et une machine qui revient est installée de la même façon que la première fois.

En bref

- Laissez la mise à jour automatique active pour les installations des développeurs. Figez-la avec `OPENLATCH_AUTO_UPDATE=false` sur un parc géré, et gardez la vérification active pour qu'une machine signale toujours ce qui lui manque.
- Une nouvelle version prend effet au redémarrage du client : terminez la fenêtre de maintenance par `openlatch restart` ou par une déconnexion et une reconnexion.
- Configuration et enregistrements locaux se trouvent dans un seul répertoire. Une demande de support s'ouvre avec l'archive expurgée produite par `openlatch doctor --rescue`.
- `openlatch uninstall` laisse chaque agent tel qu'il était. `--purge` supprime aussi le répertoire et l'identifiant conservé, et laisse les deux fichiers installés, qu'une suppression complète retire à part.

Traitement des données et sécurité

Remettez cette page à une revue de sécurité. Elle décrit ce qui quitte une machine qui exécute le client, ce qui y reste, combien de temps chaque enregistrement est conservé, et comment l'accès à la console est contrôlé.

OpenLatch contrôle un agent qui se comporte mal : un agent qui tourne en boucle ou dépense du budget pour un travail que personne n'a demandé, un agent qui a subi une injection par son contexte et agit en conséquence par ses outils, et un agent qui enfreint une politique tout en se comportant par ailleurs comme prévu. Ces défaillances produisent de vrais incidents, et c'est ce que la boucle de contrôle intercepte avant que l'action ne s'exécute. OpenLatch ne revendique aucune résistance face à un agent conçu pour le contourner.

Principes

Principe	Ce qu'il signifie
La décision est prise sur la machine	Chaque verdict est calculé sur la machine, avant que l'action ne s'exécute. La console n'est pas sur le chemin de décision
OpenLatch n'est sur aucun chemin réseau	Le client n'est pas une passerelle et n'inspecte aucun trafic de votre organisation. Vos agents atteignent leurs points d'accès de modèles exactement comme avant
Données minimales	Les valeurs sont masquées et les chaînes tronquées avant que quoi que ce soit ne quitte la machine, et des métadonnées sont envoyées partout où le contenu n'est pas nécessaire
Aucun identifiant ne quitte la machine	Les clés qu'utilisent vos agents restent sur la machine. Les mots de passe de proxy sont conservés par le système d'exploitation, jamais dans un fichier de configuration ni dans un journal

Inventaire des données

Lisez d'abord la deuxième colonne.

Donnée	Quitte la machine	Masquage et limites	Où elle est conservée	Conservation
Métadonnées d'action (nom d'outil, session, verdict, horodatage)	Oui	Aucun masquage. Les métadonnées ne portent pas de texte libre	Console	180 jours
Commandes shell et arguments d'outils	Oui	Identifiants masqués par des marqueurs étiquetés. Chaque chaîne plafonnée à 4 Ko	Console, et sous forme hachée dans les agrégats d'usage	180 jours ; agrégats 30 jours
Chemins de fichiers	Oui	Masqués de la même façon, et hachés avec une clé propre à l'organisation dans les agrégats d'usage	Console, et sous forme hachée dans les agrégats d'usage	180 jours ; agrégats 30 jours
Contenu des fichiers	Configurable	Filtré par défaut, ou empreinte seule. Capture en ligne plafonnée à 64 Ko	Console	180 jours
Prompts	Métadonnées seules	Occurrence, horodatage, session et longueur. Le texte du prompt n'est pas envoyé sauf si votre organisation l'active	Console	180 jours
Requêtes et réponses de modèles	Non	Le contenu n'est jamais envoyé. Le nombre de jetons, le modèle utilisé et la	Console	90 jours

Donnée	Quitte la machine	Masquage et limites	Où elle est conservée	Conservation
		répartition du cache sont enregistrés		
Identité de l'utilisateur	Oui	Utilisateur du système d'exploitation et e-mail git, pour attribuer une action à une personne	Console	180 jours, avec l'enregistrement
Identité de la machine	Oui	Système d'exploitation, architecture, adresses IP locale et publique, versions logicielles, un identifiant d'installation	Console	180 jours, avec l'enregistrement
Identifiant de compte de modèle	Oui	Un identifiant de compte, ou un marqueur indiquant qu'une clé partagée est utilisée. Jamais un jeton ni une clé	Console	180 jours, avec l'enregistrement
Politiques	Descendant uniquement	Envoyées à la machine, jamais remontées depuis elle	Sur la machine	Jusqu'à ce qu'un ensemble plus récent les remplace
Verdicts et anomalies	Oui	Aucun masquage	Console	180 jours

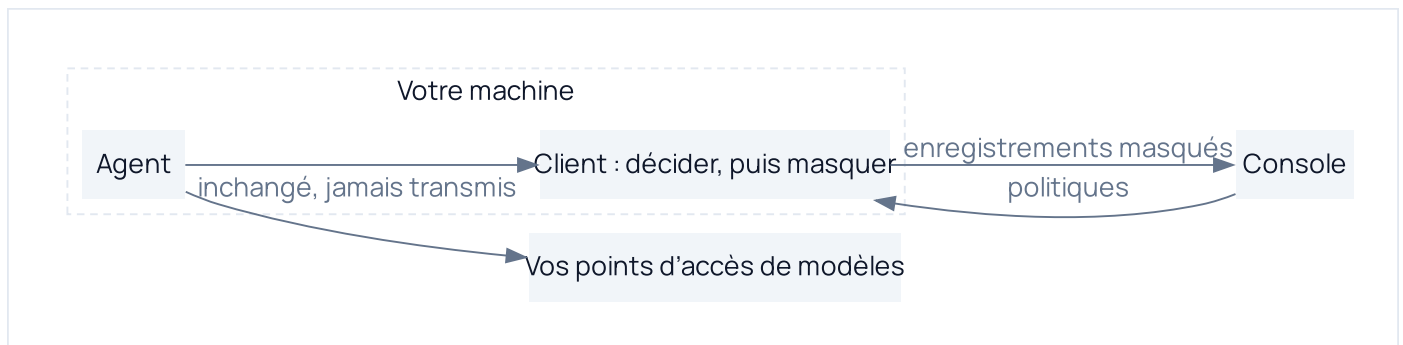


Figure 1. Flux de données et frontières de confiance : les enregistrements masqués remontent, les politiques descendent, et le trafic de modèles ne passe jamais par OpenLatch.

Masquage, et ce qui reste sur la machine

Le masquage s'exécute dans le client, sur la machine, avant qu'un enregistrement ne soit mis en file d'envoi. Il parcourt le contenu et remplace les valeurs qui ressemblent à des identifiants par un marqueur étiqueté qui garde le préfixe public reconnaissable et supprime le secret, sous la forme `[AWS_KEY:AKIA***]`. Seules les valeurs de chaîne sont masquées ; les noms de champs ne le sont jamais. Le masquage s'exécute avant la troncature : couper une chaîne au plafond de 4 Ko ne peut pas révéler la fin d'un secret partiellement masqué. Il s'exécute après le verdict : le texte expurgé ne change jamais les politiques qui correspondent. Votre organisation peut ajouter ses propres formes d'identifiants à la configuration du client, et elles s'appliquent en plus des formes intégrées.

Attention

Le client conserve en texte clair, sur la machine, les enregistrements locaux des actions, dans `~/ .openlatch` sur macOS et Linux ou `%APPDATA%\openlatch` sur Windows. Ils contiennent le texte des commandes et les chemins de fichiers masqués, et sont protégés par les permissions de fichiers de l'utilisateur et par le chiffrement de disque que votre parc applique. Les identifiants font exception : ils sont conservés par le magasin d'identifiants du système d'exploitation quand il en existe un, et sinon dans un fichier lisible par son seul propriétaire.

Conservation

Donnée	Durée	Configurable	Export
Enregistrements d'actions	180 jours	Non	—
Enregistrements de décisions et anomalies	180 jours	Non	CSV
Agrégats d'usage	30 jours	Non	Console
Archives hachées	90 jours	Non	Console
Enregistrements de coûts et de jetons	90 jours	Non	Console
Journal d'audit	90, 180, 365 ou 730 jours, ou une durée personnalisée	Oui, avec purge automatique	CSV ou JSON

Le journal d'audit est la seule durée que vous définissez vous-même, dans la console, avec purge automatique ; [Configurer l'organisation](#) montre où. Toutes les autres durées ci-dessus sont fixes. Les politiques présentes sur la machine n'ont pas de durée de conservation : elles sont remplacées quand le client récupère un ensemble plus récent, et supprimées quand le client est désinstallé avec `--purge`.

Hébergement

La console est un service hébergé aux États-Unis, dans la région US-East. Les données de chaque organisation sont séparées de celles de toutes les autres par la base de données elle-même : la sécurité au niveau des lignes est imposée sur chaque table de locataire, sans aucun rôle d'exécution qui la contourne. La liste des sous-traitants est disponible sur demande.

En transit, le client et la console communiquent en TLS validé avec le magasin de confiance du système d'exploitation, avec la possibilité d'approuver une autorité de certification supplémentaire là où votre parc termine TLS avec la sienne, et sans aucune option qui désactive la vérification. Au repos, les secrets stockés sont chiffrés.

Identité et accès

La console prend en charge l'authentification unique SAML 2.0 par organisation. Vous fournissez l'identifiant d'entité du fournisseur d'identité, son URL d'authentification, et un certificat de signature ou

une URL de métadonnées, puis vous définissez les domaines de messagerie autorisés. Un test effectue l'échange et indique le résultat avant la mise en service de la connexion. Microsoft Entra ID, Okta, PingOne et Active Directory Federation Services se connectent ainsi.

Deux portes contrôlent qui rejoint l'organisation. Les domaines de messagerie autorisés limitent les adresses que la console accepte, et les invitations contrôlent qui est admis. Une connexion SAML propre à l'organisation est la seule source d'identité qui admet un membre sans invitation : une connexion via un réseau social ne permet jamais de contourner le provisionnement.

Rôle	Ce qu'il permet
owner	Tout ce que permet admin, plus la propriété de l'organisation elle-même
admin	Réglages d'identité, politiques, identifiants du client et réglages de conservation
editor	Politiques et remplacements de propriétaire d'agent
viewer	Lecture des agents, des exécutions, des décisions et du journal d'audit. Les personnes invitées commencent ici

Un identifiant client hérite du rôle de la personne qui l'a émis. Un identifiant émis par un editor ne peut pas modifier un réglage d'identité, et révoquer l'identifiant ou rétrograder la personne ferment la même porte.

Le journal d'audit est en ajout seul et chaîné par hachage : une ligne supprimée ou modifiée rompt la chaîne. Chaque changement de politique écrit une ligne qui nomme l'auteur, avec l'état avant et après. Le journal s'exporte en CSV ou JSON, sa durée de conservation est configurable avec purge automatique, et l'export lui-même écrit un événement d'audit.

Protéger le client et ses politiques

Les mises à jour du client sont vérifiées en intégrité avant tout remplacement : l'empreinte SHA-512 du téléchargement doit correspondre à celle que le registre de paquets publie pour cette version. Cette vérification détecte la corruption et la troncature ; elle ne protège pas contre un registre compromis.

Les politiques présentes sur la machine sont vérifiées avant utilisation, puis à chaque chargement, pas seulement quand elles sont récupérées. Un ensemble qui échoue à la vérification est écarté et signalé, et la machine fonctionne sans politique jusqu'à l'arrivée d'un ensemble plus récent. Cet état est inscrit sur chaque enregistrement plutôt que passé sous silence, et il en va de même quand aucune politique n'a

jamais été récupérée : l'action est autorisée et l'enregistrement dit pourquoi. OpenLatch ne bloque jamais une action parce qu'une politique manque.

Les éléments secrets sont conservés par le magasin d'identifiants du système d'exploitation quand il en existe un, et sinon dans un fichier lisible par son seul propriétaire. Un identifiant passé en ligne de commande est refusé, car une ligne de commande est lisible par les autres processus de la machine.

Télémétrie produit et rapports de plantage

La télémétrie produit porte sur le logiciel OpenLatch, pas sur les agents de votre organisation. Elle contient des noms de commandes, des durées, l'agent, des codes d'erreur et des volumes, et jamais de texte de commande, de chemin de fichier, de contenu de fichier ni de prompt. Le consentement est demandé à l'installation, peut être modifié ensuite, et est désactivé par défaut en intégration continue. Les rapports de plantage contiennent un message d'échec nettoyé et rien d'autre, et sont envoyés sauf si vous les désactivez.

Les deux voyagent vers `app.openlatch.ai` avec le trafic dont le client a besoin pour fonctionner : une règle réseau ne peut pas les arrêter sans couper le client de la console. Les réglages sont donc sur le client : refusez la télémétrie à l'installation ou lancez `openlatch system telemetry disable`, et désactivez les rapports de plantage avec `OPENLATCH_CRASH_REPORTING=0`. Désactivés, le client se comporte exactement comme lorsqu'ils sont actifs. [Confidentialité et télémétrie](#) les explique plus en détail.

Signaler une vulnérabilité

Envoyez les signalements de sécurité à security@openlatch.ai. OpenLatch accuse réception d'un signalement sous deux jours ouvrés et traite la question avec la personne qui l'a signalée.

En bref

- Les métadonnées d'action, les commandes et chemins de fichiers masqués, les verdicts et les anomalies quittent la machine. Le texte des prompts reste sur la machine sauf si votre organisation l'active, et les requêtes et réponses de modèles ne sont jamais envoyées.
- Un seul répertoire contient la configuration et les enregistrements locaux, en texte clair, protégés par les permissions de fichiers de l'utilisateur et votre chiffrement de disque. Les identifiants sont conservés par le système d'exploitation, ou dans un fichier réservé à son propriétaire.

- Les enregistrements d'actions et de décisions sont conservés 180 jours, les agrégats d'usage 30 jours, les archives hachées et les enregistrements de coûts 90 jours. Le journal d'audit est la seule durée que vous définissez, avec purge automatique, et il s'exporte en CSV ou JSON.
- La console est hébergée aux États-Unis (US-East), et les données de chaque organisation sont isolées par la base de données. SAML 2.0 par organisation, quatre rôles d'owner à viewer, des identifiants client qui héritent du rôle de leur émetteur, et un journal d'audit en ajout seul, chaîné par hachage.
- Les mises à jour sont vérifiées en intégrité avant d'être installées, et les politiques de la machine sont vérifiées avant utilisation et à chaque chargement. Une machine sans politique utilisable autorise l'action et le dit sur l'enregistrement.

FAQ du déploiement

Utilisez cette page pour répondre aux questions que soulève un déploiement. Quand une autre page traite une question en entier, la réponse renvoie vers elle.

Installation

Question	Réponse
De quoi avez-vous besoin de notre part pour commencer ?	Trois choses : une organisation dans la console avec un administrateur qui peut inviter tous les autres, un accès HTTPS sortant sur 443 vers <code>app.openlatch.ai</code> , et le périmètre, c'est-à-dire les machines de la première vague. Les scripts d'installation apportent le client avec eux : une machine n'a besoin d'aucun environnement d'exécution. Prérequis en fait une liste de contrôle
Faut-il des droits d'administrateur sur la machine ?	Non. Le client s'installe et s'exécute au niveau de l'utilisateur, ne charge aucun pilote et n'ouvre aucun port entrant
Pouvons-nous le déployer avec nos outils de gestion existants ?	Oui. Comme le client s'exécute en contexte utilisateur, il se livre par un script au niveau de l'utilisateur plutôt que par un programme d'installation au niveau de l'appareil, avec l'identifiant transmis depuis votre coffre de secrets par l'environnement et jamais écrit dans un script, un paquet ou une ligne de commande. Créez une clé par vague de déploiement et révoquez-la quand la vague se termine. Voir Déployer le client
Quels agents couvre-t-il ?	L'installation détecte les agents présents sur une machine et raccorde chacun de ceux qu'elle trouve. Niveaux de contrôle liste les agents raccordés aujourd'hui, plan par plan. Chaque agent apparaît dans la console avec son niveau de contrôle, Enforced, Monitored ou Known, qui indique le contrôle dont vous disposez sur lui
Le développeur doit-il faire quelque chose ?	En général non. Certains agents demandent au développeur de confirmer la connexion une fois, au premier lancement après l'installation, et openlatch status montre quels agents attendent encore. Voir Vérifier et déployer

Réseau et proxy

Question	Réponse
Que doit atteindre OpenLatch, et ouvre-t-il quelque chose ?	Une seule destination, en sortie : <code>app.openlatch.ai</code> sur TCP 443 en HTTPS, autorisée par nom car il n'existe aucune plage d'adresses statique. Rien en entrée : le client n'écoute que sur l'interface de bouclage. Voir Exigences réseau
Notre seule sortie est un proxy authentifié. Qu'est-ce qui est pris en charge ?	HTTP et HTTPS CONNECT, SOCKS5, les proxys transparents, et PAC ou WPAD là où le résolveur du système d'exploitation l'évalue. Les authentifications Basic et Kerberos (Negotiate) sont prises en charge. NTLM ne l'est pas, par choix de conception ; Negotiate est la voie sur ces parcs. Les identifiants de proxy sont conservés dans le magasin d'identifiants du système d'exploitation
L'inspection TLS pose-t-elle problème ?	Non, à condition que l'autorité de certification d'interception soit approuvée. Installez-la dans le magasin de confiance du système d'exploitation, ou indiquez-la au client, auquel cas elle est fusionnée par-dessus le magasin du système. La vérification n'est jamais désactivée, et OpenLatch n'émet aucun certificat
Le trafic propre de nos agents change-t-il ?	Non. Vos agents atteignent leurs points d'accès de modèles exactement comme avant. Ces points d'accès doivent rester joignables et être exemptés de la mise en tampon des réponses et de l'inspection des flux
Que se passe-t-il quand une machine est hors ligne, ou la console injoignable ?	Les politiques restent en vigueur sur la machine, car la décision est prise localement. Les enregistrements sont conservés sur la machine et envoyés quand la console redevient joignable. Voir Architecture

Données et sécurité

Question	Réponse
Est-ce un proxy ou une passerelle devant nos agents ?	Ni l'un ni l'autre. OpenLatch s'exécute dans le cycle de vie de l'agent lui-même, et la décision est prise sur la machine avant que l'action ne s'exécute. Rien de chez nous n'est sur votre chemin réseau, et OpenLatch ne termine aucune connexion TLS
Qu'est-ce qui quitte la machine ? Nos prompts ? Notre code source ?	Les prompts sont capturés en métadonnées par défaut : qu'il y en a eu un, quand, dans quelle session et sa longueur. Le texte du prompt n'est pas envoyé sauf si votre organisation l'active, et les requêtes et réponses échangées avec un modèle ne sont jamais envoyées à OpenLatch. Ce qui part, ce sont les métadonnées d'action avec les identifiants et les chemins de fichiers masqués, plus les signaux d'identité nécessaires pour attribuer une action ; le contenu des fichiers est filtré par défaut. OpenLatch ne se connecte à aucun de vos dépôts. Voir Traitement des données et sécurité
Où la console est-elle hébergée ?	Aux États-Unis, dans la région US-East, avec les données de chaque organisation isolées de celles de toutes les autres. Les verdicts sont produits sur vos propres machines : le lieu de stockage des enregistrements n'affecte pas l'application. La liste des sous-traitants est disponible sur demande
Qui peut se connecter à la console, et qui voit quoi ?	L'authentification unique SAML 2.0 se configure par organisation, et l'accès dans la console suit quatre rôles. Chaque changement d'administration est écrit dans un journal d'audit en ajout seul que vous pouvez exporter. Voir Configurer l'organisation
Quelqu'un peut-il le désactiver sur sa propre machine ?	OpenLatch vise un agent qui se comporte mal : qui tourne en boucle, gaspille du budget, subit une injection et agit par ses outils, ou enfreint une politique tout en fonctionnant normalement par ailleurs. Le client s'installe au niveau de l'utilisateur : une personne qui contrôle sa machine peut le retirer. Ce qu'elle y perd, c'est la couverture, pas la discrétion : la couverture est signalée machine par machine, et une machine qui cesse de remonter apparaît comme un trou au lieu de passer pour couverte

Exploitation

Question	Réponse
Est-ce que cela ralentit l'agent ?	La décision est prise sur la machine, sans appel réseau ni modèle sur le chemin : une action couverte n'attend aucun aller-retour. OpenLatch ne publie aucun chiffre de latence. Ce qu'il affirme, c'est que chaque action couverte atteint le point de décision des politiques, et que le verdict produit par une politique explicite est celui qui est appliqué
Comment confirmer qu'une machine est vraiment connectée ?	<code>openlatch status</code> indique si la machine est connectée, si des politiques sont présentes et si les agents sont raccordés, et <code>openlatch doctor</code> prouve chacun de ces points. Les agents de la machine apparaissent ensuite dans le registre des agents de la console avec un propriétaire. Voir Vérifier et déployer
Comment fonctionnent les mises à jour, et peut-on les figer ?	Le client peut se mettre à jour, et chaque téléchargement est vérifié en intégrité avant tout remplacement. Laissez cela actif pour les installations des développeurs. Un parc géré le désactive, garde la vérification active, et livre la nouvelle version dans sa propre fenêtre de maintenance. Voir Exploiter
Comment faire entrer les notifications et les enregistrements dans nos outils ?	Les notifications partent par e-mail, vers un canal de messagerie comme Slack ou Microsoft Teams, ou vers un webhook qui vous appartient, et le journal d'audit s'exporte en CSV ou JSON selon la conservation que vous définissez. Voir Configurer les canaux de notification
Comment le désinstaller ?	<code>openlatch uninstall</code> détache le client de chaque agent de la machine et laisse chacun tel qu'il était. Ajouter <code>--purge</code> supprime aussi le répertoire local et l'identifiant conservé. Voir Exploiter
Comment ouvrir une demande de support ?	Contactez votre référent technique OpenLatch par le canal convenu avec votre organisation. Lancez <code>openlatch doctor --rescue</code> sur la machine concernée et joignez l'archive expurgée qu'il produit, avec le système d'exploitation et l'heure du problème avec son fuseau horaire. Les signalements de sécurité vont à security@openlatch.ai

Termes et flux sortants

Utilisez cette page pour retrouver un terme de la section Déployer, ou pour copier la règle de sortie dans une demande de changement.

Termes et acronymes

Terme	Signification
Agent Autonomy Zone	Le périmètre opérationnel dans lequel un agent travaille, exprimé en politiques d'intention selon cinq dimensions : Qualité, Économie, Sécurité, Sûreté et Conformité. Voir Autonomy Zone
Agent Intent Policy (AIP)	La forme sous laquelle l'intention de votre organisation est rédigée dans la console, et compilée en règles déterministes qui décident d'une action
Verdicts	Allow, Ask, Block ou Optimize, dans cet ordre. Allow laisse l'action se poursuivre. Ask la laisse se poursuivre et la signale pour une revue asynchrone. Block l'arrête avant qu'elle ne s'exécute. Optimize modifie le chemin : en Enforce, un levier agit, et en Monitor, il mesure ce qu'il aurait fait. Voir Verdicts
Client	Le logiciel OpenLatch installé une fois sur chaque machine qui exécute un agent. Il décide d'une action couverte, sur la machine, avant qu'elle ne s'exécute
Console	L'application web OpenLatch à <code>app.openlatch.ai</code> , où vous configurez l'organisation, rédigez les politiques et voyez ce qu'ont fait vos agents
Niveau de contrôle	Le contrôle dont OpenLatch dispose sur un agent. Enforced : le verdict atteint l'agent et il est respecté. Monitored : les événements circulent, mais aucun verdict ne peut être délivré. Known : l'agent est identifié et classé, avec les limites du contrôle énoncées. Voir Niveaux de contrôle
IdP	Fournisseur d'identité : l'annuaire et le service d'authentification de votre organisation
MDM	Gestion des appareils mobiles : les outils qui livrent logiciels et réglages aux machines gérées
PAC / WPAD	Configuration automatique de proxy et Web Proxy Auto-Discovery : un script qui indique à un client quel proxy utiliser, et le mécanisme qui le localise
SAML2.0	Security Assertion Markup Language, version 2.0 : le protocole d'authentification unique que la console prend en charge par organisation
SNI	Server Name Indication : l'extension TLS qui porte en clair le nom de l'hôte de destination, et que lit une autorisation par nom

Terme	Signification
Inspection TLS	Un proxy qui termine et réémet TLS pour inspecter le trafic. Prise en charge quand l'autorité de certification d'interception est approuvée
Vague	Une étape d'un déploiement. La section Déployer compte trois vagues de taille croissante

Flux sortants requis

Remettez ce tableau à la personne qui gère vos flux sortants. Une machine qui exécute le client a besoin d'exactly une destination d'OpenLatch, et la télémétrie produit et les rapports de plantage facultatifs y transitent aussi. La deuxième ligne n'est nécessaire que pendant l'installation ou la mise à jour du client. La troisième concerne les navigateurs qui se connectent à la console ; aucune machine qui exécute le client ne la contacte.

Destination	Port	Sens	Protocole	Comment l'autoriser
<code>app.openlatch.ai</code>	TCP 443	Sortant	HTTPS	Par nom. Il n'existe aucune plage d'adresses statique. Cette entrée porte aussi la télémétrie produit et les rapports de plantage facultatifs
Votre registre de paquets : <code>registry.npmjs.org</code> , sauf si vous pointez le client sur un miroir	TCP 443	Sortant	HTTPS	Par nom, ou utilisez votre miroir interne. Nécessaire pour une installation ou une mise à jour, pas pour fonctionner
<code>challenges.cloudflare.com</code> , depuis le navigateur de toute personne qui se connecte à la console	TCP 443	Sortant	HTTPS	Par nom, pour les navigateurs uniquement. Il sert la vérification que l'étape e-mail effectuée à la connexion

Vos agents atteignent leurs points d'accès de modèles exactement comme avant. Ces points d'accès doivent rester joignables, et exemptés de la mise en tampon des réponses et de l'inspection des flux.

Aucun port entrant n'est requis, ni sur la machine ni ailleurs. OpenLatch n'émet aucun certificat et n'en exige aucun. [Exigences réseau](#) traite le sujet en entier, dont les proxys, l'authentification et l'inspection TLS.

Référence des AIP

Utilisez cette page pour lire ou relire une Agent Intent Policy (AIP) : de quoi elle est faite, quelle dimension de l'Autonomy Zone elle couvre, dans quel mode elle s'exécute et quel verdict elle rend pour une action couverte.

Une politique d'intention énonce ce que veut dire votre organisation, plutôt que chaque surface qu'un agent pourrait utiliser. OpenLatch écrit chaque politique d'intention sous forme d'AIP et la compile en règles déterministes que le client applique avant l'exécution de l'action.

Atomes d'intention

Une AIP est un ensemble d'un ou plusieurs **atomes d'intention**. Une clause de votre document source devient un atome, et chaque atome a les mêmes champs fixes.

Champ	Contenu
who	Les agents auxquels l'atome s'applique. Quand la source ne le dit pas, l'atome indique que c'est une déduction
does what	L'action, comme un déploiement ou la suppression d'un fichier
to what	La cible de l'action, comme un espace de noms de production
only if	Les conditions dans lesquelles l'action est acceptable
unless	Les exceptions à l'atome
class	Si l'atome peut être appliqué : <code>enforceable</code> , <code>monitor-only</code> ou <code>out-of-scope</code>

À partir de ces champs, OpenLatch génère une **phrase** et un ensemble de **cas de test**. Vous ne modifiez jamais les règles compilées.

Vous voyez	À quoi cela sert
La phrase	Ce que vous ratifiez, par un seul Oui ou Non
La source	La clause de votre document dont vient l'atome, citée pour que vous vérifiiez qu'elle a été bien lue
Les cas de test	Des situations concrètes avec le verdict que chacune obtiendrait, affichées en lecture seule à côté de la phrase pour éclairer votre réponse

Un Non renvoie l'atome en correction : vous dites ce que vous vouliez dire, les champs changent, et l'atome revient avec une nouvelle phrase et de nouveaux cas pour une nouvelle réponse. Modifier un atome annule toujours la ratification donnée à sa version précédente.

La class d'un atome est une propriété de l'atome, pas un réglage. Un atome `monitor-only` décrit quelque chose qui peut être observé mais pas arrêté avant qu'il se produise ; il reste donc en Monitor même quand sa politique est en Enforce.

Dimensions

Chaque atome appartient à l'une des cinq dimensions de l'Autonomy Zone. Une politique couvre l'union des dimensions de ses atomes.

Dimension	La question à laquelle elle répond	Exemples de politiques d'intention
Qualité	Le travail est-il au niveau attendu ?	Le code est testé avant sa livraison ; le processus imposé est suivi
Économie	Le coût est-il justifié ?	Pas de boucles incontrôlées ni de nouvelles tentatives inutiles ; le modèle correspond à la tâche
Sécurité	Peut-il y toucher ?	Aucun identifiant ne sort du périmètre de l'entreprise ; seulement des outils et serveurs approuvés
Sûreté	La conséquence est-elle acceptable ?	Pas d'accès à la production sans approbation ; pas d'opération irréversible ou destructrice
Conformité	Est-ce légal et conforme aux règles ?	Politiques et normes internes ; obligations contractuelles

La mission n'est pas une dimension : chaque agent apporte sa propre mission, et la zone apporte les règles qu'il doit suivre.

Modes

Une politique s'exécute dans l'un de trois modes. Le mode appartient à la politique, jamais à un atome, et se règle dans chaque Autonomy Zone qui contient la politique.

Mode	Ce qui se passe en cas de correspondance
Disabled	Rien. La politique et son historique sont conservés, mais elle n'est pas évaluée
Monitor	La correspondance est enregistrée comme anomalie et signalée ; l'action se poursuit
Enforce	Le verdict est appliqué avant l'exécution de l'action, partout où un adaptateur couvre cette action, et l'enregistrement indique s'il a réellement été appliqué

Quand les zones d'un agent contiennent plusieurs fois la même politique, elle s'exécute une seule fois, dans le mode le plus strict : Enforce avant Monitor avant Disabled.

Les politiques gagnent le mode Enforce sur preuves : exécutez une politique en Monitor, examinez ses anomalies, et passez-la en Enforce une fois qu'elle est juste. OpenLatch n'élargit jamais de lui-même l'autorité d'un agent.

Verdicts

Chaque action couverte qui correspond à une politique en Enforce reçoit l'un de quatre verdicts avant son exécution. Un verdict est déterministe, car une politique explicite l'a produit.

Verdict	Ce qui arrive à l'action
Allow	L'action se poursuit sans changement. En Monitor, elle peut être autorisée et signalée
Ask	L'action est autorisée et signalée pour revue par défaut
Block	L'action est empêchée avant son exécution. L'agent en reçoit la raison et peut essayer autre chose ; la session continue
Optimize	L'action prend un autre chemin, par l'un des leviers ci-dessous

Ask

Ask autorise l'action et la signale pour une revue asynchrone par défaut. La décision de revue, avec la portée choisie par le relecteur, parvient au client dans le prochain bundle de politiques ; l'action similaire suivante est donc décidée sur la machine, sans personne.

Ask ne retient une action et n'attend une approbation que sur un opt-in par atome pour une action irréversible, comme la suppression d'une base de données de production.

Codex CLI n'a pas de demande de confirmation native. Sur Codex CLI, Ask apparaît pour l'agent comme un message portant la raison de la règle, et l'appel se poursuit.

Ask empêche la récurrence, pas la première occurrence. Écrivez en Block tout ce qui ne doit jamais se produire sans revue, ou marquez l'action comme irréversible sur son atome.

Leviers Optimize

Optimize change le chemin d'une action au lieu de l'autoriser ou de la bloquer. Le mode décide de ce que fait un levier : en Enforce un levier agit, et en Monitor un levier se contente de mesurer ce qu'il aurait fait.

Levier	Ce qu'il change	Statut
loop_stop	Arrête un agent qui répète le même appel, et lui dit pourquoi	Agit en Enforce
steer	Refuse l'action et indique à l'agent quoi faire à la place, comme pousser une branche plutôt que la branche principale	Agit en Enforce
narrow_output	Renvoie à l'agent un extrait filtré d'une longue sortie de commande, et garde la sortie complète sur le disque	Agit en Enforce, sur Claude Code
effort_clamp	Réduit l'effort de raisonnement d'une requête au modèle	Mesure en Monitor et en Enforce ; agit en Enforce seulement là où l'organisation cliente l'active
context_edit	Resserre les réglages d'effacement du contexte d'une requête au modèle	Mesure en Monitor et en Enforce ; agit en Enforce seulement là où l'organisation cliente l'active
prefix_guard	Garde stable le préfixe de prompt mis en cache d'une requête au modèle	Mesure en Monitor et en Enforce ; agit en Enforce seulement là où l'organisation cliente l'active
substitute	Remplace une commande par un équivalent approuvé par votre organisation	N'est pas livré : il n'a pas d'actionneur et ne fait que mesurer en Monitor

En Enforce, loop_stop, steer et narrow_output agissent (narrow_output sur Claude Code). effort_clamp, context_edit et prefix_guard ne font que mesurer, même en Enforce, sauf si l'organisation cliente l'active. substitute n'est pas livré : il ne fait donc que mesurer en Monitor. En Monitor, chaque levier ne fait que mesurer.

L'enregistrement de preuve indique lequel s'applique : appliqué en Enforce, mesuré en Monitor.

Ce que couvre un verdict

Chaque action couverte atteint le point de décision des politiques, et son verdict est appliqué. Une action hors d'une surface couverte, comme un script que l'agent a écrit plus tôt et exécute plus tard, n'est pas

décidée par le client. Le niveau de contrôle de chaque agent indique la couverture qui s'applique à lui.

Configuration du client

Utilisez cette page pour changer le comportement du client sur une machine : où il conserve son état, comment il joint la plateforme, et lesquelles de ses parties s'exécutent.

`openlatch init` écrit la configuration pour vous. Ne la modifiez à la main que pour un paramètre qu'aucune commande ne couvre, puis exécutez `openlatch restart` pour l'appliquer : le daemon ne lit sa configuration qu'une fois, au démarrage.

D'où viennent les paramètres

Le client lit chaque paramètre dans la première source qui le définit, dans cet ordre :

1. Une option sur la ligne de commande.
2. Une variable d'environnement (`OPENLATCH_*`).
3. Le fichier de configuration, `config.toml` dans le répertoire d'état : `~/ .openlatch` sous macOS et Linux, `%APPDATA%\openlatch` sous Windows.
4. La valeur par défaut intégrée.

La route du proxy est la seule exception : elle se résout clé par clé et, après le fichier de configuration, elle tient aussi compte des variables standard de la famille `https_proxy`, puis des réglages de proxy du système d'exploitation, avant de passer en direct.

Le daemon supervisé n'hérite pas des variables de votre shell. Tout ce dont il a besoin, comme l'adresse de la plateforme, `openlatch init` l'écrit dans `config.toml` pour que le daemon l'utilise aussi.

Variables d'environnement

État et connexion

Variable	Ce qu'elle définit	Par défaut
OPENLATCH_DIR	Le répertoire d'état : configuration, identifiant, journaux et outbox	~/ .openlatch sous macOS et Linux, %APPDATA%\openlatch sous Windows
OPENLATCH_PORT	Le port du daemon. Définie, openlatch init utilise exactement ce port ; non définie, il cherche un port libre de 7443 à 7543	7443
OPENLATCH_API_URL	L'adresse de la plateforme à laquelle le client envoie les événements et où il récupère les politiques	https://app.openlatch.ai
OPENLATCH_API_KEY	L'identifiant d' enrôlement. openlatch init l'enregistre dans le trousseau du système d'exploitation, ou dans un fichier chiffré quand aucun trousseau n'est utilisable	—
OPENLATCH_SKIP_KEYRING	Considère le trousseau comme indisponible, pour que la variable d'environnement l'emporte	non définie
OPENLATCH_LOG	Niveau de journalisation : error, warn, info, debug ou trace	info

Proxy

Variable	Ce qu'elle définit	Par défaut
OPENLATCH_PROXY	L'URL du proxy pour chaque requête sortante. C'est le seul paramètre qui peut contenir un identifiant ; <code>openlatch init</code> déplace le mot de passe dans le trousseau au lieu de l'écrire sur le disque	non définie
OPENLATCH_NO_PROXY	Destinations supplémentaires qui contournent le proxy. Les adresses de bouclage le contournent toujours	non définie
OPENLATCH_PROXY_MODE	<code>auto</code> , <code>manual</code> ou <code>direct</code>	<code>auto</code>
OPENLATCH_PROXY_AUTH	<code>auto</code> , <code>none</code> , <code>basic</code> ou <code>negotiate</code> (Kerberos). NTLM n'est pas pris en charge	<code>auto</code>
OPENLATCH_PROXY_PAC_URL	Une URL explicite de configuration automatique du proxy, utilisée sur macOS et Windows. Refusée sur Linux avec <code>OL-1225</code>	non définie
OPENLATCH_PROXY_SPN	Le nom de service Kerberos du proxy	<code>HTTP/<proxy-host></code>
OPENLATCH_CA_BUNDLE	Un bundle PEM ajouté aux certificats approuvés du système d'exploitation, pour un proxy qui inspecte le trafic TLS	non définie

Parties du client

Variable	Ce qu'elle définit	Par défaut
OPENLATCH_POLICY_ENABLED	L'évaluation locale des politiques. false désactive complètement l'application sur cette machine ; ce n'est pas un mode Monitor (OL-1215)	true
OPENLATCH_MODEL_RELAY_ENABLED	Le relais de modèle, l'écouteur local qui enregistre les appels de modèle de chaque agent. Le daemon n'y dirige un agent qu'une fois qu'une requête de test prouve qu'il joint le fournisseur	true
OPENLATCH_MODEL_RELAY_PORT	Le port du relais de modèle	7600
OPENLATCH_INVENTORY_ENABLED	Le suivi des fichiers de configuration des agents (le plan de configuration)	true
OPENLATCH_CLINE_PLUGIN_DELIVERY	La livraison du plugin OpenLatch à Cline.app et à l'extension Cline de VS Code. Équivaut à plugin_delivery dans [cline]	true

Envoi vers la plateforme

Variable	Ce qu'elle définit	Par défaut
OPENLATCH_CLOUD_BATCH_MAX_EVENTS	Nombre d'événements envoyés par lot, de 1 à 100	50
OPENLATCH_CLOUD_BATCH_MAX_WAIT_MS	Combien de temps le premier événement d'un lot attend avant l'envoi du lot, en millisecondes	5000
OPENLATCH_CLOUD_OUTBOX_MAX_BYTES	Le volume maximal d'événements non envoyés que garde l'outbox, en octets, avant l'abandon des plus anciens. 0 supprime la limite	104857600
OPENLATCH_CLOUD_FALLBACK_MAX_BYTES	Le volume maximal d'événements non rejoués que garde le journal de secours, en octets, avant que les plus anciens soient ignorés. Il limite cette fenêtre, pas le fichier : pendant une longue coupure, le fichier sur disque continue de grossir jusqu'à ce que le daemon le rejoue. Lu par le daemon uniquement ; le hook garde sa propre limite de 50 Mo. 0 supprime la limite	52428800

L'envoi des événements à la plateforme ne peut pas être désactivé. Pour les envoyer ailleurs, modifiez OPENLATCH_API_URL.

Mises à jour et télémétrie

Variable	Ce qu'elle définit	Par défaut
OPENLATCH_AUTO_UPDATE	Les mises à jour automatiques : le daemon recherche régulièrement une nouvelle version et l'installe dès qu'aucun hook ne s'exécute. true ou 1 les active ; toute autre valeur les désactive. Équivaut à auto_update dans [update]	true
OPENLATCH_UPDATE_CHECK	La vérification de version au démarrage. false ou 0 la désactive. Équivaut à check dans [update]	true
OPENLATCH_NPM_REGISTRY	Le registre depuis lequel openlatch update télécharge	https://registry.npmjs.org
OPENLATCH_TELEMETRY_DISABLED	1 désactive la télémétrie d'usage	non définie
DO_NOT_TRACK	1 désactive la télémétrie d'usage, comme dans les autres outils qui suivent cette convention	non définie
OPENLATCH_CRASH_REPORTING	0, false, no ou off arrête les rapports de plantage, indépendamment de la télémétrie d'usage	activée

Les mises à jour automatiques ne s'exécutent jamais sur les runners de CI.

Fichier de configuration

`config.toml` dans le répertoire d'état contient les paramètres qu'écrivent les commandes, regroupés en sections. Les sections que vous lirez ou modifierez le plus probablement :

Section	Clé	Ce qu'elle définit
<code>[policy]</code>	<code>enabled</code>	L'évaluation locale des politiques, comme <code>OPENLATCH_POLICY_ENABLED</code>
<code>[model_relay]</code>	<code>enabled</code>	Le relais de modèle, comme <code>OPENLATCH_MODEL_RELAY_ENABLED</code> . <code>openlatch system model-relay enable</code> et <code>disable</code> l'écrivent
<code>[cline]</code>	<code>plugin_delivery</code>	La livraison du plugin OpenLatch aux hôtes Cline
<code>[update]</code>	<code>auto_update</code> <code>check</code>	Les mises à jour automatiques et la vérification de version au démarrage
<code>[proxy]</code>	voir ci-dessous	La route empruntée par chaque requête sortante

La section `[proxy]`

`openlatch init` trouve la route seul quand il le peut et l'écrit ici. `openlatch system proxy set`, `discover` et `clear` la modifient ; préférez-les à la modification du fichier.

```
[proxy]
mode = "auto"
url = "http://proxy.corp:8080"
auth = "auto"
no_proxy = ".corp.example,10.0.0.0/8"
ca_bundle = "/etc/ssl/corp-root.pem"
allow_direct = true
```


Clé	Ce qu'elle définit
<code>mode</code>	<code>auto</code> cherche une route qui fonctionne, <code>manual</code> utilise exactement ce qui est défini et n'est jamais remplacé, <code>direct</code> n'utilise jamais de proxy
<code>url</code>	Le proxy, en <code>http://</code> , <code>https://</code> , <code>socks5://</code> ou <code>socks5h://</code> . Elle ne contient jamais de nom d'utilisateur ni de mot de passe
<code>username</code>	Le compte avec lequel s'authentifier. Le mot de passe est dans le trousseau, jamais dans ce fichier
<code>auth</code>	<code>auto</code> choisit un schéma au démarrage du daemon ; <code>basic</code> , <code>negotiate</code> et <code>none</code> sont explicites
<code>no_proxy</code>	Destinations supplémentaires qui contournent le proxy, comme <code>example.com</code> , <code>.example.com</code> ou <code>10.0.0.0/8</code>
<code>ca_bundle</code>	Un bundle PEM ajouté aux certificats approuvés, jamais à leur place
<code>allow_direct</code>	<code>false</code> refuse une connexion directe quand aucun proxy n'est utilisable (0L-1227)
<code>spn</code>	Le nom de service Kerberos du proxy
<code>http1_only</code>	Force HTTP/1.1, pour un proxy d'inspection qui gère mal HTTP/2
<code>pac_url</code>	Une URL explicite de configuration automatique du proxy. Refusée sur Linux (0L-1225)

Exécutez `openlatch system proxy test` après chaque modification pour vérifier la route.

Ce que le client écrit dans les réglages des agents

Tant que le relais de modèle est activé, le daemon dirige les appels de modèle de chaque agent détecté vers lui, et retire cette liaison quand vous arrêtez le daemon, désactivez le relais ou désinstallez. Une valeur que vous avez définie vous-même n'est jamais écrasée : `openlatch doctor` la signale à la place (0L-RELAY-SETTINGS).

Agent	Où va la liaison
Claude Code	Des clés d'environnement dans ses réglages : l'adresse du relais, un identifiant d'installation et des entrées de bouclage dans <code>NO_PROXY</code>
Codex CLI	Une entrée de fournisseur <code>openlatch</code> dans <code>~/.codex/config.toml</code>
Cursor	L'adresse du relais dans les réglages utilisateur de l'IDE, et un wrapper de shell pour la CLI Cursor qu'un nouveau terminal prend en compte
Cline	Un point de terminaison local du relais par fournisseur configuré, dans les fichiers de réglages de Cline

Relier Claude Code au relais de modèle définit son `ANTHROPIC_BASE_URL`, ce qui désactive Claude Code Remote Control tant que le relais est activé. Sur une machine derrière un proxy, exportez aussi `NO_PROXY=127.0.0.1,localhost` dans le shell qui lance Codex CLI, pour que ses appels de modèle atteignent le relais local plutôt que le proxy de l'entreprise.

Exécutez `openlatch system model-relay status` pour voir quels agents ont leurs appels de modèle qui passent par le relais.

Glossaire

Utilisez cette page quand un terme de la console, de la sortie du client ou de cette documentation demande un sens précis. Les noms de produit, les verdicts et les modes gardent leur forme anglaise dans toutes les langues.

Autonomy Zone et politiques

Terme	Sens
Agent Autonomy Zone	Le cadre d'action qu'une organisation définit pour ses agents IA : ce qu'un agent peut faire, à quelles conditions, et avec quelles conséquences acceptées. Forme courte : Autonomy Zone
Dimension	L'un des cinq côtés de l'Autonomy Zone : Qualité, Économie, Sécurité, Sûreté et Conformité
Zone effective	La zone unique à laquelle se rattache un agent. Une unité de votre organisation sans zone propre hérite de la plus proche au-dessus d'elle
Politique d'intention	Une règle qui énonce ce que veut dire votre organisation, plutôt que chaque surface qu'un agent pourrait utiliser
Agent Intent Policy (AIP)	La forme que prend une politique d'intention dans OpenLatch : un ensemble d'atomes d'intention, compilé en règles déterministes. Voir la référence des AIP
Atome d'intention	Une clause d'une politique, tenue dans des champs fixes (who, does what, to what, only if, unless) à partir desquels une phrase et des cas de test sont générés
Ratification	Le Oui ou le Non du responsable de la politique sur la phrase d'un atome, avec ses cas de test affichés à côté
Mode	La façon dont une politique s'exécute dans une zone : Disabled, Monitor ou Enforce
Disabled	Un mode dans lequel la politique est conservée mais pas évaluée
Monitor	Un mode dans lequel une correspondance est enregistrée et signalée, et l'action se poursuit
Enforce	Un mode dans lequel le verdict est appliqué avant l'exécution de l'action, partout où un adaptateur la couvre

Décisions

Terme	Sens
Verdict	La décision qu'une politique rend pour une action couverte, avant son exécution : Allow, Ask, Block ou Optimize
Allow	L'action se poursuit sans changement
Ask	L'action est autorisée et signalée pour revue par défaut. Elle n'est retenue pour approbation que sur un opt-in par atome pour une action irréversible
Block	L'action est empêchée avant son exécution, et l'agent en reçoit la raison
Optimize	L'action prend un autre chemin, par un levier Optimize
Levier Optimize	Une façon dont Optimize change une action. En Enforce, un levier agit là où il est livré ; en Monitor, il se contente de mesurer ce qu'il aurait fait
Anomalie	Une action d'agent qui enfreint une politique d'intention de l'une des dimensions de la zone
Déterministe	Se dit d'un verdict produit par une politique explicite : la même action dans le même contexte obtient toujours le même verdict

Agents et couverture

Terme	Sens
Agent approuvé	Un agent que votre organisation a approuvé et relié à OpenLatch
Niveau de contrôle	Le degré de contrôle d'OpenLatch sur un agent : Enforced, Monitored ou Known
Enforced	Un niveau de contrôle : un adaptateur OpenLatch se trouve dans le chemin d'exécution de l'agent et applique les verdicts
Monitored	Un niveau de contrôle : les actions de l'agent sont visibles et ses anomalies enregistrées, mais les verdicts ne sont pas appliqués
Known	Un niveau de contrôle : l'agent est identifié et classé, et les limites du contrôle sont indiquées
Plan des commandes	Les actions qu'un agent mène par ses outils : commandes, modifications de fichiers, appels d'outils et de serveurs
Plan des requêtes	Les requêtes qu'un agent envoie à son fournisseur de modèle
Graphe d'exécution	La trace de ce qu'un agent a fait pendant une session : appels de modèle, appels d'outils, commandes, modifications de fichiers, accès aux données et coût, dans l'ordre
Enregistrement de preuve	Ce qu'OpenLatch conserve pour chaque décision : l'action, la politique, le mode, le verdict et s'il a été appliqué

Le client

Terme	Sens
Client	Le programme OpenLatch installé sur chaque machine où tournent des agents
Daemon	Le processus d'arrière-plan du client, qui reçoit chaque événement d'agent et le décide
Hook	Le point d'entrée qu'un agent appelle avant et après chaque action, et qui transmet l'action au daemon
Bundle de politiques	Le fichier de règles à intégrité contrôlée que le client récupère auprès de la plateforme et conserve sur le disque, pour continuer à décider réseau débranché
Relais de modèle	Un écouteur local vers lequel le client dirige les appels de modèle de chaque agent, pour que le plan des requêtes soit enregistré
Outbox	La file locale où le client conserve les événements jusqu'à ce que la plateforme les ait reçus
Superviseur	Le service du système d'exploitation (launchd, systemd ou le Planificateur de tâches) qui démarre le daemon à l'ouverture de session et le redémarre
Plan de configuration	Les fichiers de configuration des agents dont le client suit les changements
Code d'erreur	Un code comme 0L-1002 que le client affiche avec un problème. Chacun a sa propre page, listée dans la référence de la CLI

La plateforme

Terme	Sens
Plateforme	Le service OpenLatch où vous définissez les zones et les politiques et examinez ce qu'ont fait les agents, accessible dans la console sur <code>app.openlatch.ai</code>
Console	L'interface web de la plateforme
Organisation	Le compte de votre entreprise sur la plateforme, qui contient ses unités, zones, politiques et agents
Clé d'API	Un identifiant qui permet au client ou à un script d'agir pour votre organisation sur la plateforme

Référence de la CLI

Toutes les commandes et sous-commandes du client OpenLatch.

openlatch init

`openlatch setup`

Installe les hooks des agents trouvés sur cette machine, démarre le daemon et vérifie qu'il répond. Peut être relancée sans risque : elle libère d'abord les ports du client, puis confirme que le daemon qu'elle a démarré est bien celui qui répond.

openlatch status

Affiche l'état du daemon, sa durée de fonctionnement et le nombre d'événements, l'état de chaque partie du client et la ligne de licence : son état, les hôtes, l'usage raisonnable et la fin du contrat.

openlatch start

Démarre le daemon. Refuse avec OL-1501 quand un daemon tourne déjà sur cette machine.

openlatch stop

Arrête le daemon. Retire aussi la liaison des agents au relais de modèle, et le signale.

openlatch restart

Arrête le daemon et le redémarre, par exemple pour appliquer un changement de configuration.

openlatch logs

Affiche le journal des événements du client.

openlatch doctor

Diagnostic la configuration et la connectivité du client, et donne la cause et la solution de chaque problème trouvé. `--fix` applique les réparations possibles, `--restore` les annule, et `--rescue` écrit un fichier de diagnostic pour le support.

openlatch uninstall

Retire les hooks des agents et arrête le daemon. `--agent` la limite aux agents que vous nommez.

openlatch docs

Ouvre la documentation dans votre navigateur.

openlatch system

Regroupe les commandes qui inspectent ou pilotent une partie du client : `aip`, `auth`, `hooks`, `model-relay`, `proxy`, `supervision`, `telemetry` et `inventory`. Chaque solution que donne `openlatch doctor` nomme la commande `system` exacte à exécuter.

openlatch system aip

Récupère et inspecte le bundle de politiques présent sur cette machine, qui contient les règles de l'Autonomy Zone appliquées par le client.

openlatch system aip refresh

`openlatch system aip update`

Fait récupérer le bundle de politiques par le daemon en cours, tout de suite, et attend le résultat jusqu'à 15 secondes : modifié, inchangé, en attente, refusé ou hors ligne, avec les caractéristiques du bundle en service. Un résultat en attente sort avec le code 7 ; un refus ou un daemon hors ligne sort avec le code 1.

openlatch system auth

Regroupe les commandes qui connectent le client à la plateforme.

openlatch system auth login

Connecte le client à la plateforme via votre navigateur et enregistre l'identifiant dans le trousseau du système d'exploitation.

openlatch system auth logout

Déconnecte le client et supprime l'identifiant enregistré.

openlatch system auth status

Indique si le client est connecté, et d'où vient son identifiant.

openlatch system auth switch

Déplace cette machine vers une autre de vos organisations.

openlatch system hooks

Regroupe les commandes qui gèrent les hooks des agents. `system hooks install` équivaut à `openlatch init`.

openlatch system hooks install

Installe les hooks des agents ; équivaut à `openlatch init`.

openlatch system hooks uninstall

Retire les hooks des agents ; équivaut à `openlatch uninstall`.

openlatch system hooks status

Affiche en détail les hooks installés pour chaque agent.

openlatch system model-relay

Regroupe les commandes qui inspectent ou basculent le relais de modèle, l'écouteur local qui enregistre les appels de modèle de chaque agent.

openlatch system model-relay status

Indique si le relais de modèle fonctionne, son port, et si les appels de modèle de chaque agent passent par lui. Un agent dont les hooks fonctionnent mais dont les appels de modèle ne sont pas enregistrés apparaît comme appliqué, mais non mesuré.

openlatch system model-relay enable

Active le relais de modèle dans la configuration, puis propose de redémarrer le daemon pour l'appliquer.

openlatch system model-relay disable

Désactive le relais de modèle dans la configuration, puis propose de redémarrer le daemon pour l'appliquer.

openlatch system model-relay explain

Affiche un enregistrement du relais de modèle par son identifiant, lu depuis le stockage local. Son contenu ne quitte jamais la machine.

openlatch system proxy

Regroupe les commandes qui inspectent, découvrent, définissent ou testent le proxy emprunté par chaque requête sortante.

openlatch system proxy status

Affiche la route empruntée par les requêtes sortantes, pour la CLI et, si elle diffère, pour le daemon.

openlatch system proxy discover

Relance la découverte de proxy du système d'exploitation, teste chaque candidat et conserve celui qui fonctionne.

openlatch system proxy set

Définit le proxy à la main. Un proxy défini ainsi n'est jamais remplacé par la découverte ni par la réparation automatique.

openlatch system proxy clear

Supprime le proxy et son identifiant ; les requêtes vont alors directement à leur destination.

openlatch system proxy test

Teste la route étape par étape : la connexion au proxy, TLS, HTTP et une réponse en flux.

openlatch system supervision

Regroupe les commandes qui gèrent le superviseur du système d'exploitation (launchd, systemd ou le Planificateur de tâches) qui démarre le daemon à l'ouverture de session et le redémarre. Tout reste au niveau de l'utilisateur.

openlatch system supervision install

Installe le superviseur du système d'exploitation, pour que le daemon démarre à l'ouverture de session et après un redémarrage.

openlatch system supervision uninstall

Retire le superviseur du système d'exploitation. Le daemon continue de tourner mais ne démarre plus seul.

openlatch system supervision status

Indique si le superviseur est installé et en marche, et lequel ce système utilise.

openlatch system supervision enable

Réactive la supervision après disable, et réinstalle le superviseur.

openlatch system supervision disable

Retire le superviseur et enregistre que la supervision est désactivée sur cette machine.

openlatch system telemetry

Regroupe les commandes qui gèrent la télémétrie d'usage, que vous pouvez désactiver.

openlatch system telemetry status

Indique si la télémétrie d'usage est activée, quel réglage en décide, et si cette version l'inclut.

openlatch system telemetry enable

Active la télémétrie d'usage.

openlatch system telemetry disable

Désactive la télémétrie d'usage.

openlatch system telemetry purge

Désactive la télémétrie d'usage et vide la file des événements non envoyés d'un daemon en cours.

openlatch system telemetry debug

Montre comment exécuter n'importe quelle commande pour qu'elle affiche les événements de télémétrie qu'elle enverrait.

openlatch system inventory

Regroupe les commandes qui inspectent le plan de configuration : les sources de configuration des agents que suit le client, leurs événements de changement, les nouvelles analyses et l'état de l'observateur.

openlatch system inventory list

Liste les sources de configuration que suit le client, d'après le journal d'audit local.

openlatch system inventory log

Affiche les événements de changement de configuration du journal d'audit local.

openlatch system inventory rescan

Analyse à nouveau, tout de suite, chaque source de configuration suivie.

openlatch system inventory status

Indique l'état de l'observateur de configuration : s'il est activé, si sa liste de sources est chargée et combien de sources il contient.

openlatch system inventory inspect

Affiche une source de configuration en détail : son empreinte actuelle et ses alertes en attente.

openlatch system inventory projects

Liste les dossiers de projet que le client connaît d'après les fichiers de configuration qu'il suit.

openlatch system inventory ack

Accuse réception d'une alerte de configuration en attente, pour que l'agent ne la voie plus rappelée.

openlatch update

Recherche une nouvelle version du client. Dans un terminal, elle demande ensuite s'il faut l'installer, et l'installe si vous acceptez. Sans terminal (un script, un pipe, un outil de gestion), elle se contente de signaler la version. --yes installe sans demander ; --check ne fait que vérifier.

Codes d'erreur

Chaque code que le client peut afficher, avec ce qui s'est passé et ce qu'il faut faire.

- [OL-1002 – Événement trop volumineux](#)
- [OL-1003 – Événement en double ignoré](#)
- [OL-1100 – Expression personnalisée invalide](#)
- [OL-1200 – Plateforme injoignable](#)
- [OL-1201 – Identifiant refusé par la plateforme](#)
- [OL-1202 – Limite de débit de la plateforme atteinte](#)
- [OL-1204 – Écriture dans l'outbox impossible](#)
- [OL-1205 – Outbox envoyée en partie seulement](#)
- [OL-1206 – Canal vers la plateforme saturé](#)

- [OL-1207 – Entrée de l'outbox mise de côté](#)
- [OL-1208 – Événement conservé expiré](#)
- [OL-1210 – Récupération du bundle de politiques impossible](#)
- [OL-1211 – Échec de la vérification du bundle de politiques](#)
- [OL-1212 – Bundle de politiques inutilisable](#)
- [OL-1213 – Rafraîchissement du bundle de politiques en retard](#)
- [OL-1214 – Une règle du bundle mise de côté](#)
- [OL-1215 – Politique locale désactivée](#)
- [OL-1216 – La nouvelle version ne lit pas le cache de politiques](#)
- [OL-1220 – Aucune route vers la plateforme](#)
- [OL-1221 – Proxy injoignable](#)
- [OL-1222 – Identifiant refusé par le proxy](#)
- [OL-1223 – Schéma de proxy non pris en charge](#)
- [OL-1224 – Certificat TLS non approuvé](#)
- [OL-1225 – Configuration automatique du proxy indisponible](#)
- [OL-1226 – Configuration du proxy invalide](#)
- [OL-1227 – Connexion directe non autorisée](#)
- [OL-1228 – Le proxy met en mémoire tampon les réponses en flux](#)
- [OL-1229 – Aucun ticket Kerberos pour le proxy](#)
- [OL-1240 – Ligne d'entrée d'évaluation mal formée](#)
- [OL-1241 – Bundle d'évaluation refusé](#)
- [OL-1242 – Aucun bundle pour évaluer](#)
- [OL-1243 – L'état de session ne correspond pas au bundle](#)
- [OL-1244 – Échec de l'évaluation d'une ligne](#)
- [OL-1250 – Réception suspendue pour raison de licence](#)
- [OL-1300 – Valeur de configuration invalide](#)
- [OL-1301 – Champ de configuration requis manquant](#)
- [OL-1302 – Configuration modifiée mais non appliquée](#)
- [OL-1400 – Aucun agent pris en charge trouvé](#)
- [OL-1401 – Écriture des réglages de l'agent impossible](#)
- [OL-1402 – Fichier de réglages de l'agent illisible](#)
- [OL-1403 – Autres hooks trouvés dans les réglages de l'agent](#)

- [OL-1404 – Installation du binaire de hook impossible](#)
- [OL-1405 – Hooks installés mais non exécutés](#)
- [OL-1406 – Configuration TOML de l'agent illisible](#)
- [OL-1407 – L'agent choisi ne peut pas recevoir de hooks](#)
- [OL-1408 – Composant Cline attendu absent](#)
- [OL-1409 – État d'un composant Cline inconnu](#)
- [OL-1410 – Les actions de Cline ne peuvent pas être refusées](#)
- [OL-1411 – Appels de modèle de Cline.app non capturés](#)
- [OL-1412 – Plugin OpenLatch non chargé dans Cline](#)
- [OL-1413 – Format inconnu du fichier de hooks de Cursor](#)
- [OL-1414 – Installation de Cursor non confirmée](#)
- [OL-1415 – Le hook de Cursor ne joint pas le daemon](#)
- [OL-1416 – Des agents restent sans daemon](#)
- [OL-1417 – Plugin de l'agent en attente de son premier chargement](#)
- [OL-1418 – Un autre fichier occupe le chemin du plugin](#)
- [OL-1419 – Version de l'agent sous le minimum pris en charge](#)
- [OL-1420 – État d'un composant de l'agent inconnu](#)
- [OL-1421 – Le plugin de l'agent ne joint pas le daemon](#)
- [OL-1422 – Le plugin de l'agent n'a pas été chargé](#)
- [OL-1423 – Relais de l'agent pas encore prouvé](#)
- [OL-1424 – Surface de l'agent non couverte par le relais](#)
- [OL-1425 – L'outil en ligne de commande d'un agent contourne le relais](#)
- [OL-1426 – Application de bureau de l'agent pas encore enregistrée](#)
- [OL-1427 – Le service de bureau de l'agent n'a pas pu démarrer](#)
- [OL-1500 – Port déjà utilisé](#)
- [OL-1501 – Daemon déjà en cours d'exécution](#)
- [OL-1502 – Le daemon n'est pas devenu opérationnel](#)
- [OL-1503 – Nouvelle version du client disponible](#)
- [OL-1504 – Mise à jour impossible à appliquer](#)
- [OL-1505 – Compilé depuis les sources, mise à jour automatique impossible](#)
- [OL-1506 – Le daemon n'a pas répondu à la mise à jour](#)
- [OL-1507 – Arrêt du daemon impossible](#)

- [OL-1510 – Une partie du daemon a redémarré](#)
- [OL-1511 – Une partie du daemon échoue sans cesse](#)
- [OL-1512 – Daemon dégradé](#)
- [OL-1513 – Rien ne redémarre le daemon](#)
- [OL-1600 – Aucun identifiant trouvé](#)
- [OL-1601 – Identifiant expiré](#)
- [OL-1602 – Trousseau indisponible](#)
- [OL-1603 – Accès au trousseau refusé](#)
- [OL-1604 – Échec du fichier d'identifiant chiffré](#)
- [OL-1605 – Délai de connexion dépassé](#)
- [OL-1606 – Échec de la connexion](#)
- [OL-1607 – Révocation de la clé impossible à la déconnexion](#)
- [OL-1608 – Organisation à choisir](#)
- [OL-1609 – Sélecteur d'organisation sans réponse](#)
- [OL-1610 – Organisation introuvable](#)
- [OL-1611 – Nom d'organisation ambigu](#)
- [OL-1612 – La clé d'API appartient à une autre organisation](#)
- [OL-1613 – Organisation non confirmée](#)
- [OL-1614 – Machine inscrite dans une autre organisation](#)
- [OL-1615 – Changement d'organisation non appliqué](#)
- [OL-1616 – Changement refusé : la clé vient de l'environnement](#)
- [OL-1617 – Code de connexion expiré](#)
- [OL-1618 – Compte sans organisation](#)
- [OL-1619 – Organisation indisponible](#)
- [OL-1620 – Trop de connexions en cours](#)
- [OL-1700 – Fichier de consentement à la télémétrie corrompu](#)
- [OL-1701 – Écriture du consentement à la télémétrie impossible](#)
- [OL-1702 – Lot de télémétrie non envoyé](#)
- [OL-1703 – Démarrage de la télémétrie impossible](#)
- [OL-1704 – Transmission de télémétrie refusée](#)
- [OL-1800 – Journal de réparation de doctor corrompu](#)
- [OL-1801 – Rien à restaurer](#)

- [OL-1802 – Écriture du fichier de diagnostic impossible](#)
- [OL-1803 – Fichier de diagnostic incomplet](#)
- [OL-1900 – Clé d'intégrité des hooks indisponible](#)
- [OL-1901 – Fichier d'état des hooks corrompu](#)
- [OL-1902 – Écriture du fichier d'état des hooks impossible](#)
- [OL-1903 – Version du marqueur de hook non prise en charge](#)
- [OL-1904 – Traitement d'une entrée de hook impossible](#)
- [OL-1905 – Ancien marqueur de hook mis à niveau](#)
- [OL-1906 – Hooks modifiés réparés](#)
- [OL-1910 – Remplacement du fichier de réglages impossible](#)
- [OL-1911 – Lien du fichier de réglages non résolu](#)
- [OL-1950 – Supervision non prise en charge sur ce système](#)
- [OL-1951 – Installation du superviseur impossible](#)
- [OL-1952 – Enregistrement du superviseur impossible](#)
- [OL-1953 – Commande refusée par le superviseur](#)
- [OL-2000 – Démarrage de la surveillance de configuration impossible](#)
- [OL-2001 – Liste des sources de configuration introuvable](#)
- [OL-2002 – Liste des sources de configuration invalide](#)
- [OL-2003 – Chemin suivi impossible à développer](#)
- [OL-2004 – Un chemin ne peut pas être suivi](#)
- [OL-2005 – Empreinte d'un fichier de configuration impossible](#)
- [OL-2006 – Nouvelle analyse de configuration partielle](#)
- [OL-2007 – Détails de début de session indisponibles](#)
- [OL-2008 – Suivi de configuration désactivé](#)
- [OL-9999 – Erreur interne inattendue](#)
- [OL-RELAY-CA – Certificat du relais de modèle non approuvé](#)
- [OL-RELAY-CURSOR-UNPROVEN – Relais de Cursor pas encore prouvé](#)
- [OL-RELAY-CURSOR-UNSUPPORTED – Surface de Cursor non couverte par le relais](#)
- [OL-RELAY-DELIVERY – Acheminement de l'agent vers le relais non en place](#)
- [OL-RELAY-DOWN – Relais de modèle arrêté](#)
- [OL-RELAY-FOREIGN – Un autre processus occupe le port du relais](#)
- [OL-RELAY-IO – Erreur d'entrée-sortie du relais de modèle](#)

- [OL-RELAY-MISCONFIG](#) – Réglage du relais remplacé
- [OL-RELAY-NEXTSTART](#) – Le relais s'applique au prochain démarrage de l'éditeur
- [OL-RELAY-NOFIND](#) – Aucun enregistrement local pour cet identifiant
- [OL-RELAY-PORT](#) – Port du relais de modèle occupé
- [OL-RELAY-PORTS](#) – Aucun port de relais pour un fournisseur
- [OL-RELAY-PREFLIGHT](#) – Le relais de modèle ne joint pas le fournisseur
- [OL-RELAY-PROVIDER](#) – La configuration de l'agent contient une autre entrée de fournisseur
- [OL-RELAY-SERVE](#) – Relais de modèle arrêté sur une erreur
- [OL-RELAY-SETTINGS](#) – Les réglages de l'agent contiennent votre propre proxy.
- [OL-RELAY-STATEFILE](#) – Modification sûre du fichier d'état de l'agent impossible
- [OL-RELAY-TRUSTPROOF](#) – Confiance du relais pas encore prouvée ici
- [OL-RELAY-UNCOVERED](#) – Fournisseur non pris en charge par le relais

Journal des modifications

Produit Tous les produits ▼ 49 entrées

Atom : [tout](#) · [client](#) · [plateforme](#)

13 octobre 2026 Client

Client 0.6.19

`openlatch init` et `openlatch doctor` indiquent si les appels de modèle de chaque agent sont prêts à être enregistrés, et l'installation sous Windows demande moins.

- `openlatch init` et `openlatch doctor` indiquent, pour chaque agent, si ses appels de modèle sont prêts à être enregistrés, et ce qui manque encore.
- `openlatch init` demande avant de redémarrer un agent qui tournait déjà, et nomme ceux qu'il vous laisse redémarrer.
- Un Cursor déjà ouvert pendant `openlatch init` est couvert dès que vous le quittez et le rouvrez ; recharger ses fenêtres ne suffit pas.
- Sous Windows, l'installation ne déclenche plus d'invite du pare-feu, et la boîte de dialogue d'approbation du certificat est précédée de son empreinte pour que vous puissiez la vérifier.
- Sous Windows, des correctifs de stabilité pour Cline, pour une réinstallation après `openlatch uninstall --purge`, et pour les redémarrages.
- Travaux préparatoires pour un agent de code qu'OpenLatch ne prend pas encore en charge. Rien ne change pour les agents déjà connectés.

13 octobre 2026 Client

Client 0.6.18

Choisissez votre organisation à la connexion, et déplacez une machine vers une autre organisation.

- La connexion demande quelle organisation utiliser quand votre compte en a plusieurs ; `--org` ou `OPENLATCH_ORG` la nomme sans le demander.
- La nouvelle commande `openlatch system auth switch` déplace cette machine vers une autre de vos organisations.

- Sous Windows, `openlatch doctor` signale l'outil en ligne de commande d'un agent dont l'installation à l'échelle de la machine contourne le relais (OL-1425).
 - Travaux préparatoires pour un agent de code qu'OpenLatch ne prend pas encore en charge. Rien ne change pour les agents déjà connectés.
-

12 octobre 2026 Client

Client 0.6.17

Cursor est de nouveau couvert dès que ses fenêtres se rechargent.

- Un Cursor en attente est de nouveau couvert dès que ses fenêtres se rechargent.
 - Le relais enregistre chaque appel de Cursor une seule fois, et classe le trafic d'un agent de code qu'OpenLatch ne prend pas encore en charge.
-

12 octobre 2026 Client

Client 0.6.16

Le client connecte aussi l'application de bureau Codex, sans outil en ligne de commande et sans nouvel `openlatch init`.

- Le client connecte l'application de bureau Codex sur une machine sans outil en ligne de commande Codex, sans exécuter à nouveau `openlatch init`.
 - `openlatch doctor` ne signale plus de faux problème pour un agent de code qu'OpenLatch ne prend pas encore en charge.
-

12 octobre 2026 Plateforme

Une politique dans plusieurs Autonomy Zones

Rattachez une Agent Intent Policy à plusieurs zones, chacune avec son propre mode.

11 octobre 2026 Client

Client 0.6.15

Sous Windows, les outils en ligne de commande de Cursor et de Cline passent par le relais sans configuration.

- Sous Windows, les outils en ligne de commande de Cursor et de Cline font passer leur trafic de modèle par le relais sans configuration.
 - Sous Windows, la vérification du certificat du relais ne lit plus un fichier encore ouvert.
-

11 octobre 2026 Client

Client 0.6.14

Travaux préparatoires pour un agent de code qu'OpenLatch ne prend pas encore en charge.

- Travaux préparatoires du relais pour un agent de code qu'OpenLatch ne prend pas encore en charge. Rien ne change pour les agents déjà connectés.
-

10 octobre 2026 Client

Client 0.6.13

Travaux préparatoires pour un agent de code qu'OpenLatch ne prend pas encore en charge, et un correctif du relais sous Windows.

- Travaux préparatoires des hooks et de l'inventaire pour un agent de code qu'OpenLatch ne prend pas encore en charge. Rien ne change pour les agents déjà connectés.
 - Sous Windows, le certificat du relais est vérifié dans le contexte de l'utilisateur, et les processus d'arrière-plan démarrent sans ouvrir de fenêtre de console.
-

10 octobre 2026 Client

Client 0.6.12

Un suivi plus juste de l'usage des modèles et une meilleure lecture de PowerShell.

- `openlatch doctor --fix` indique l'état dans lequel il laisse le poste.
 - L'usage déjà signalé par un flux de modèle interrompu est conservé.
 - Un modèle n'est marqué inconnu que lorsque son usage est estimé.
 - Une clé d'API se distingue d'un jeton d'abonnement par sa forme.
 - Un paramètre PowerShell écrit `-Param: value` est lu de la même façon que `-Param value`.
-

9 octobre 2026 Client

Client 0.6.11

Les commandes PowerShell sont lues plus précisément.

- Une URL dans une commande PowerShell est désormais lue comme une URL, et non comme un chemin de fichier.
-

8 octobre 2026 Client

Client 0.6.10

Correctifs pour l'enrôlement et la lecture des commandes PowerShell.

- La vérification de santé du daemon au repos envoie désormais la clé d'API.
 - `openlatch init` enregistre désormais un `OPENLATCH_API_URL` exporté dans la configuration du client.
 - Un chemin de fichier est classé là où il aboutit, et non tel qu'il est écrit.
 - Les commandes PowerShell sont découpées en mots selon les règles de PowerShell.
-

8 octobre 2026 Plateforme

Les Autonomy Zones sur l'arborescence de votre organisation

Activez une Autonomy Zone sur n'importe quelle unité, et voyez quels agents chaque zone gouverne et pourquoi.

8 octobre 2026 Plateforme

Autonomy Zones et Agent Intent Policies

Définissez ce que chaque agent peut faire sous forme d'Agent Intent Policies, et exécutez chacune en mode Disabled, Monitor ou Enforce.

8 octobre 2026 Plateforme

API REST publique

Automatisez les opérations de la console avec des clés d'API d'organisation.

7 octobre 2026 Client

Client 0.6.9

Des rapports de configuration en direct, et des correctifs pour Cline, Cursor et macOS.

- Les changements de configuration d'un poste sont signalés au moment où ils se produisent.
 - Une session coéquipière de Claude Code est rattachée à la session qui la dirige.
 - Les skills, règles et serveurs MCP que chargent Codex CLI, Cursor et Cline sont inventoriés tels que chaque agent les charge réellement.
 - Les rapports de Cursor nomment le serveur MCP à chaque appel.
 - La mise à jour de Cline.app n'interrompt plus une conversation.
 - Une session Claude Code lancée à partir d'un plan enregistre ce plan comme premier prompt.
 - Sous macOS, le client ne demande plus l'accès aux dossiers Bureau, Téléchargements et Documents.
 - `openlatch init` ne présente plus la télémétrie d'usage comme anonyme.
 - Les chemins Windows natifs sont correctement classés lors des lectures et écritures de fichiers.
-

6 octobre 2026 Plateforme

Isolation des organisations appliquée par la base de données

La base de données elle-même sépare les données de chaque organisation.

6 octobre 2026 Client

Client 0.6.8

Correctifs du bundle de politiques et du relais.

- Un bundle de politiques en cache est correctement revalidé lorsque la plateforme le signale inchangé.
 - L'usage des sessions qu'aucun hook n'a enregistrées n'est plus consigné.
-

5 octobre 2026 Client

Client 0.6.7

Une évaluation qui tient compte des chemins, et des correctifs pour Cline.

- Les effets d'une action sont mis en relation avec les chemins qu'elle cible.
- Le plugin Cline s'applique à tous les outils, et pas seulement au shell.
- `openlatch init` attend la première livraison du plugin Cline au lieu d'échouer.
- Chaque action est décidée une seule fois, et non une seconde fois après son exécution.

- Les fils secondaires de Codex CLI sont rangés sous leur session parente.
 - L'usage est rattaché à la bonne installation.
-

5 octobre 2026 Plateforme

Des politiques qui lisent ce que fait une action

Les politiques d'intention reconnaissent ce que fait une commande, comme lire des secrets ou joindre un hôte réseau, et non la façon dont elle est écrite.

2 octobre 2026 Client

Client 0.6.6

Les appels de modèle de l'IDE Cursor passent par le relais.

- Les appels de modèle de l'IDE Cursor sont acheminés par le relais.
 - Un nouvel `openlatch init` n'échoue plus sur des composants encore en cours de démarrage.
 - `openlatch doctor` affiche une ligne d'état par agent et reconnaît le hub de Cline.app.
 - Une substitution de processus du shell n'est plus lue comme une cible de redirection.
-

1 octobre 2026 Client

Client 0.6.5

Un vocabulaire d'outils commun, et des verdicts Ask livrés selon chaque agent.

- Les outils de chaque agent sont rattachés à un vocabulaire commun ; un outil sans correspondance est tout de même enregistré sous son propre nom.
 - L'évaluation des politiques lit un catalogue de classification unique depuis le bundle de politiques.
 - Les verdicts Ask parviennent à chaque agent sous la forme qu'il prend en charge, avec un verdict défini à l'avance lorsqu'un agent n'a pas de ask propre.
-

1 octobre 2026 Client

Client 0.6.4

Prise en charge de Cursor.

- Les politiques décident sur les actions de Cursor, et le verdict parvient à Cursor dans son propre format.

- Le trafic de modèle de Cursor passe par le relais et est enregistré comme usage, jamais comme contenu.
 - Le raccordement de Cursor est vérifié, réparé et inventorié comme celui de tous les autres agents.
 - `openlatch uninstall` nomme les agents qu'une désinstallation partielle laisse connectés.
-

30 septembre 2026 Client

Client 0.6.3

Le plugin Cline parvient à Cline.app et à VS Code.

- Le plugin Cline est livré à Cline.app et à VS Code.
 - Le client vérifie que le plugin s'est chargé sur chaque hôte.
-

29 septembre 2026 Plateforme

Licence et consommation dans la console

Comparez ce pour quoi votre organisation est licenciée à ce qu'elle consomme, sur une seule page.

29 septembre 2026 Plateforme

Propriété des agents

Chaque agent est rattaché à un propriétaire issu de votre annuaire, et sa fonction métier peut piloter une politique.

29 septembre 2026 Plateforme

Des agents reconnaissables

Les agents portent un nom lisible et indiquent les sessions et projets qu'ils regroupent.

29 septembre 2026 Plateforme

Graphe d'exécution

Visualisez tout ce qu'un agent a fait sous forme de graphe, filtrez-le, et créez une règle à partir d'une commande.

29 septembre 2026 Plateforme

Économie des agents

Consultez les dépenses, la prévision et les sessions de chaque agent, avec des pistes d'économies classées.

29 septembre 2026 Client

Client 0.6.2

Remontée des données de Cline, réparation automatique et purge complète.

- Les sessions, prompts et appels d'outils de Cline sont transmis à la plateforme.
 - Le câblage du relais, la confiance des hooks et les fichiers du plugin Cline sont contrôlés contre toute altération et réparés.
 - Le client signale lorsqu'un bundle de politiques est en cours de livraison.
 - `openlatch uninstall` envoie un événement de désinstallation, et `openlatch uninstall --purge` supprime tout l'état du client.
-

29 septembre 2026 Client

Client 0.6.1

Cline via le relais, et gestion du certificat du relais.

- Le trafic de modèle de Cline est capté par le relais, y compris l'application de bureau Cline.
 - Le relais prouve que son certificat est approuvé avant qu'un agent ne soit dirigé vers lui.
 - Le certificat du relais peut être consulté, renouvelé, libéré et supprimé.
 - Seuls les agents détectés figurent dans les informations sur le poste.
 - Les hooks de Codex CLI sont approuvés sous Windows.
-

29 septembre 2026 Plateforme

Des politiques qui continuent de décider hors ligne

Chaque client conserve localement son bundle de politiques : l'application continue même réseau débranché.

24 septembre 2026 Client

Client 0.6.0

Un `openlatch update` étape par étape, avec un changement incompatible dans sa sortie JSON.

- Changement incompatible : `openlatch update --check --json` ne renvoie plus `min_supported_client.current`, `latest` et `severity` sont inchangés.
 - `openlatch update` affiche sa progression étape par étape, et le daemon redémarre ensuite.
 - Les hooks, règles et workflows de Cline sont remontés par l'inventaire de configuration.
-

23 septembre 2026 Client

Client 0.5.8

Capture des prompts de Cline.

- Les prompts sont captés sur la CLI de Cline et sur l'application de bureau Cline.
 - Les types d'agents non connectés ne sont plus signalés.
-

23 septembre 2026 Client

Client 0.5.7

Correctif de l'affichage du terminal sous Windows.

- Les couleurs et la progression s'affichent correctement sous Windows pour toutes les commandes, et pas seulement `openlatch init`.
-

23 septembre 2026 Client

Client 0.5.6

Maintenance de la publication.

- Changements dans la façon dont le client est construit et publié. Le client se comporte comme dans la version précédente.
-

22 septembre 2026 Client

Client 0.5.4

Correctifs de la supervision du daemon et de l'envoi de la télémétrie.

- Sous Windows, la tâche planifiée qui maintient le daemon en marche conserve son marqueur de version.
 - Le premier lot de télémétrie du daemon est envoyé en HTTPS.
-

22 septembre 2026 Client

Client 0.5.3

La télémétrie passe par la plateforme, et les binaires Linux démarrent sur des hôtes plus anciens.

- La télémétrie et les rapports de plantage passent par le point d'ingestion de la plateforme.
 - Les binaires Linux démarrent sur des hôtes minimaux et plus anciens.
-

21 septembre 2026 Client

Client 0.5.2

Correctifs du démarrage et de l'envoi des événements.

- Sous Windows, `openlatch` s'exécute comme un programme console, et ses binaires démarrent sur un poste neuf.
 - Les événements encore en file d'attente à l'arrêt sont envoyés.
 - Le daemon prend en compte une nouvelle connexion sans redémarrer.
 - L'envoi des événements reprend après chaque reconnexion à la plateforme.
-

20 septembre 2026 Client

Client 0.5.1

Maintenance de la publication.

- Changements dans la façon dont le client est construit et publié. Le client se comporte comme dans la version précédente.
-

8 septembre 2026 Client

Client 0.4.0

Les commandes des sous-systèmes passent sous `openlatch system`.

- Changement incompatible : `openlatch auth, hooks, model-relay, proxy, supervision, telemetry` et `inventory` deviennent `openlatch system <name>`, sans alias.
 - Changement incompatible : la section de configuration `[boundary]` devient `[model_relay]`, les variables `OPENLATCH_BOUNDARY_*` deviennent `OPENLATCH_MODEL_RELAY_*`, et les codes d'erreur `OL-BND-*` deviennent `OL-RELAY-*`.
 - `openlatch doctor` indique l'état en direct de chaque hook.
 - Les diagnostics de proxy indiquent le proxy que le client utilise réellement.
-

6 septembre 2026 Client

Client 0.3.3

Maintenance de la publication.

- Changements dans la façon dont le client est construit et publié. Le client se comporte comme dans la version précédente.
-

28 août 2026 Client

Client 0.2.2

Maintenance de la publication.

- Changements dans la façon dont le client est construit et publié. Le client se comporte comme dans la version précédente.
-

19 août 2026 Client

Client 0.1.18

Maintenance de la publication.

- Changements dans la façon dont le client est construit et publié. Le client se comporte comme dans la version précédente.
-

23 avril 2026 Client

Client 0.1.14

Correctif de l'adresse d'API par défaut.

- L'adresse d'API par défaut ne répète plus `/api`.
-

23 avril 2026 Client

Client 0.1.13

Le paquet npm se résout avec `npx @openlatch/client init`.

- Le paquet npm gagne un point d'entrée `client`, si bien que `npx @openlatch/client init` se résout.
 - L'aide n'affiche plus de bannière.
-

23 avril 2026 Client

Client 0.1.12

Maintenance de la publication.

- Changements dans la façon dont le client est construit et publié. Le client se comporte comme dans la version précédente.
-

23 avril 2026 Client

Client 0.1.8

Maintenance de la publication.

- Changements dans la façon dont le client est construit et publié. Le client se comporte comme dans la version précédente.
-

23 avril 2026 Client

Client 0.1.7

Le paquet npm fournit le point d'entrée du hook.

- Le paquet npm installe le point d'entrée `openlatch-hook`.
-

23 avril 2026 Client

Client 0.1.6

Maintenance de la publication.

- Changements dans la façon dont le client est construit et publié. Le client se comporte comme dans la version précédente.
-

OL-1002 — Événement trop volumineux

OL-1002 · ERR_EVENT_TOO_LARGE

Ce qui s'est passé

Le corps de l'événement dépasse la limite de taille de 1 Mo.

Que faire

Exécutez `openlatch doctor` pour vérifier le client. Si l'erreur se répète, créez un fichier de diagnostic avec `openlatch doctor --rescue` et envoyez-le au support.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1003 — Événement en double ignoré

OL-1003 · ERR_EVENT_DEDUPED

Ce qui s'est passé

L'événement a été dédoublonné dans la fenêtre de TTL (informatif).

Que faire

Rien à faire. Si cela arrive souvent, exécutez `openlatch doctor` pour vérifier que les hooks de chaque agent ne sont installés qu'une fois.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1100 — Expression personnalisée invalide

OL-1100 · ERR_INVALID_REGEX

Ce qui s'est passé

Un motif d'expression régulière personnalisé de `config.toml` n'a pas pu être compilé.

Que faire

Corrigez l'expression dans `~/.openlatch/config.toml`, puis exécutez `openlatch restart` pour la charger.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1200 — Plateforme injoignable

OL-1200 · ERR_CLOUD_UNREACHABLE

Ce qui s'est passé

Le point de terminaison de l'API Cloud est injoignable (erreur réseau ou échec DNS).

Que faire

Exécutez `openlatch system proxy test` pour vérifier la route entre cette machine et la plateforme.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1201 — Identifiant refusé par la plateforme

OL-1201 · ERR_CLOUD_AUTH_FAILED

Ce qui s'est passé

Le Cloud a renvoyé 401/403 — la clé d'API est invalide ou révoquée.

Que faire

Reconnectez-vous avec `openlatch system auth login`, ou définissez une clé valide dans `OPENLATCH_API_KEY` et exécutez `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1202 — Limite de débit de la plateforme atteinte

OL-1202 · ERR_CLOUD_RATE_LIMITED

Ce qui s'est passé

Le Cloud a renvoyé 429 — limite de débit dépassée ; respectez l'en-tête Retry-After.

Que faire

Rien à faire si cela se résout seul. Si cela persiste, exécutez `openlatch status` et transmettez le résultat au support.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1204 — Écriture dans l’outbox impossible

OL-1204 · ERR_OUTBOX_WRITE_FAILED

Ce qui s’est passé

Échec d’écriture dans l’outbox durable (disque plein, permission refusée, erreur d’E/S). Le daemon continue de traiter les événements — l’outbox est une aide à la reprise, pas un mécanisme de correction — mais l’événement est perdu pour la relecture vers le Cloud.

Que faire

Libérez de l’espace disque ou corrigez les autorisations de `~/ .openlatch`, puis exécutez `openlatch doctor` pour confirmer que l’outbox est accessible en écriture.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu’affiche le client.

OL-1205 — Outbox envoyée en partie seulement

OL-1205 · ERR_OUTBOX_DRAIN_PARTIAL

Ce qui s'est passé

La vidange de l'outbox s'est terminée avec des entrées encore en attente (défaillance transitoire du Cloud pendant la relecture, ou erreur d'E/S à la lecture du spool). Les entrées restantes sont conservées sur disque et réessayées au prochain signal de vidange.

Que faire

Rien à faire si la prochaine tentative réussit. Si des entrées attendent toujours, exécutez `openlatch doctor`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1206 — Canal vers la plateforme saturé

OL-1206 · ERR_CLOUD_CHANNEL_EMERGENCY

Ce qui s'est passé

Le canal de transfert vers le Cloud subit une congestion prolongée — le mode d'urgence est engagé. Des événements de hook en direct étaient abandonnés plus vite que le worker ne pouvait vider la file ; la relecture est suspendue et la file de repli peut être évincée pour retrouver de la marge. Levé automatiquement quand les abandons cessent pendant deux cycles de santé consécutifs.

Que faire

Exécutez `openlatch status` pour suivre le nombre d'événements, et `openlatch system proxy test` si la connexion à la plateforme est lente.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1207 — Entrée de l’outbox mise de côté

OL-1207 · ERR_OUTBOX_QUARANTINED

Ce qui s’est passé

Une entrée de l’outbox a échoué à répétition pendant la durée de vie actuelle du daemon et a été mise en quarantaine pour que la file puisse se vider au-delà. L’entrée est retirée de l’outbox sans jamais atteindre le Cloud — émis uniquement lorsque le nombre de tentatives par identifiant dépasse `OUTBOX_MAX_ATTEMPTS`.

Que faire

Exécutez `openlatch doctor --rescue` et envoyez le fichier de diagnostic au support pour que l’entrée soit examinée.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu’affiche le client.

OL-1208 — Événement conservé expiré

OL-1208 · ERR_OUTBOX_EXPIRED

Ce qui s'est passé

Un événement de compatibilité conservé a atteint son expiration durable explicite avant de pouvoir être livré. Il s'agit d'une expiration définitive, pas d'un problème d'authentification ou de transport.

Que faire

Exécutez `open\latch doctor` pour vérifier la connexion à la plateforme, afin que les événements soient livrés avant d'expirer.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1210 — Récupération du bundle de politiques impossible

OL-1210 · ERR_BUNDLE_FETCH_FAILED

Ce qui s'est passé

Une interrogation du policy bundle a échoué (erreur réseau, 5xx ou 401/403). Le bundle activé précédemment est conservé et continue d'appliquer les règles — fail-static, jamais fail-open. Niveau avertissement pour les échecs transitoires, niveau erreur pour les échecs d'authentification.

Que faire

Exécutez `openlatch doctor` pour trouver la cause, puis `openlatch system aip refresh` pour récupérer à nouveau le bundle.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1211 — Échec de la vérification du bundle de politiques

OL-1211 · ERR_BUNDLE_REJECTED

Ce qui s'est passé

Un bundle récupéré ou présent sur disque a échoué à la vérification et n'a PAS été activé : empreinte non conforme à l'ETag, un `organization_id` qui n'est pas le nôtre, une réponse 200 sans en-tête ETag (l'empreinte ne se trouve pas dans le corps, donc un corps invérifiable ne doit jamais être activé), ou un `bundle.json` en cache dont les octets ne correspondent plus à `bundle.meta.json`. Le bundle précédent reste résident ; le validateur stocké n'est délibérément PAS avancé, de sorte que la prochaine interrogation redemande le bundle et retente l'activation.

Que faire

Exécutez `openlatch system aip refresh` pour récupérer une copie neuve. Si l'erreur se répète, exécutez `openlatch doctor --rescue` et envoyez le fichier au support.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1212 — Bundle de politiques inutilisable

OL-1212 · ERR_BUNDLE_INVALID

Ce qui s'est passé

Un bundle de politiques n'a pas pu être lu : il est mal formé, il utilise une version de format que ce client ne connaît pas, ou il porte une signature que ce client ne peut pas vérifier.

Que faire

Mettez le client à jour avec `openlatch update`, puis exécutez `openlatch system aip refresh`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1213 — Rafraîchissement du bundle de politiques en retard

OL-1213 · ERR_BUNDLE_STALE

Ce qui s'est passé

Aucune interrogation réussie dans le délai `stale_warn_after_secs`. Simple avertissement — le bundle résident continue d'appliquer les règles. Mesuré depuis la dernière interrogation réussie (connectivité), et non depuis `built_at` (âge de la politique) ; une réponse 304 compte comme une interrogation réussie.

Que faire

Vérifiez la connexion de la machine à la plateforme avec `openlatch doctor`. Une fois la connexion rétablie, exécutez `openlatch system aip refresh` pour récupérer aussitôt le dernier bundle.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1214 — Une règle du bundle mise de côté

OL-1214 · ERR_RULE_SKIPPED

Ce qui s'est passé

Une règle d'un bundle par ailleurs valide a été traitée individuellement au moment de la projection — retirée, ou maintenue dans un mode plus faible — et le reste du bundle a été activé normalement. Délibérément PAS OL-1212, qui signifie *le bundle a été rejeté et le précédent conservé* — le réutiliser inverserait son sens pour quiconque alerte dessus.

Que faire

Mettez le client à jour avec `openlatch update` pour qu'il prenne en charge chaque règle du bundle, puis exécutez `openlatch system aip refresh`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1215 — Politique locale désactivée

OL-1215 · ERR_POLICY_DISABLED

Ce qui s'est passé

Le plan des politiques est désactivé dans la configuration : aucune règle n'est évaluée et chaque action est autorisée.

Que faire

Supprimez `enabled = false` de la section `[policy]` de `~/.openlatch/config.toml` (ou retirez `OPENLATCH_POLICY_ENABLED`), puis exécutez `openlatch restart`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1216 — La nouvelle version ne lit pas le cache de politiques

OL-1216 · ERR_READER_RETIREMENT_BLOCKED

Ce qui s'est passé

Un exécutable candidat ne peut pas charger le cache de politiques résident exact et ne peut donc pas remplacer l'évaluateur qui le peut.

Que faire

Exécutez `openlatch system aip refresh`, puis relancez la mise à jour avec `openlatch update`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1220 — Aucune route vers la plateforme

OL-1220 · ERR_EGRESS_UNREACHABLE

Ce qui s'est passé

La plateforme est totalement injoignable : aucun proxy n'est configuré et la connexion directe à la plateforme a échoué au niveau TCP/DNS. Distinct de OL-1221 ci-dessous — ce code se déclenche quand un `*proxy*` est sur le chemin et injoignable, alors que celui-ci signifie que le trafic sortant n'avait aucun proxy à incriminer. Vérifiez la résolution DNS sortante et les règles de pare-feu de l'hôte qui exécute le daemon.

Que faire

Vérifiez la résolution DNS et les règles de pare-feu de cette machine, puis exécutez `openlatch system proxy test` pour confirmer la route.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1221 — Proxy injoignable

OL-1221 · ERR_PROXY_UNREACHABLE

Ce qui s'est passé

Un proxy est configuré, mais le proxy lui-même n'a pas répondu : la connexion TCP à l'hôte du proxy a échoué, ou un `CONNECT HTTPS` à travers lui a expiré ou a été refusé. La destination n'est jamais atteinte, car le client ne parvient même pas à établir le tunnel vers elle. Vérifiez que l'hôte et le port du proxy sont corrects et joignables depuis cette machine avant de soupçonner la destination.

Que faire

Vérifiez l'hôte et le port du proxy avec `openlatch system proxy status`, puis exécutez `openlatch system proxy test`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1222 — Identifiant refusé par le proxy

OL-1222 · ERR_PROXY_AUTH_FAILED

Ce qui s'est passé

Le proxy a répondu `407 Proxy Authentication Required` : soit les identifiants proposés par le client ont été rejetés, soit il n'en avait aucun à proposer. Distinct de OL-1201/ERR_CLOUD_AUTH_FAILED — là, c'est la *plateforme* qui rejette notre clé d'API ; ici, c'est le contrôle propre du proxy, déclenché avant même que la requête n'atteigne la plateforme. Un identifiant valide la veille et un proxy d'entreprise dont le mot de passe a changé sont identiques sur le réseau ; l'opérateur doit donc revérifier l'identifiant du proxy en particulier, et non la clé d'API de la plateforme.

Que faire

Définissez l'identifiant actuel du proxy avec `openlatch system proxy set`, puis exécutez `openlatch system proxy test`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1223 — Schéma de proxy non pris en charge

OL-1223 · ERR_PROXY_SCHEME_UNSUPPORTED

Ce qui s'est passé

L'URL de proxy configurée indique un schéma que cette version ne sait pas utiliser : tout ce qui sort de `http/https/socks5/socks5h`, ou `negotiate` demandé sur une version compilée sans la fonctionnalité Kerberos. L'échec est immédiat lors de la résolution du proxy plutôt qu'un repli silencieux vers une connexion simple — un déclassement silencieux ferait contourner au trafic le proxy même que l'opérateur a configuré. Corrigez le schéma dans l'URL du proxy, ou recompilez avec la fonctionnalité que ce schéma exige.

Que faire

Définissez une URL de proxy commençant par `http`, `https`, `socks5` ou `socks5h` avec `openlatch system proxy set`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1224 — Certificat TLS non approuvé

OL-1224 · ERR_EGRESS_TLS_FAILED

Ce qui s'est passé

La négociation TLS avec la destination a échoué à la vérification du certificat. Sur un réseau d'entreprise, la cause la plus fréquente est un proxy ou un pare-feu à inspection TLS qui re-signe le trafic sortant avec une autorité d'interception absente du magasin de confiance du système et non fournie via `ca_bundle` — du point de vue du client, c'est identique à une véritable interception, et c'est précisément pourquoi il refuse la négociation au lieu de deviner. La solution consiste presque toujours à pointer `ca_bundle` vers la racine d'interception, jamais à désactiver la vérification.

Que faire

Faites pointer `OPENLATCH_CA_BUNDLE` (ou `ca_bundle` dans la section `[proxy]`) vers le certificat racine de votre réseau, puis exécutez `openlatch system proxy test`. Ne désactivez jamais la vérification.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1225 — Configuration automatique du proxy indisponible

OL-1225 · ERR_PAC_UNAVAILABLE

Ce qui s'est passé

Une source PAC (Proxy Auto-Config) ou WPAD est configurée, mais le système n'a pas pu l'évaluer : l'URL PAC n'a pas été résolue ou n'a pas servi de script, ou le moteur PAC de la plateforme n'est pas disponible sur ce système. Le trafic sortant se replie sur le proxy statique ou le comportement direct que la configuration déclare sous l'entrée PAC ; ce code est donc informatif, sauf si ce repli échoue lui aussi.

Que faire

Exécutez `openlatch system proxy discover` pour trouver une route qui fonctionne, ou définissez-en une à la main avec `openlatch system proxy set`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1226 — Configuration du proxy invalide

OL-1226 · ERR_PROXY_CONFIG_INVALID

Ce qui s'est passé

La configuration du proxy elle-même est mal formée : une URL de proxy illisible, un schéma auquel manque un champ d'identifiant requis, ou `allow_direct` et les paramètres du proxy qui se contredisent.

Contrairement à tous les autres codes de ce bloc — qui décrivent chacun un état réseau à l'exécution qui ne bloque jamais le démarrage — celui-ci FAIT échouer le démarrage : une configuration que le client ne peut même pas lire est une erreur de rédaction, pas un état réseau transitoire, et laisser le daemon démarrer avec elle ne ferait que déplacer l'échec à la première requête.

Que faire

Corrigez le proxy avec `openlatch system proxy set`, ou supprimez-le avec `openlatch system proxy clear`, puis exécutez `openlatch start`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1227 — Connexion directe non autorisée

OL-1227 · ERR_DIRECT_FORBIDDEN

Ce qui s'est passé

`allow_direct = false` et aucun proxy utilisable n'a été trouvé pour cette destination. Le trafic sortant refuse la requête plutôt que de se replier silencieusement sur une connexion directe — sur un réseau dont la politique impose que tout le trafic sortant passe par un proxy, un repli direct silencieux est un chemin d'exfiltration, pas une commodité. Corrigez la configuration du proxy, ou définissez `allow_direct = true` si le trafic sortant direct depuis cet hôte est réellement acceptable.

Que faire

Réparez le proxy avec `openlatch system proxy discover` ou `openlatch system proxy set`. Ne définissez `allow_direct = true` que si les connexions directes depuis cette machine sont acceptables.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1228 — Le proxy met en mémoire tampon les réponses en flux

OL-1228 · ERR_STREAM_BUFFERED

Ce qui s'est passé

Le proxy met en tampon une réponse en flux au lieu de transmettre les octets à mesure qu'ils arrivent : le tunnel CONNECT lui-même fonctionne, mais un flux Server-Sent Events s'accumule dans le tampon interne du proxy et arrive d'un seul bloc au lieu d'être diffusé. Il s'agit d'un diagnostic sur le comportement du proxy, pas d'un échec de connectivité — la requête aboutit quand même — et il est donc signalé comme un avertissement plutôt que de faire échouer l'appel. Cherchez un paramètre de mise en tampon des réponses ou d'inspection de contenu sur le proxy pour le point de terminaison en flux.

Que faire

Demandez à votre équipe réseau de désactiver la mise en mémoire tampon ou l'inspection de contenu pour la plateforme sur le proxy, puis exécutez `openlatch system proxy test`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1229 — Aucun ticket Kerberos pour le proxy

OL-1229 · ERR_NEGOTIATE_NO_TICKET

Ce qui s'est passé

`negotiate` (Kerberos/SPNEGO) a été choisi pour l'authentification au proxy, mais aucun ticket utilisable n'a pu être obtenu : pas de cache d'identifiants Kerberos, un TGT expiré, ou le SPN du proxy qui ne se résout pas depuis cet hôte. Distinct de OL-1222/ERR_PROXY_AUTH_FAILED — le proxy ne s'est même jamais vu proposer de jeton à accepter ou rejeter. La correction se fait côté hôte : exécutez `kinit`, ou vérifiez que la machine est jointe au domaine pour que Windows obtienne un ticket automatiquement.

Que faire

Exécutez `kinit`, ou vérifiez que la machine est jointe au domaine, puis exécutez `openlatch system proxy test`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1240 — Ligne d'entrée d'évaluation mal formée

OL-1240 · ERR_EVAL_ROW_MALFORMED

Ce qui s'est passé

Une trame row qui ne s'est pas désérialisée en ligne de requête : un event qui n'est pas un objet, un now_ms qui n'est pas un entier, un state_in qui n'est pas un fichier de registres. ****Une ligne, jamais le flux**** — la trame reçoit une réponse d'erreur et l'exécution continue, de sorte qu'une relecture indique « tous les événements évalués sauf quatre » au lieu de s'interrompre en cours de route.

Que faire

Corrigez la ligne indiquée dans la sortie et relancez l'évaluation.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1241 — Bundle d'évaluation refusé

OL-1241 · ERR_EVAL_BUNDLE_REJECTED

Ce qui s'est passé

L'étape 1 de l'analyse du bundle a échoué, donc **rien n'a été chargé** : l'enveloppe ne s'est pas désérialisée, ou son `client_floor` dépasse ce que ce moteur implémente. Écrit sous forme de trame `bundle_error`, un résultat différent d'un `bundle_ack` portant un `skipped` non vide — ce dernier signifie que le bundle EST chargé et qu'un seul artefact ne l'est pas.

Que faire

Mettez le client à jour avec `openlatch update`, puis relancez l'évaluation.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1242 — Aucun bundle pour évaluer

OL-1242 · ERR_EVAL_NO_BUNDLE

Ce qui s'est passé

Une row est arrivée sans bundle contre lequel l'évaluer : aucune trame d'en-tête n'a été acceptée et la ligne ne porte pas son propre bundle en ligne. La réponse est donnée ligne par ligne, car la prochaine trame d'en-tête peut très bien corriger la situation.

Que faire

Envoyez le bundle de politiques avant les lignes, ou joignez un bundle à la ligne.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1243 — L'état de session ne correspond pas au bundle

OL-1243 · ERR_EVAL_STATE_MISMATCH

Ce qui s'est passé

Le `state_in` de la ligne ne concorde pas avec le `state_layout` du bundle. ****Mal formé, jamais complété**** — deux implémentations ne peuvent pas s'accorder sur quel compteur est lequel si l'une d'elles invente silencieusement un emplacement.

Que faire

Fournissez l'état de session que le client a renvoyé pour la ligne précédente de la même session.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1244 — Échec de l'évaluation d'une ligne

OL-1244 · ERR_EVAL_PANICKED

Ce qui s'est passé

Le moteur a paniqué pendant l'évaluation d'une ligne.

Que faire

Exécutez `openlatch doctor --rescue` et envoyez le fichier de diagnostic au support avec la ligne en échec.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1250 — Réception suspendue pour raison de licence

OL-1250 · ERR_CLOUD_LICENSE_BLOCKED

Ce qui s'est passé

La plateforme a refusé l'ingestion pour des raisons de **licence** : une réponse 402, ou une 503 dont le corps indique un code de licence de site.

Que faire

Suivez le lien de licence affiché par `openlatch status`, ou contactez votre interlocuteur OpenLatch si aucun n'est affiché.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1300 — Valeur de configuration invalide

OL-1300 · ERR_INVALID_CONFIG

Ce qui s'est passé

Le fichier de configuration contient une valeur invalide.

Que faire

Exécutez `openlatch doctor` pour trouver la valeur, corrigez-la dans `~/.openlatch/config.toml`, puis exécutez `openlatch restart`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1301 — Champ de configuration requis manquant

OL-1301 · ERR_MISSING_CONFIG_FIELD

Ce qui s'est passé

Un champ de configuration obligatoire est absent et n'a pas de valeur par défaut.

Que faire

Exécutez `openlatch init` pour réécrire une configuration complète.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1302 — Configuration modifiée mais non appliquée

OL-1302 · ERR_CONFIG_NOT_APPLIED

Ce qui s'est passé

Le fichier sur disque a changé depuis que le daemon en cours d'exécution l'a lu ; le processus sert donc encore les valeurs précédentes. Le fichier n'a aucun problème — `openlatch system proxy set` et les commandes voisines l'écrivent correctement et le signalent — et le daemon non plus. Les deux sont simplement désynchronisés, et seul un redémarrage comble l'écart.

Que faire

Exécutez `openlatch restart` pour appliquer la nouvelle configuration.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1400 — Aucun agent pris en charge trouvé

OL-1400 · ERR_HOOK_AGENT_NOT_FOUND

Ce qui s'est passé

Aucun agent IA pris en charge n'a été détecté sur cette machine.

Que faire

Installez un agent pris en charge, puis exécutez à nouveau `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1401 — Écriture des réglages de l'agent impossible

OL-1401 · ERR_HOOK_WRITE_FAILED

Ce qui s'est passé

Impossible de lire ou d'écrire le settings.json de l'agent (permissions, erreur d'E/S).

Que faire

Corrigez les autorisations du fichier de réglages de l'agent, puis exécutez à nouveau `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1402 — Fichier de réglages de l'agent illisible

OL-1402 · ERR_HOOK_MALFORMED_JSONC

Ce qui s'est passé

Le fichier `settings.json` contient du JSONC mal formé qui ne peut pas être analysé.

Que faire

Corrigez l'erreur de syntaxe dans le fichier `settings.json` de l'agent, puis exécutez à nouveau `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1403 — Autres hooks trouvés dans les réglages de l'agent

OL-1403 · ERR_HOOK_CONFLICT

Ce qui s'est passé

Des hooks non OpenLatch existants ont été détectés dans settings.json (avertissement, non bloquant).

Que faire

Rien à faire si ces hooks sont attendus. Exécutez `openlatch system hooks status` pour voir chaque hook installé.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1404 — Installation du binaire de hook impossible

OL-1404 · ERR_HOOK_BINARY_UNRESOLVABLE

Ce qui s'est passé

Aucun binaire `openlatch-hook` n'a pu être placé dans `<ol_dir>/bin/` : toute commande de hook écrite maintenant se résoudrait en un nom nu et mourrait avec le code de sortie 127 à chaque appel d'outil de l'agent. Fatal pour `init` — une installation qui ne peut pas résoudre son propre binaire de hook n'est pas une installation réussie — et signalé (non appliqué) par `doctor --fix`.

Que faire

Vérifiez les autorisations et l'espace libre dans `~/ .openlatch`, puis exécutez à nouveau `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1405 — Hooks installés mais non exécutés

OL-1405 · ERR_HOOK_NOT_ARMED

Ce qui s'est passé

Les hooks de l'agent sont installés et corrects sur disque, et pourtant l'agent ne les exécutera jamais — rien n'est donc appliqué. Trois façons d'y arriver, un code et trois remèdes : notre binaire de hook ne se résout pas ; l'agent n'a pas été informé qu'il doit faire confiance au hook (Codex marque une commande de hook nouvelle ou modifiée pour révision et l'ignore jusqu'à ce que le développeur exécute `/hooks`) ; ou un paramètre d'administration supprime en bloc les hooks non gérés.

Que faire

Exécutez `openLatch doctor` : il indique laquelle des trois causes il a trouvée et l'étape qui la corrige.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1406 — Configuration TOML de l'agent illisible

OL-1406 · ERR_HOOK_MALFORMED_TOML

Ce qui s'est passé

Un fichier de configuration d'agent au format TOML — le `config.toml` de Codex CLI — est présent et n'est pas du TOML valide : la table de fournisseur du model relay ne peut donc être ni écrite ni annulée sans détruire ce qu'elle contient déjà.

Que faire

Corrigez l'erreur de syntaxe dans le `config.toml` de l'agent, puis exécutez à nouveau `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1407 — L'agent choisi ne peut pas recevoir de hooks

OL-1407 · ERR_HOOK_AGENT_NOT_INSTALLABLE

Ce qui s'est passé

L'agent est installé sur cette machine et sa surface de hooks ne peut pas du tout être modifiée : `--agent` n'a donc sélectionné rien sur quoi cette commande puisse agir.

Que faire

Relancez la commande sans `--agent`, ou nommez un agent pris en charge par cette version.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1408 — Composant Cline attendu absent

OL-1408 · ERR_CLINE_SURFACE_ABSENT

Ce qui s'est passé

Une surface Cline que cette version attendait et n'a pas trouvée — la racine du stockage, le répertoire Hooks/, plugins/, le registre MCP, ou une extension de la lignée Cline dans une racine d'éditeur connue.

Que faire

Exécutez `openlatch doctor` pour voir quelle partie manque, installez ou mettez à jour Cline, puis exécutez `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1409 — État d'un composant Cline inconnu

OL-1409 · ERR_CLINE_SURFACE_UNDETERMINED

Ce qui s'est passé

Une surface Cline dont l'état n'a pas pu être déterminé.

Que faire

Exécutez `openlatch doctor` pour voir quel chemin a échoué, et corrigez ses autorisations.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1410 — Les actions de Cline ne peuvent pas être refusées

OL-1410 · ERR_CLINE_NOT_ENFORCING

Ce qui s'est passé

Cline est installé, ses dix scripts de hook sont sur disque et capturent, et **rien** de ce qu'ils observent ne peut être refusé**.**

Que faire

Exécutez `openlatch doctor` pour savoir pourquoi le plugin n'est pas chargé, et suivez la solution qu'il donne.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1411 — Appels de modèle de Cline.app non capturés

OL-1411 · ERR_CLINE_APP_NOT_CAPTURED

Ce qui s'est passé

Le trafic de modèle de Cline.app ne passe pas par le relay : son hub tourne sans la capture de cette installation (démarré avant elle, ou depuis une version plus ancienne de Cline.app), le port du hub est occupé par autre chose, le hub est géré sur une plateforme où cela n'est pas prouvé (D-47), la plateforme n'a pas de chemin d'installation établi et l'application ne peut donc pas être détectée (A-8), ou 06 §0 a prouvé que l'application ne peut pas être capturée (§F). Une seule condition, « Cline.app n'est pas capturé » ; la raison figure sur la ligne de détail.

Que faire

Fermez toutes les conversations Cline et quittez Cline.app, puis rouvrez-la pour que son hub redémarre à travers le relais.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1412 — Plugin OpenLatch non chargé dans Cline

OL-1412 · ERR_CLINE_PLUGIN_NOT_LOADED

Ce qui s'est passé

Un hôte Cline (le hub de Cline.app, l'extension VS Code) ne peut pas charger ou ne charge pas le plugin OpenLatch : un verdict ne peut donc pas y être délivré et l'hôte est en Monitor seulement, via les hooks de fichiers. Une seule condition ; la raison est le détail : host-too-old, delivery-off, probe-failed, pending-restart, runtime-missing, windows-launcher-unproven, not-managed, customer-override, not-detected-on-platform, windows-launcher-missing. Désactivé n'est jamais un succès (cli-output-contract).

Que faire

Exécutez `openlatch doctor` pour connaître la raison. Si la livraison du plugin est désactivée, définissez `plugin_delivery = true` dans la section `[cline]` de `~/.openlatch/config.toml` et exécutez `openlatch restart`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1413 — Format inconnu du fichier de hooks de Cursor

OL-1413 · ERR_CURSOR_HOOK_FILE_MALFORMED

Ce qui s'est passé

Le `hooks.json` de Cursor n'est pas un fichier dans lequel cette version peut écrire sans risque : il ne s'analyse pas, ou sa version racine est présente et n'est pas l'entier 1 que Cursor documente.

L'installation refuse plutôt que d'écrire dans un format qu'elle ne connaît pas — une supposition à cet endroit pourrait empêcher le chargement de tout le fichier, y compris les hooks du développeur. Le remède consiste à corriger ou déplacer le fichier et à relancer `openlatch init`.

Que faire

Corrigez le fichier ou déplacez-le, puis exécutez à nouveau `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1414 — Installation de Cursor non confirmée

OL-1414 · ERR_CURSOR_SURFACE_UNDETERMINED

Ce qui s'est passé

Une surface Cursor (l'IDE ou la CLI) a sur cet hôte un emplacement d'installation qui ne peut pas être confirmé comme Cursor : l'application est présente mais son `resources/app/package.json` est absent, illisible ou ne nomme aucun produit Cursor, ou le répertoire des versions de la CLI existe mais ne peut pas être lu. `doctor` le signale comme dégradé, en nommant la surface et ce qu'il n'a pas pu lire. La version est lue uniquement depuis des fichiers — `doctor` n'exécute jamais Cursor, dont le binaire macOS lance l'interface graphique.

Que faire

Exécutez `open\latch doctor` pour voir quel fichier il n'a pas pu lire, puis réinstallez ou mettez à jour Cursor.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1415 — Le hook de Cursor ne joint pas le daemon

OL-1415 · ERR_CURSOR_HOOK_UNREACHABLE

Ce qui s'est passé

Cursor a exécuté le hook `OpenLatch`, mais le hook n'a pas pu joindre le daemon. Les événements de Cursor n'atteignent le client que plus tard, depuis le journal de secours, et chaque action de Cursor se poursuit sans décision jusqu'à ce que ce soit corrigé.

Que faire

Exécutez `openlatch start` pour vous assurer que le daemon tourne, puis `openlatch doctor`. Les erreurs de hook propres à Cursor se trouvent dans son canal de sortie `Hooks`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1416 — Des agents restent sans daemon

OL-1416 · ERR_HOOK_AGENTS_STRANDED

Ce qui s'est passé

`openlatch uninstall --agent` n'a déconnecté qu'une partie des agents de cet hôte et, comme documenté, a tout de même arrêté le daemon et enregistré la supervision `mode = disabled` pour toute la machine (#325 : un daemon en cours d'exécution interprète la suppression comme une altération et le réinstalle). Chaque agent laissé de côté reste relié à un daemon qui ne tourne plus : chacun de ses appels de hook échoue en mode ouvert vers `fallback.jsonl` et rien n'est appliqué.

Que faire

Exécutez `openlatch start`, et `openlatch system supervision enable` si le daemon était supervisé.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1417 — Plugin de l'agent en attente de son premier chargement

OL-1417 · ERR_OPENCODE_PLUGIN_PENDING

Ce qui s'est passé

Ce code provient de l'intégration du client avec OpenCode, un agent qui n'est pas encore pris en charge. Le client a écrit le fichier de plugin de l'agent, et aucun chargement n'a été signalé depuis. L'agent ne charge ses plugins qu'au démarrage de son serveur ; la vérification reste donc en attente jusque-là.

Que faire

Redémarrez l'agent, puis exécutez `openlatch doctor` pour confirmer que la vérification est levée.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1418 — Un autre fichier occupe le chemin du plugin

OL-1418 · ERR_OPENCODE_PLUGIN_FOREIGN

Ce qui s'est passé

Ce code provient de l'intégration du client avec OpenCode, un agent qui n'est pas encore pris en charge. Un fichier qu'OpenLatch n'a pas écrit se trouve là où le client écrirait le plugin de l'agent, et le client ne l'écrase jamais.

Que faire

Déplacez ce fichier ailleurs, puis exécutez à nouveau `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1419 — Version de l'agent sous le minimum pris en charge

OL-1419 · ERR_OPENCODE_VERSION_UNSUPPORTED

Ce qui s'est passé

Ce code provient de l'intégration du client avec OpenCode, un agent qui n'est pas encore pris en charge. La version installée de l'agent est plus ancienne que le minimum attendu par le client.

Que faire

Mettez l'agent à jour vers une version récente, puis exécutez `open\latch doctor`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1420 — État d'un composant de l'agent inconnu

OL-1420 · ERR_OPENCODE_SURFACE_UNDETERMINED

Ce qui s'est passé

Ce code provient de l'intégration du client avec OpenCode, un agent qui n'est pas encore pris en charge. L'agent est présent, mais le client n'a pas pu établir l'état de l'un de ses composants.

Que faire

Exécutez `open\latch doctor` pour voir quel composant il n'a pas pu vérifier.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1421 — Le plugin de l'agent ne joint pas le daemon

OL-1421 · ERR_OPENCODE_HOOK_UNREACHABLE

Ce qui s'est passé

Ce code provient de l'intégration du client avec OpenCode, un agent qui n'est pas encore pris en charge. La vérification que le plugin de l'agent peut joindre le daemon a échoué.

Que faire

Exécutez `openlatch doctor --fix`, puis redémarrez l'agent.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1422 — Le plugin de l'agent n'a pas été chargé

OL-1422 · ERR_OPENCODE_PLUGIN_NOT_LOADED

Ce qui s'est passé

Ce code provient de l'intégration du client avec OpenCode, un agent qui n'est pas encore pris en charge. L'agent a envoyé des appels de modèle par le relais, mais son plugin n'a jamais signalé son chargement ; il a donc pu être ignoré.

Que faire

Redémarrez l'agent et vérifiez qu'il n'est pas lancé avec `--pure`, qui ignore tous les plugins.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1423 — Relais de l'agent pas encore prouvé

OL-1423 · ERR_OPENCODE_RELAY_UNPROVEN

Ce qui s'est passé

Ce code provient de l'intégration du client avec OpenCode, un agent qui n'est pas encore pris en charge. Le relais de modèle est configuré pour l'agent, mais aucun de ses appels de modèle n'a encore été enregistré.

Que faire

Lancez l'agent une fois depuis un shell qui charge l'environnement OpenLatch (`env . sh`, ou `env . ps1` dans PowerShell).

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1424 — Surface de l’agent non couverte par le relais

OL-1424 · ERR_OPENCODE_RELAY_UNSUPPORTED

Ce qui s’est passé

Ce code provient de l’intégration du client avec OpenCode, un agent qui n’est pas encore pris en charge. Une surface ou un transport de l’agent ne peut pas passer par le relais de modèle ; ses appels de modèle vont donc directement au fournisseur. La sortie de doctor en indique la raison.

Que faire

Exécutez `open\latch doctor` pour voir quelle surface est concernée.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu’affiche le client.

OL-1425 — L'outil en ligne de commande d'un agent contourne le relais

OL-1425 · ERR_CLI_SHIM_SHADOWED

Ce qui s'est passé

Sous Windows, l'outil en ligne de commande d'un agent connecté, comme la CLI de Cursor ou de Cline, se résout par le PATH de la machine vers un exécutable autre que le lanceur OpenLatch. Une installation à l'échelle de la machine masque le lanceur ; les appels de modèle de cet outil vont donc directement au fournisseur. Doctor lit seulement les chemins et les métadonnées des fichiers : il n'exécute jamais l'exécutable et ne modifie jamais le PATH de la machine.

Que faire

Supprimez l'exécutable que `openlatch doctor` indique, ou demandez à un administrateur de placer l'entrée du PATH de la machine après celle du Path de l'utilisateur. Exécutez ensuite à nouveau `openlatch doctor`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1426 — Application de bureau de l'agent pas encore enregistrée

OL-1426 · ERR_OPENCODE_DESKTOP_SERVICE_UNARMED

Ce qui s'est passé

Ce code provient de l'intégration du client avec OpenCode, un agent qui n'est pas encore pris en charge. Le service d'arrière-plan partagé de l'agent tourne sans le relais de modèle ; les appels de modèle de l'application de bureau ne sont donc pas encore enregistrés. L'agent continue de fonctionner. La sortie de doctor en indique la cause.

Que faire

Quittez l'application de bureau de l'agent et toutes ses sessions de terminal, puis rouvrez l'application de bureau.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1427 — Le service de bureau de l’agent n’a pas pu démarrer

OL-1427 · ERR_OPENCODE_DESKTOP_SERVICE_FAILED

Ce qui s’est passé

Ce code provient de l’intégration du client avec OpenCode, un agent qui n’est pas encore pris en charge. Le daemon a trouvé l’outil en ligne de commande fourni avec l’application de bureau de l’agent, mais n’a pas pu démarrer ni enregistrer le service d’arrière-plan de l’agent, ou le service a redémarré trop de fois. L’agent continue de fonctionner.

Que faire

Exécutez `openlatch doctor` : il indique l’exécutable et l’erreur. Corrigez ce qu’il indique, puis exécutez `openlatch restart`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu’affiche le client.

OL-1500 — Port déjà utilisé

OL-1500 · ERR_PORT_IN_USE

Ce qui s'est passé

Le port sélectionné est déjà utilisé par un autre processus.

Que faire

Libérez le port, ou choisissez-en un autre avec `OPENLATCH_PORT`, puis exécutez `openlatch start`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1501 — Daemon déjà en cours d'exécution

OL-1501 · ERR_ALREADY_RUNNING

Ce qui s'est passé

Une instance du daemon est déjà en cours d'exécution sur cette machine.

Que faire

Exécutez `openlatch status` pour voir le daemon en cours, ou `openlatch restart` pour le redémarrer.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1502 — Le daemon n'est pas devenu opérationnel

OL-1502 · ERR_DAEMON_START_FAILED

Ce qui s'est passé

Le processus du daemon a démarré mais le contrôle de santé a échoué dans le délai imparti. C'est aussi ce que signale `openlatch update` lorsque le daemon redémarré sur une nouvelle version n'est pas revenu, a échoué à son contrôle de santé après redémarrage, ou a été ramené à l'ancienne version.

Que faire

Exécutez `openlatch doctor` pour savoir pourquoi le daemon n'est pas opérationnel, puis `openlatch start`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1503 — Nouvelle version du client disponible

OL-1503 · ERR_VERSION_OUTDATED

Ce qui s'est passé

Une version plus récente d'openlatch est disponible (avertissement, non bloquant).

Que faire

Exécutez `openlatch update` pour l'installer.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1504 — Mise à jour impossible à appliquer

OL-1504 · ERR_UPDATE_VERIFY_FAILED

Ce qui s'est passé

La mise à jour n'a pas abouti : la vérification ou le téléchargement a échoué, ou les fichiers téléchargés ont échoué à un contrôle d'intégrité ou de démarrage. Avant l'échange, rien ne change sur le disque. Si l'échange lui-même a échoué, des copies de sauvegarde restent à côté des binaires pour la restauration.

Que faire

Exécutez à nouveau `openlatch update`. Si cela échoue une seconde fois, exécutez `openlatch doctor --rescue` et envoyez le fichier au support.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1505 — Compilé depuis les sources, mise à jour automatique impossible

OL-1505 · ERR_UPDATE_REFUSED_CARGO_INSTALL

Ce qui s'est passé

Ce client a été compilé depuis les sources avec `cargo install`, donc la mise à jour intégrée ne peut pas le remplacer.

Que faire

Recompilez-le depuis les sources pour le mettre à jour, ou réinstallez-le avec l'une des commandes de la page d'installation pour que la mise à jour intégrée puisse le gérer.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1506 — Le daemon n’a pas répondu à la mise à jour

OL-1506 · ERR_UPDATE_DAEMON_UNREACHABLE

Ce qui s’est passé

La CLI s’attendait à communiquer avec le point de terminaison `/admin/update` d’un daemon en cours d’exécution, mais n’a pas pu l’atteindre. Distinct de OL-1500 (le daemon de *l'utilisateur* n’a pas pu se lier à un port au démarrage). C’est aussi ce que signale `openlatch update` lorsque le daemon qu’il suit cesse de rapporter sa progression : même étape et même nombre d’octets pendant deux minutes. La mise à jour peut encore aboutir ; exécutez `openlatch status` pour voir quelle version est en service.

Que faire

Exécutez `openlatch status` pour voir quelle version tourne.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu’affiche le client.

OL-1507 — Arrêt du daemon impossible

OL-1507 · ERR_DAEMON_STOP_FAILED

Ce qui s'est passé

`openlatch stop/restart` n'a pas pu arrêter le daemon : il a survécu au `/shutdown` gracieux, à `SIGTERM` ET à `SIGKILL` (en pratique jamais — un processus qui survit à `SIGKILL` est bloqué dans une attente noyau non interruptible). L'utilisateur doit faire le nettoyage manuellement. Distinct de OL-1300 (une erreur de valeur de `*configuration*`), que ce chemin utilisait auparavant à tort.

Que faire

Arrêtez le processus avec le gestionnaire de tâches de votre système, ou redémarrez la machine, puis exécutez `openlatch start`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1510 — Une partie du daemon a redémarré

OL-1510 · ERR_TASK_PANICKED

Ce qui s'est passé

Une tâche supervisée interne au processus a paniqué. Le superviseur l'a interceptée via le `JoinHandle` de la tâche et redémarre le sous-système après un délai ; le reste du daemon continue de fonctionner. Journalisé au niveau `ERROR` à la première occurrence et au niveau `DEBUG` pour chaque occurrence consécutive, afin qu'un sous-système qui panique à chaque fenêtre de délai ne puisse pas inonder `daemon.log`.

Que faire

Rien à faire si elle se rétablit. Si cela se répète, exécutez `openlatch doctor --rescue` et envoyez le fichier au support.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1511 — Une partie du daemon échoue sans cesse

OL-1511 · ERR_TASK_RESTART_LIMIT

Ce qui s'est passé

Une tâche supervisée a échoué à chaque redémarrage pendant `[crate::supervision::task::RESTART_LIMIT_WARN_AT]` tentatives consécutives. Le superviseur continue de réessayer — il n'abandonne jamais — mais la série est signalée une fois pour qu'un sous-système définitivement cassé soit visible sans suivre les journaux DEBUG.

Que faire

Exécutez `openlatch doctor --rescue` et envoyez le fichier de diagnostic au support.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1512 — Daemon dégradé

OL-1512 · ERR_SUBSYSTEM_DEGRADED

Ce qui s'est passé

`/health` signale au moins un sous-système `RestartPolicy :: Always` qui n'est pas `running`. Le daemon répond aux requêtes mais ne fait pas tout ce qu'il annonce — status devient `degraded` au lieu de `ok`.

Que faire

Exécutez `openlatch doctor` pour voir quelle partie est arrêtée, puis `openlatch restart`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1513 — Rien ne redémarre le daemon

OL-1513 · ERR_NO_SUPERVISOR

Ce qui s'est passé

Aucun superviseur système n'est installé et personne ne l'a refusé : rien ne redémarrera le daemon s'il s'arrête ou si la machine redémarre.

Que faire

Exécutez `openlatch system supervision install`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1600 — Aucun identifiant trouvé

OL-1600 · ERR_NO_CREDENTIALS

Ce qui s'est passé

Aucun identifiant trouvé dans le trousseau, la variable d'environnement ou le fichier chiffré.

Que faire

Exécutez `openlatch system auth login`, ou définissez `OPENLATCH_API_KEY` et exécutez `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1601 — Identifiant expiré

OL-1601 · ERR_TOKEN_EXPIRED

Ce qui s'est passé

La clé d'API ou le jeton a expiré ou a été révoqué.

Que faire

Exécutez `openlatch system auth login` pour vous reconnecter.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1602 — Trousseau indisponible

OL-1602 · ERR_KEYCHAIN_UNAVAILABLE

Ce qui s'est passé

Le service de trousseau du système est indisponible (par exemple, pas de Secret Service sur un Linux sans interface graphique).

Que faire

Rien à faire si `openlatch system auth status` indique que vous êtes connecté. Sinon, exécutez `openlatch system auth login`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1603 — Accès au trousseau refusé

OL-1603 · ERR_KEYCHAIN_PERMISSION

Ce qui s'est passé

Le trousseau du système a refusé l'accès (erreur de permission).

Que faire

Autorisez le client dans les réglages de votre trousseau, puis exécutez `openlatch system auth login`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1604 — Échec du fichier d'identifiant chiffré

OL-1604 · ERR_FILE_FALLBACK_ERROR

Ce qui s'est passé

Le repli sur le fichier chiffré a échoué (erreur de déchiffrement, fichier manquant ou données corrompues).

Que faire

Exécutez `openlatch system auth login` pour enregistrer à nouveau l'identifiant.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1605 — Délai de connexion dépassé

OL-1605 · ERR_AUTH_TIMEOUT

Ce qui s'est passé

Le flux de connexion a expiré après 5 minutes d'attente du retour du navigateur.

Que faire

Exécutez à nouveau `openlatch system auth login` et terminez la connexion dans le navigateur qui s'ouvre.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1606 — Échec de la connexion

OL-1606 · ERR_AUTH_FLOW_FAILED

Ce qui s'est passé

Le flux de connexion a échoué (erreur du serveur de retour, requête tronquée, erreur de lancement du navigateur).

Que faire

Exécutez à nouveau `openlatch system auth login`. Si aucun navigateur ne peut s'ouvrir sur cette machine, définissez `OPENLATCH_API_KEY` et exécutez `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1607 — Révocation de la clé impossible à la déconnexion

OL-1607 · ERR_AUTH_REVOCATION_FAILED

Ce qui s'est passé

La révocation de la clé d'API côté serveur a échoué pendant la déconnexion (simple avertissement — le nettoyage local continue).

Que faire

Révoquez la clé dans la console sur `app.openlatch.ai` si elle ne doit plus fonctionner.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1608 — Organisation à choisir

OL-1608 · ERR_ORG_CHOICE_REQUIRED

Ce qui s'est passé

Votre compte appartient à deux organisations ou plus, aucune n'a été nommée, et personne n'était devant un terminal pour choisir.

Que faire

Nommez l'organisation avec `--org` ou `OPENLATCH_ORG`, ou exécutez à nouveau la commande dans un terminal et choisissez-la.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1609 — Sélecteur d'organisation sans réponse

OL-1609 · ERR_ORG_NOT_CHOSEN

Ce qui s'est passé

Le sélecteur d'organisation n'a reçu aucune réponse : l'entrée s'est terminée, a expiré ou n'a pas pu être lue.

Que faire

Exécutez à nouveau la commande et choisissez une organisation, ou nommez-la avec `--org` ou `OPENLATCH_ORG`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1610 — Organisation introuvable

OL-1610 · ERR_ORG_NOT_FOUND

Ce qui s'est passé

L'organisation nommée par `--org` ou `OPENLATCH_ORG` ne correspond à aucune de vos organisations.

Que faire

L'erreur liste vos organisations avec leurs slugs. Exécutez à nouveau la commande avec l'un de ces slugs dans `--org` ou `OPENLATCH_ORG`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1611 — Nom d'organisation ambigu

OL-1611 · ERR_ORG_AMBIGUOUS

Ce qui s'est passé

Le nom donné par `--org` ou `OPENLATCH_ORG` correspond à deux de vos organisations ou plus.

Que faire

Nommez l'organisation par son slug, qu'aucune autre organisation ne partage, ou exécutez la commande dans un terminal sans en nommer et choisissez-la dans la liste.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1612 — La clé d'API appartient à une autre organisation

OL-1612 · ERR_ORG_KEY_MISMATCH

Ce qui s'est passé

OPENLATCH_API_KEY appartient à une autre organisation que celle nommée.

Que faire

Utilisez une clé d'API de l'organisation nommée, ou nommez l'organisation à laquelle la clé appartient.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1613 — Organisation non confirmée

OL-1613 · ERR_ORG_UNVERIFIED

Ce qui s'est passé

Une organisation a été nommée, et la plateforme n'a pas pu être jointe pour la confirmer.

Que faire

Exécutez `openlatch system proxy test` pour vérifier la route de cette machine vers la plateforme, puis exécutez à nouveau la commande.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1614 — Machine inscrite dans une autre organisation

OL-1614 · ERR_ALREADY_ENROLLED

Ce qui s'est passé

Cette machine est inscrite dans une autre organisation. Seule la commande `openlatch system auth switch` la déplace.

Que faire

Exécutez `openlatch system auth switch` pour déplacer cette machine vers l'organisation voulue.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1615 — Changement d'organisation non appliqué

OL-1615 · ERR_SWITCH_DAEMON

Ce qui s'est passé

Le démon n'a pas accepté le changement d'organisation. Rien n'a changé.

Que faire

Exécutez `openlatch doctor` pour vérifier le démon, puis exécutez à nouveau `openlatch system auth switch`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1616 — Changement refusé : la clé vient de l'environnement

OL-1616 · ERR_SWITCH_ENV_KEY

Ce qui s'est passé

`openlatch system auth switch` a refusé, car la clé vient de `OPENLATCH_API_KEY`.

Que faire

Définissez `OPENLATCH_API_KEY` avec une clé de l'organisation voulue et exécutez `openlatch init`, ou supprimez la variable et exécutez `openlatch system auth switch`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1617 — Code de connexion expiré

OL-1617 · ERR_SIGN_IN_EXPIRED

Ce qui s'est passé

Le code de connexion a expiré ou a déjà été utilisé.

Que faire

Exécutez à nouveau `openlatch system auth login` et terminez la connexion dans le navigateur qui s'ouvre.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1618 — Compte sans organisation

OL-1618 · ERR_NO_ORGANIZATION

Ce qui s'est passé

Votre compte n'appartient à aucune organisation disponible.

Que faire

Demandez à un administrateur de votre organisation de vous inviter, puis reconnectez-vous.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1619 — Organisation indisponible

OL-1619 · ERR_ORG_UNAVAILABLE

Ce qui s'est passé

L'organisation choisie est suspendue ou supprimée, ou sa licence a pris fin.

Que faire

Choisissez une autre organisation, ou contactez l'administrateur de votre organisation.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1620 — Trop de connexions en cours

OL-1620 · ERR_SIGN_IN_THROTTLED

Ce qui s'est passé

Trop de connexions sont en cours pour ce compte.

Que faire

Attendez quelques minutes, puis exécutez à nouveau `openlatch system auth login`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1700 — Fichier de consentement à la télémétrie corrompu

OL-1700 · ERR_TELEMETRY_CONFIG_CORRUPT

Ce qui s'est passé

Le fichier de consentement `telemetry.json` contient du JSON invalide et ne peut pas être analysé.

Que faire

Enregistrez à nouveau votre choix avec `openlatch system telemetry enable` ou `openlatch system telemetry disable`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1701 — Écriture du consentement à la télémétrie impossible

OL-1701 · ERR_TELEMETRY_WRITE_FAILED

Ce qui s'est passé

Impossible d'écrire le fichier de consentement telemetry.json (permissions ou erreur d'E/S).

Que faire

Corrigez les autorisations de `~/ .openlatch`, puis enregistrez à nouveau votre choix avec `openlatch system telemetry disable` ou `openlatch system telemetry enable`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1702 — Lot de télémétrie non envoyé

OL-1702 · ERR_TELEMETRY_POST_FAILED

Ce qui s'est passé

Un POST de lot de télémétrie a échoué (informatif — événements abandonnés, pas de nouvel essai).

Que faire

Rien à faire. Pour ne plus envoyer de télémétrie d'usage, exécutez `openlatch system telemetry disable`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1703 — Démarrage de la télémétrie impossible

OL-1703 · ERR_TELEMETRY_INIT_FAILED

Ce qui s'est passé

Le sous-système de télémétrie n'a pas pu s'initialiser (clé intégrée manquante, erreur de canal).

Que faire

Rien à faire. Exécutez `openlatch system telemetry status` pour voir l'état actuel.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1704 — Transmission de télémétrie refusée

OL-1704 · ERR_TELEMETRY_HANDOFF_REJECTED

Ce qui s'est passé

Le daemon a refusé la transmission de télémétrie d'un processus CLI (`POST /admin/telemetry/events`) : le corps dépassait la limite d'octets, ou n'était pas un lot d'événements de télémétrie dans la limite d'événements. Rien n'a été relayé et rien n'est réessayé — les événements sont abandonnés (14). Seul un autre processus voit ceci, et la CLI l'ignore ; l'opérateur n'a rien à faire.

Que faire

Rien à faire.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1800 — Journal de réparation de doctor corrompu

OL-1800 · ERR_DOCTOR_JOURNAL_CORRUPT

Ce qui s'est passé

Le fichier journal de doctor `--fix` est présent mais ne peut pas être analysé (`--restore` ne peut pas s'en servir).

Que faire

Exécutez `openlatch doctor` pour vérifier l'état actuel, puis `openlatch init` pour réinstaller proprement si besoin.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1801 — Rien à restaurer

OL-1801 · ERR_DOCTOR_RESTORE_NO_JOURNAL

Ce qui s'est passé

`doctor --restore` a été invoqué sans journal --fix antérieur sur disque.

Que faire

Rien à faire. Exécutez `openlatch doctor` pour vérifier l'état actuel.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1802 — Écriture du fichier de diagnostic impossible

OL-1802 · ERR_DOCTOR_RESCUE_WRITE_FAILED

Ce qui s'est passé

`doctor --rescue` n'a pas pu écrire l'archive (disque plein, permissions, etc.).

Que faire

Libérez de l'espace disque ou corrigez les autorisations, puis exécutez à nouveau `openlatch doctor -rescue`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1803 — Fichier de diagnostic incomplet

OL-1803 · ERR_DOCTOR_RESCUE_PARTIAL

Ce qui s'est passé

`doctor --rescue` a produit une archive, mais un ou plusieurs collecteurs ont rencontré une erreur.

Que faire

Envoyez l'archive au support telle quelle ; elle indique les parties manquantes.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1900 — Clé d'intégrité des hooks indisponible

OL-1900 · ERR_HMAC_KEY_UNAVAILABLE

Ce qui s'est passé

Clé HMAC indisponible — le trousseau du système et le repli sur fichier ont tous deux échoué.

Que faire

Exécutez `openlatch doctor` pour vérifier le trousseau, puis `openlatch init` pour réinstaller.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1901 — Fichier d'état des hooks corrompu

OL-1901 · ERR_STATE_FILE_CORRUPT

Ce qui s'est passé

Le fichier `hook-state.json` existe mais ne peut pas être analysé (JSON invalide ou `schema_version` inconnue).

Que faire

Exécutez `openlatch init` pour réinstaller les hooks et réécrire le fichier.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1902 — Écriture du fichier d'état des hooks impossible

OL-1902 · ERR_STATE_FILE_WRITE_FAILED

Ce qui s'est passé

L'écriture atomique de `hook-state.json` a échoué après l'écriture de `settings.json`.

Que faire

Corrigez les autorisations de `~/ .openlatch`, puis exécutez à nouveau `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1903 — Version du marqueur de hook non prise en charge

OL-1903 · ERR_MARKER_SCHEMA_UNSUPPORTED

Ce qui s'est passé

Le champ `_openlatch.v` porte une version que nous ne comprenons pas.

Que faire

Mettez le client à jour avec `openlatch update`, puis exécutez `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1904 — Traitement d’une entrée de hook impossible

OL-1904 · ERR_CANONICALIZATION_FAILED

Ce qui s’est passé

La canonicalisation JCS a rejeté l’entrée de hook (erreur interne ; ne devrait pas se produire).

Que faire

Exécutez `openlatch doctor --rescue` et envoyez le fichier de diagnostic au support.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu’affiche le client.

OL-1905 — Ancien marqueur de hook mis à niveau

OL-1905 · ERR_LEGACY_MARKER_DETECTED

Ce qui s'est passé

Un ancien marqueur booléen `_open\latch: true` a été trouvé et sera mis à niveau (niveau info).

Que faire

Rien à faire.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1906 — Hooks modifiés réparés

OL-1906 · ERR_TAMPER_DETECTED

Ce qui s'est passé

Des entrées de hook ont été modifiées en dehors d'OpenLatch et le réconciliateur les a réparées.

Que faire

Trouvez l'outil ou le script qui modifie le fichier de réglages de l'agent, puis exécutez `openlatch system hooks status` pour confirmer que les hooks tiennent.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1910 — Remplacement du fichier de réglages impossible

OL-1910 · ERR_ATOMIC_WRITE_FAILED

Ce qui s'est passé

Le client n'a pas pu écrire la nouvelle version d'un fichier de réglages de l'agent à la place de l'ancienne, à cause d'une erreur d'autorisation ou de disque. Le fichier d'origine est inchangé.

Que faire

Corrigez les autorisations du dossier de réglages de l'agent, puis exécutez à nouveau `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1911 — Lien du fichier de réglages non résolu

OL-1911 · ERR_SYMLINK_CANONICALIZE_FAILED

Ce qui s'est passé

Un fichier de réglages de l'agent est un lien symbolique, et le client n'a pas pu résoudre sa cible.

Que faire

Vérifiez que la cible du lien existe et est lisible, puis exécutez à nouveau `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1950 — Supervision non prise en charge sur ce système

OL-1950 · ERR_SUPERVISION_UNSUPPORTED_OS

Ce qui s'est passé

Le client ne peut pas installer de superviseur du système d'exploitation sur ce système ; rien ne redémarre donc le daemon automatiquement.

Que faire

Démarrez le daemon vous-même avec `openlatch start` après chaque redémarrage.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1951 — Installation du superviseur impossible

OL-1951 · ERR_SUPERVISION_INSTALL_FAILED

Ce qui s'est passé

Le client n'a pas pu installer le superviseur du système d'exploitation (launchd, systemd ou Task Scheduler) : l'écriture de ses fichiers a échoué, ou le superviseur a refusé l'enregistrement.

Que faire

Lisez l'erreur du système d'exploitation affichée avec le code, qui en nomme la cause : une permission de dossier, un échec de `systemctl` ou un bus de session utilisateur absent sous Linux, ou un échec de `schtasks` sous Windows. Corrigez-la, puis exécutez `openlatch system supervision install`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1952 — Enregistrement du superviseur impossible

OL-1952 · ERR_SUPERVISION_BOOTSTRAP_FAILED

Ce qui s'est passé

Sur macOS, launchd a refusé d'enregistrer le superviseur du client, par exemple parce qu'aucune session utilisateur n'est attachée via SSH.

Que faire

Exécutez `openlatch system supervision install` depuis une session de bureau sur la machine.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-1953 — Commande refusée par le superviseur

OL-1953 · ERR_SUPERVISION_CONTROL_FAILED

Ce qui s'est passé

Un start / stop / restart confié au superviseur système a échoué — l'unité est enregistrée mais `systemctl` / `launchctl` / `schtasks` a refusé le verbe de contrôle. Les appelants se rabattent sur une action directe sur le processus, le comportement d'avant la supervision : il s'agit donc d'un chemin dégradé et non d'un échec bloquant.

Que faire

Exécutez `openlatch system supervision status` pour vérifier le superviseur.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-2000 — Démarrage de la surveillance de configuration impossible

OL-2000 · ERR_INVENTORY_INIT_FAILED

Ce qui s'est passé

L'initialisation du moniteur de configuration a échoué (erreur d'analyse du manifeste ou erreur de mise en place de l'observateur).

Que faire

Exécutez `openlatch system inventory status` pour voir la cause, puis `openlatch restart`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-2001 — Liste des sources de configuration introuvable

OL-2001 · ERR_INVENTORY_MANIFEST_NOT_FOUND

Ce qui s'est passé

Fichier manifeste introuvable au chemin configuré.

Que faire

Exécutez `openlatch init` pour le restaurer.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-2002 — Liste des sources de configuration invalide

OL-2002 · ERR_INVENTORY_MANIFEST_PARSE

Ce qui s'est passé

La validation du schéma du manifeste a échoué.

Que faire

Exécutez `openlatch update`, puis `openlatch init` pour restaurer une liste valide.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-2003 — Chemin suivi impossible à développer

OL-2003 · ERR_INVENTORY_PATH_EXPANSION

Ce qui s'est passé

L'expansion de chemin a échoué (variable inconnue, variable d'environnement manquante, échec d'un appel système).

Que faire

Cherchez OL-2003 dans `daemon.log`, dans le dossier `logs` du répertoire d'état du client : chaque entrée nomme le chemin qui n'a pas pu être développé. Corrigez la variable qu'il utilise, puis exécutez `openlatch restart`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-2004 — Un chemin ne peut pas être suivi

OL-2004 · ERR_INVENTORY_WATCHER_FAILED

Ce qui s'est passé

L'initialisation de l'observateur a échoué pour un chemin précis (il continue de fonctionner pour les autres chemins).

Que faire

Cherchez OL-2004 dans `daemon.log`, dans le dossier `logs` du répertoire d'état du client : chaque entrée nomme le chemin qui ne peut pas être surveillé. Vérifiez les permissions de ce chemin.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-2005 — Empreinte d'un fichier de configuration impossible

OL-2005 · ERR_INVENTORY_HASH_FAILED

Ce qui s'est passé

La canonicalisation JCS ou la normalisation NFC a échoué (fail-open : journalisation puis poursuite).

Que faire

Rien à faire. Si cela se répète pour le même fichier, exécutez `openlatch system inventory inspect` sur cette source.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-2006 — Nouvelle analyse de configuration partielle

OL-2006 · ERR_INVENTORY_RESCAN_PARTIAL

Ce qui s'est passé

Nouvelle analyse périodique partielle — certains chemins sont inaccessibles ; l'analyse se termine pour les autres.

Que faire

Cherchez OL-2006 dans `daemon.log`, dans le dossier `logs` du répertoire d'état du client : chaque entrée nomme un chemin que la nouvelle analyse n'a pas pu lire. Vérifiez les permissions de ces chemins.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-2007 — Détails de début de session indisponibles

OL-2007 · ERR_INVENTORY_ENRICH_FAILED

Ce qui s'est passé

L'enrichissement au démarrage de session a échoué (journalise un avertissement, continue avec l'événement non enrichi).

Que faire

Rien à faire. Si cela se répète, exécutez `openlatch system inventory status`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-2008 — Suivi de configuration désactivé

OL-2008 · ERR_INVENTORY_DISABLED

Ce qui s'est passé

Le moniteur du plan de configuration est désactivé dans la configuration : la dérive de configuration sur cet hôte n'est ni observée ni signalée.

Que faire

Retirez `OPENLATCH_INVENTORY_ENABLED` (ou définissez-la à `true`), et définissez `enabled = true` sous `[inventory_monitor]` dans `config.toml` (ou retirez cette ligne), puis exécutez `openlatch restart`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-9999 — Erreur interne inattendue

OL-9999 · ERR_BUG

Ce qui s'est passé

Code attribué à toutes les erreurs internes ou inattendues acheminées par `bug_report()`.

Que faire

Exécutez `openlatch doctor --rescue` et envoyez le fichier de diagnostic au support.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-CA — Certificat du relais de modèle non approuvé

OL-RELAY-CA · ERR_MODEL_RELAY_CA_UNTRUSTED

Ce qui s'est passé

L'autorité de certification du relay n'est pas approuvée par le magasin de confiance du système de l'utilisateur — l'installation a été ignorée (pas de session de bureau), refusée (boîte de dialogue annulée), ou la preuve de chaîne ou de magasin a échoué — aucune valeur de proxy n'a donc été écrite et l'agent passe en direct (non capturé). Remède : `openlatch doctor --fix` depuis une session de bureau, ou la commande d'installation de la plateforme. Affiché sous `PreflightFailed`.

Que faire

Exécutez `openlatch doctor --fix` depuis une session de bureau sur la machine.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-CURSOR-UNPROVEN — Relais de Cursor pas encore prouvé

OL-RELAY-CURSOR-UNPROVEN · ERR_CURSOR_RELAY_UNPROVEN

Ce qui s'est passé

Le relay de Cursor est câblé et aucune requête Cursor déchiffrée ne l'a atteint depuis (`agents.cursor.relay.traffic = "unproven"` dans `doctor`). Une seule preuve pour tout l'agent : le relay ne peut pas distinguer les requêtes de la CLI de celles de l'IDE. Remède : lancez un tour d'agent Cursor.

Que faire

Lancez un tour d'agent Cursor, puis exécutez `openlatch doctor`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-CURSOR-UNSUPPORTED — Surface de Cursor non couverte par le relais

OL-RELAY-CURSOR-UNSUPPORTED · ERR_CURSOR_RELAY_UNSUPPORTED

Ce qui s'est passé

Une surface Cursor que cette version ne peut pas diriger vers le relay (`agents.cursor.relay.{cli, ide} = "unsupported"`) : l'IDE, tant que sa preuve n'est pas validée. Ses appels de modèle partent en direct et ne sont pas capturés ; le plan des commandes applique toujours les règles. Ne figure pas non plus dans `[ERR_MODEL_RELAY_CODES]`.

Que faire

Gardez le client à jour avec `openlatch update` ; la surface est couverte dès qu'une version l'ajoute.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-DELIVERY — Acheminement de l’agent vers le relais non en place

OL-RELAY-DELIVERY · ERR_MODEL_RELAY_NOT_DELIVERED

Ce qui s’est passé

Le relais de modèle tourne et est prouvé, mais l’acheminement propre à l’agent vers le relais n’est pas en place : une clé de réglage pointe vers un autre proxy ou ne peut pas être lue, un environnement de lanceur manque, ou le runtime Node de l’agent n’a aucun certificat à approuver. Les appels de modèle de l’agent ne sont pas enregistrés tant que ce n’est pas réglé. La sortie de doctor indique lequel.

Que faire

Exécutez `openlatch doctor --fix`, puis redémarrez l’agent. Si `openlatch doctor` signale encore ce code, corrigez ce qu’il indique.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu’affiche le client.

OL-RELAY-DOWN — Relais de modèle arrêté

OL-RELAY-DOWN · ERR_MODEL_RELAY_NOT_RUNNING

Ce qui s'est passé

Rien ne répond au point de terminaison d'état du model relay sur le port fixé (connexion refusée).

Que faire

Exécutez `openlatch restart`, puis `openlatch system model-relay status`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-FOREIGN — Un autre processus occupe le port du relais

OL-RELAY-FOREIGN · ERR_MODEL_RELAY_PORT_FOREIGN

Ce qui s'est passé

Le port fixé du model relay est occupé par un processus qui n'est PAS le model relay OpenLatch (signature d'état erronée ou absente). Signalé par `openlatch doctor` : un agent câblé sur un écouteur loopback étranger envoie l'identifiant de son fournisseur à ce processus.

Que faire

Arrêtez le processus qui occupe le port, puis exécutez `openlatch restart`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-IO — Erreur d'entrée-sortie du relais de modèle

OL-RELAY-IO · ERR_MODEL_RELAY_IO

Ce qui s'est passé

Erreur d'E/S générique du model relay (utilisée par le pont `From<std::io::Error>`).

Que faire

Exécutez `openlatch system model-relay status`. Si l'erreur se répète, exécutez `openlatch doctor -rescue` et envoyez le fichier au support.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-MISCONFIG — Réglage du relais remplacé

OL-RELAY-MISCONFIG · ERR_MODEL_RELAY_MISCONFIGURED

Ce qui s'est passé

L'éditeur a manifestement chargé l'URL du relay d'un emplacement de fournisseur et utilise ce fournisseur, mais aucun trafic n'atteint le point de terminaison — quelque chose au-dessus du fichier (la configuration distante d'une organisation) la remplace.

Que faire

Vérifiez l'adresse de ce fournisseur dans la configuration à distance ou gérée de l'éditeur, puis exécutez `openlatch doctor`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-NEXTSTART — Le relais s'applique au prochain démarrage de l'éditeur

OL-RELAY-NEXTSTART · ERR_MODEL_RELAY_NEXT_START

Ce qui s'est passé

Un emplacement de fournisseur est câblé sur son point de terminaison du relay, et il n'est pas encore prouvé que l'éditeur qui le lit l'utilise.

Que faire

Rien à faire. La vérification est levée au premier appel de modèle passant par le relais après le redémarrage de l'éditeur.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-NOFIND — Aucun enregistrement local pour cet identifiant

OL-RELAY-NOFIND · ERR_MODEL_RELAY_FINDING_NOT_FOUND

Ce qui s'est passé

`model-relay explain <finding_id>` n'a trouvé aucun bloc de churn correspondant dans le stockage de rétention local borné (jamais produit ici, ou expiré ou purgé).

Que faire

Vérifiez l'identifiant, puis exécutez la commande sur la machine qui a produit l'enregistrement.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-PORT — Port du relais de modèle occupé

OL-RELAY-PORT · ERR_MODEL_RELAY_PORT_IN_USE

Ce qui s'est passé

Le port loopback fixé du model relay est déjà utilisé. Signalé haut et fort, jamais par un nouveau sondage silencieux (D-25) — la configuration de l'agent pointe vers ce port exact.

Que faire

Libérez le port, puis redémarrez le daemon avec `openlatch start`. Pour fonctionner sans relais de modèle, désactivez-le avec `openlatch system model-relay disable`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-PORTS — Aucun port de relais pour un fournisseur

OL-RELAY-PORTS · ERR_MODEL_RELAY_ENDPOINT_PORTS

Ce qui s'est passé

Un emplacement de fournisseur n'a pas pu obtenir de port de point de terminaison : son port dans le bloc de points de terminaison (`[port + 1, port + 32]` au-dessus du port principal du relay) est occupé par un autre processus, ou tous les ports du bloc sont déjà attribués.

Que faire

Libérez un port juste au-dessus du port du relais de modèle, puis exécutez `openlatch restart`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-PREFLIGHT — Le relais de modèle ne joint pas le fournisseur

OL-RELAY-PREFLIGHT · ERR_MODEL_RELAY_PREFLIGHT_FAILED

Ce qui s'est passé

Le model relay s'est lié à son port, mais une requête synthétique n'a pas pu faire l'aller-retour vers le fournisseur à travers lui : l'agent a donc été délibérément laissé non câblé (`model_relay::preflight`). Signalé par `init` — qui se termine avec un code non nul tout en laissant les hooks, le daemon et la supervision entièrement installés — et par `doctor`. Ce n'est PAS une condition fatale pour le daemon : il continue de fonctionner et se recâble dès que la sonde réussit.

Que faire

Exécutez `openlatch system proxy test` pour vérifier la route vers le fournisseur, puis `openlatch doctor`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-PROVIDER — La configuration de l’agent contient une autre entrée de fournisseur

OL-RELAY-PROVIDER · ERR_MODEL_RELAY_FOREIGN_PROVIDER

Ce qui s’est passé

Une table de fournisseur portant le nom qu’écrit OpenLatch existe déjà dans la configuration de l’agent et ne pointe ****pas**** vers notre écouteur loopback — elle appartient à quelqu’un d’autre. L’installation refuse de l’écraser, car la désinstallation ne pourrait alors pas restaurer le fichier à l’identique : l’annulation supprimerait une table qui n’a jamais été la nôtre et restaurerait un pointeur que nous n’avons jamais enregistré.

Que faire

Renommez ou supprimez la table `[model_providers.<name>]` en conflit, puis exécutez `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu’affiche le client.

OL-RELAY-SERVE — Relais de modèle arrêté sur une erreur

OL-RELAY-SERVE · ERR_MODEL_RELAY_SERVE

Ce qui s'est passé

Le serveur axum du model relay s'est arrêté sur une erreur d'E/S pendant le service.

Que faire

Exécutez `openlatch restart`, puis `openlatch system model-relay status`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-SETTINGS — Les réglages de l'agent contiennent votre propre proxy

OL-RELAY-SETTINGS · ERR_MODEL_RELAY_SETTINGS

Ce qui s'est passé

Le fichier de paramètres d'un agent contient, dans la clé qu'OpenLatch utiliserait, un proxy qu'OpenLatch n'a pas écrit, ou ne peut pas être analysé comme du JSONC. Il est laissé intact et cette surface n'est pas capturée. Remède : déplacez ce proxy dans la configuration [proxy] d'OpenLatch, par laquelle le trafic sortant du relay passe déjà.

Que faire

Déplacez ce proxy dans la configuration du client avec `openlatch system proxy set`, puis exécutez `openlatch init`.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-STATEFILE — Modification sûre du fichier d'état de l'agent impossible

OL-RELAY-STATEFILE · ERR_MODEL_RELAY_STATE_FILE

Ce qui s'est passé

Un fichier d'état d'agent qu'OpenLatch modifie pour câbler un emplacement de fournisseur n'a pas pu être modifié sans risque : il dépasse la limite de taille sous laquelle il est lu, ou ce n'est pas du JSON valide.

Que faire

Exécutez `openlatch doctor` pour savoir de quel fichier il s'agit, et corrigez-le ou réduisez-le.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-TRUSTPROOF — Confiance du relais pas encore prouvée ici

OL-RELAY-TRUSTPROOF · ERR_MODEL_RELAY_TRUST_UNPROVEN

Ce qui s'est passé

Le magasin de confiance de l'utilisateur a été vérifié, sur une plateforme dont le chemin de confiance n'a pas encore de preuve sur une machine réelle (Windows, Linux). State :: Degraded avec Resolution :: NothingToDo : une limite de couverture (D-30).

Que faire

Rien à faire.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

OL-RELAY-UNCOVERED — Fournisseur non pris en charge par le relais

OL-RELAY-UNCOVERED · ERR_MODEL_RELAY_UNCOVERED

Ce qui s'est passé

Un fournisseur configuré ne peut pas être pris en charge par le relay, pour une raison nommée : ses requêtes sont signées sur l'hôte, cette version ne sait pas où elle les enverrait autrement, ou son client n'expose aucun paramètre à rediriger.

Que faire

Rien à faire. Exécutez `openlatch doctor` pour voir quels fournisseurs ne sont pas pris en charge.

Note

Cette page est générée à partir du code source du client, à un commit figé. Le message ci-dessus est celui qu'affiche le client.

Client 0.6.19

13 octobre 2026 · Client

`openlatch init` et `openlatch doctor` indiquent si les appels de modèle de chaque agent sont prêts à être enregistrés, et l'installation sous Windows demande moins.

- `openlatch init` et `openlatch doctor` indiquent, pour chaque agent, si ses appels de modèle sont prêts à être enregistrés, et ce qui manque encore.
- `openlatch init` demande avant de redémarrer un agent qui tournait déjà, et nomme ceux qu'il vous laisse redémarrer.
- Un Cursor déjà ouvert pendant `openlatch init` est couvert dès que vous le quittez et le rouvrez ; recharger ses fenêtres ne suffit pas.
- Sous Windows, l'installation ne déclenche plus d'invite du pare-feu, et la boîte de dialogue d'approbation du certificat est précédée de son empreinte pour que vous puissiez la vérifier.
- Sous Windows, des correctifs de stabilité pour Cline, pour une réinstallation après `openlatch uninstall --purge`, et pour les redémarrages.
- Travaux préparatoires pour un agent de code qu'OpenLatch ne prend pas encore en charge. Rien ne change pour les agents déjà connectés.

[@openlatch/client@0.6.19](#)

Client 0.6.18

13 octobre 2026 · Client

Choisissez votre organisation à la connexion, et déplacez une machine vers une autre organisation.

- La connexion demande quelle organisation utiliser quand votre compte en a plusieurs ; `--org` ou `OPENLATCH_ORG` la nomme sans le demander.
- La nouvelle commande `openlatch system auth switch` déplace cette machine vers une autre de vos organisations.
- Sous Windows, `openlatch doctor` signale l'outil en ligne de commande d'un agent dont l'installation à l'échelle de la machine contourne le relais (OL-1425).
- Travaux préparatoires pour un agent de code qu'OpenLatch ne prend pas encore en charge. Rien ne change pour les agents déjà connectés.

[@openlatch/client@0.6.18](#)

Client 0.6.17

12 octobre 2026 · Client

Cursor est de nouveau couvert dès que ses fenêtres se rechargent.

- Un Cursor en attente est de nouveau couvert dès que ses fenêtres se rechargent.
- Le relais enregistre chaque appel de Cursor une seule fois, et classe le trafic d'un agent de code qu'OpenLatch ne prend pas encore en charge.

[@openlatch/client@0.6.17](#)

Client 0.6.16

12 octobre 2026 · Client

Le client connecte aussi l'application de bureau Codex, sans outil en ligne de commande et sans nouvel `openlatch init`.

- Le client connecte l'application de bureau Codex sur une machine sans outil en ligne de commande Codex, sans exécuter à nouveau `openlatch init`.
- `openlatch doctor` ne signale plus de faux problème pour un agent de code qu'OpenLatch ne prend pas encore en charge.

[@openlatch/client@0.6.16](#)

Une politique dans plusieurs Autonomy Zones

12 octobre 2026 · Plateforme

Une Agent Intent Policy peut s'appliquer dans plusieurs zones à la fois, chaque rattachement avec son propre mode. Avant de rattacher, de détacher ou de changer un mode, la console prévisualise l'impact. Un rattachement qui assouplirait ce qu'une zone supérieure bloque déjà est refusé.

Client 0.6.15

11 octobre 2026 · Client

Sous Windows, les outils en ligne de commande de Cursor et de Cline passent par le relais sans configuration.

- Sous Windows, les outils en ligne de commande de Cursor et de Cline font passer leur trafic de modèle par le relais sans configuration.
- Sous Windows, la vérification du certificat du relais ne lit plus un fichier encore ouvert.

[@openlatch/client@0.6.15](#)

Client 0.6.14

11 octobre 2026 · Client

Travaux préparatoires pour un agent de code qu'OpenLatch ne prend pas encore en charge.

- Travaux préparatoires du relais pour un agent de code qu'OpenLatch ne prend pas encore en charge.
Rien ne change pour les agents déjà connectés.

[@openlatch/client@0.6.14](#)

Client 0.6.13

10 octobre 2026 · Client

Travaux préparatoires pour un agent de code qu'OpenLatch ne prend pas encore en charge, et un correctif du relais sous Windows.

- Travaux préparatoires des hooks et de l'inventaire pour un agent de code qu'OpenLatch ne prend pas encore en charge. Rien ne change pour les agents déjà connectés.
- Sous Windows, le certificat du relais est vérifié dans le contexte de l'utilisateur, et les processus d'arrière-plan démarrent sans ouvrir de fenêtre de console.

[@openlatch/client@0.6.13](#)

Client 0.6.12

10 octobre 2026 · Client

Un suivi plus juste de l'usage des modèles et une meilleure lecture de PowerShell.

- `openlatch doctor --fix` indique l'état dans lequel il laisse le poste.
- L'usage déjà signalé par un flux de modèle interrompu est conservé.
- Un modèle n'est marqué inconnu que lorsque son usage est estimé.
- Une clé d'API se distingue d'un jeton d'abonnement par sa forme.
- Un paramètre PowerShell écrit `-Param:vaLue` est lu de la même façon que `-Param vaLue`.

[@openlatch/client@0.6.12](#)

Client 0.6.11

9 octobre 2026 · Client

Les commandes PowerShell sont lues plus précisément.

- Une URL dans une commande PowerShell est désormais lue comme une URL, et non comme un chemin de fichier.

[@openlatch/client@0.6.11](#)

Client 0.6.10

8 octobre 2026 · Client

Correctifs pour l'enrôlement et la lecture des commandes PowerShell.

- La vérification de santé du daemon au repos envoie désormais la clé d'API.
- `openlatch init` enregistre désormais un `OPENLATCH_API_URL` exporté dans la configuration du client.
- Un chemin de fichier est classé là où il aboutit, et non tel qu'il est écrit.
- Les commandes PowerShell sont découpées en mots selon les règles de PowerShell.

[@openlatch/client@0.6.10](#)

Les Autonomy Zones sur l'arborescence de votre organisation

8 octobre 2026 · Plateforme

Toute unité de votre organisation peut porter une Autonomy Zone ; une unité qui n'en a pas hérite de la zone la plus proche au-dessus d'elle. Chaque carte d'agent indique sa zone et les niveaux dont elle hérite. Vous pouvez affecter des agents à une zone à la main après avoir prévisualisé le changement, et définir une zone par défaut pour les agents dont le propriétaire n'est pas encore résolu.

Autonomy Zones et Agent Intent Policies

8 octobre 2026 · Plateforme

Chaque agent agit dans une Autonomy Zone construite à partir d'Agent Intent Policies selon cinq dimensions : Qualité, Économie, Sécurité, Sûreté et Conformité. Vous approuvez chaque politique sous la forme d'une phrase générée, avec ses cas de test générés affichés à côté. Les verdicts sont Allow, Ask, Block ou Optimize.

API REST publique

8 octobre 2026 · Plateforme

Les opérations de la console sont disponibles via une API REST publique versionnée sous `/api/public/v1`, authentifiée par des clés d'API d'organisation. Elle prend en charge les clés d'idempotence et des limites de débit, et chaque erreur provient d'un catalogue fermé.

Client 0.6.9

7 octobre 2026 · Client

Des rapports de configuration en direct, et des correctifs pour Cline, Cursor et macOS.

- Les changements de configuration d'un poste sont signalés au moment où ils se produisent.
- Une session coéquipière de Claude Code est rattachée à la session qui la dirige.
- Les skills, règles et serveurs MCP que chargent Codex CLI, Cursor et Cline sont inventoriés tels que chaque agent les charge réellement.
- Les rapports de Cursor nomment le serveur MCP à chaque appel.
- La mise à jour de Cline.app n'interrompt plus une conversation.
- Une session Claude Code lancée à partir d'un plan enregistre ce plan comme premier prompt.
- Sous macOS, le client ne demande plus l'accès aux dossiers Bureau, Téléchargements et Documents.
- `openlatch init` ne présente plus la télémétrie d'usage comme anonyme.
- Les chemins Windows natifs sont correctement classés lors des lectures et écritures de fichiers.

[@openlatch/client@0.6.9](#)

Isolation des organisations appliquée par la base de données

6 octobre 2026 · Plateforme

La sécurité au niveau des lignes est activée et imposée sur chaque table d'organisation, sans propriétaire à l'exécution ni rôle de contournement, et elle est vérifiée en CI et à chaque démarrage. Il s'agit d'un contrôle d'ingénierie, pas d'une attestation.

Client 0.6.8

6 octobre 2026 · Client

Correctifs du bundle de politiques et du relais.

- Un bundle de politiques en cache est correctement revalidé lorsque la plateforme le signale inchangé.
- L'usage des sessions qu'aucun hook n'a enregistrées n'est plus consigné.

[@openlatch/client@0.6.8](#)

Client 0.6.7

5 octobre 2026 · Client

Une évaluation qui tient compte des chemins, et des correctifs pour Cline.

- Les effets d'une action sont mis en relation avec les chemins qu'elle cible.
- Le plugin Cline s'applique à tous les outils, et pas seulement au shell.
- `openlatch init` attend la première livraison du plugin Cline au lieu d'échouer.
- Chaque action est décidée une seule fois, et non une seconde fois après son exécution.
- Les fils secondaires de Codex CLI sont rangés sous leur session parente.
- L'usage est rattaché à la bonne installation.

[@openlatch/client@0.6.7](#)

Des politiques qui lisent ce que fait une action

5 octobre 2026 · Plateforme

Chaque action est classée selon son effet et sa cible, quel que soit l'outil de l'agent connecté qui la porte : une lecture de secrets, un appel à un hôte réseau, une écriture dans une classe de chemins. Une politique écrite une fois s'applique au même effet d'un agent à l'autre.

Client 0.6.6

2 octobre 2026 · Client

Les appels de modèle de l'IDE Cursor passent par le relais.

- Les appels de modèle de l'IDE Cursor sont acheminés par le relais.
- Un nouvel `openlatch init` n'échoue plus sur des composants encore en cours de démarrage.
- `openlatch doctor` affiche une ligne d'état par agent et reconnaît le hub de Cline.app.
- Une substitution de processus du shell n'est plus lue comme une cible de redirection.

[@openlatch/client@0.6.6](#)

Client 0.6.5

1 octobre 2026 · Client

Un vocabulaire d'outils commun, et des verdicts Ask livrés selon chaque agent.

- Les outils de chaque agent sont rattachés à un vocabulaire commun ; un outil sans correspondance est tout de même enregistré sous son propre nom.
- L'évaluation des politiques lit un catalogue de classification unique depuis le bundle de politiques.
- Les verdicts Ask parviennent à chaque agent sous la forme qu'il prend en charge, avec un verdict défini à l'avance lorsqu'un agent n'a pas de ask propre.

[@openlatch/client@0.6.5](#)

Client 0.6.4

1 octobre 2026 · Client

Prise en charge de Cursor.

- Les politiques décident sur les actions de Cursor, et le verdict parvient à Cursor dans son propre format.
- Le trafic de modèle de Cursor passe par le relais et est enregistré comme usage, jamais comme contenu.
- Le raccordement de Cursor est vérifié, réparé et inventorié comme celui de tous les autres agents.
- `openlatch uninstall` nomme les agents qu'une désinstallation partielle laisse connectés.

[@openlatch/client@0.6.4](#)

Client 0.6.3

30 septembre 2026 · Client

Le plugin Cline parvient à Cline.app et à VS Code.

- Le plugin Cline est livré à Cline.app et à VS Code.
- Le client vérifie que le plugin s'est chargé sur chaque hôte.

[@openlatch/client@0.6.3](#)

Licence et consommation dans la console

29 septembre 2026 · Plateforme

La page de licence des Paramètres compare ce pour quoi votre organisation est licenciée à ce qu'elle consomme, avec les dates du contrat, l'historique quotidien et la liste des hôtes, que vous pouvez exporter comme rapport de déploiement.

Propriété des agents

29 septembre 2026 · Plateforme

Chaque agent est rattaché à un propriétaire : une personne nommée, un responsable de service déclaré, ou Non attribué avec sa raison. Le changement de propriétaire est consigné dans la piste d'audit, et un agent qui tourne encore sous l'identité d'un employé parti est signalé.

Des agents reconnaissables

29 septembre 2026 · Plateforme

Chaque carte d'agent affiche un nom compréhensible et le nombre de sessions et de projets qu'elle regroupe. Filtrez les agents, l'inventaire et les événements par dépôt, et repérez quand une même machine est devenue plusieurs lignes d'agents.

Graphe d'exécution

29 septembre 2026 · Plateforme

L'onglet Graphe d'exécution représente les appels de modèle, outils, commandes, fichiers et appels réseau d'un agent sur une période. Filtrez-le sur les chemins les plus coûteux ou les plus risqués, ouvrez n'importe quel nœud pour voir les événements qui le composent, et créez une règle en mode Monitor à partir d'un nœud de commande, rejouée d'abord sur la période.

Économie des agents

29 septembre 2026 · Plateforme

L'onglet Économie de chaque agent présente les dépenses, une prévision et la répartition par session sur la période choisie. Les pistes d'économies sont classées par impact estimé ; elles constituent un diagnostic et ne modifient rien d'elles-mêmes.

Client 0.6.2

29 septembre 2026 · Client

Remontée des données de Cline, réparation automatique et purge complète.

- Les sessions, prompts et appels d'outils de Cline sont transmis à la plateforme.
- Le câblage du relais, la confiance des hooks et les fichiers du plugin Cline sont contrôlés contre toute altération et réparés.
- Le client signale lorsqu'un bundle de politiques est en cours de livraison.
- `openlatch uninstall` envoie un événement de désinstallation, et `openlatch uninstall --purge` supprime tout l'état du client.

[@openlatch/client@0.6.2](#)

Client 0.6.1

29 septembre 2026 · Client

Cline via le relais, et gestion du certificat du relais.

- Le trafic de modèle de Cline est capté par le relais, y compris l'application de bureau Cline.
- Le relais prouve que son certificat est approuvé avant qu'un agent ne soit dirigé vers lui.
- Le certificat du relais peut être consulté, renouvelé, libéré et supprimé.
- Seuls les agents détectés figurent dans les informations sur le poste.
- Les hooks de Codex CLI sont approuvés sous Windows.

[@openlatch/client@0.6.1](#)

Des politiques qui continuent de décider hors ligne

29 septembre 2026 · Plateforme

Les modifications de politiques parviennent à chaque client lors de sa prochaine interrogation, sans nouvelle version du client ni action sur chaque poste. Chaque client garde le dernier bundle reçu et continue de décider avec lui lorsque le réseau est indisponible. En mode Monitor, ce qu'une politique aurait bloqué est enregistré avec l'événement.

Client 0.6.0

24 septembre 2026 · Client

Un `openlatch update` étape par étape, avec un changement incompatible dans sa sortie JSON.

- Changement incompatible : `openlatch update --check --json` ne renvoie plus `min_supported_client.current`, `latest` et `severity` sont inchangés.
- `openlatch update` affiche sa progression étape par étape, et le daemon redémarre ensuite.
- Les hooks, règles et workflows de Cline sont remontés par l'inventaire de configuration.

[@openlatch/client@0.6.0](#)

Client 0.5.8

23 septembre 2026 · Client

Capture des prompts de Cline.

- Les prompts sont captés sur la CLI de Cline et sur l'application de bureau Cline.
- Les types d'agents non connectés ne sont plus signalés.

[@openlatch/client@0.5.8](#)

Client 0.5.7

23 septembre 2026 · Client

Correctif de l'affichage du terminal sous Windows.

- Les couleurs et la progression s'affichent correctement sous Windows pour toutes les commandes, et pas seulement `openlatch init`.

[@openlatch/client@0.5.7](#)

Client 0.5.6

23 septembre 2026 · Client

Maintenance de la publication.

- Changements dans la façon dont le client est construit et publié. Le client se comporte comme dans la version précédente.

[@openlatch/client@0.5.6](#)

Client 0.5.4

22 septembre 2026 · Client

Correctifs de la supervision du daemon et de l'envoi de la télémétrie.

- Sous Windows, la tâche planifiée qui maintient le daemon en marche conserve son marqueur de version.
- Le premier lot de télémétrie du daemon est envoyé en HTTPS.

[@openlatch/client@0.5.4](#)

Client 0.5.3

22 septembre 2026 · Client

La télémétrie passe par la plateforme, et les binaires Linux démarrent sur des hôtes plus anciens.

- La télémétrie et les rapports de plantage passent par le point d'ingestion de la plateforme.
- Les binaires Linux démarrent sur des hôtes minimaux et plus anciens.

[@openlatch/client@0.5.3](#)

Client 0.5.2

21 septembre 2026 · Client

Correctifs du démarrage et de l'envoi des événements.

- Sous Windows, openLatch s'exécute comme un programme console, et ses binaires démarrent sur un poste neuf.
- Les événements encore en file d'attente à l'arrêt sont envoyés.
- Le daemon prend en compte une nouvelle connexion sans redémarrer.
- L'envoi des événements reprend après chaque reconnexion à la plateforme.

[@openlatch/client@0.5.2](#)

Client 0.5.1

20 septembre 2026 · Client

Maintenance de la publication.

- Changements dans la façon dont le client est construit et publié. Le client se comporte comme dans la version précédente.

[@openlatch/client@0.5.1](#)

Client 0.4.0

8 septembre 2026 · Client

Les commandes des sous-systèmes passent sous `openlatch system`.

- Changement incompatible : `openlatch auth`, `hooks`, `model-relay`, `proxy`, `supervision`, `telemetry` et `inventory` deviennent `openlatch system <name>`, sans alias.
- Changement incompatible : la section de configuration `[boundary]` devient `[model_relay]`, les variables `OPENLATCH_BOUNDARY_*` deviennent `OPENLATCH_MODEL_RELAY_*`, et les codes d'erreur `OL-BND-*` deviennent `OL-RELAY-*`.
- `openlatch doctor` indique l'état en direct de chaque hook.
- Les diagnostics de proxy indiquent le proxy que le client utilise réellement.

[@openlatch/client@0.4.0](#)

Client 0.3.3

6 septembre 2026 · Client

Maintenance de la publication.

- Changements dans la façon dont le client est construit et publié. Le client se comporte comme dans la version précédente.

[@openlatch/client@0.3.3](#)

Client 0.2.2

28 août 2026 · Client

Maintenance de la publication.

- Changements dans la façon dont le client est construit et publié. Le client se comporte comme dans la version précédente.

[@openlatch/client@0.2.2](#)

Client 0.1.18

19 août 2026 · Client

Maintenance de la publication.

- Changements dans la façon dont le client est construit et publié. Le client se comporte comme dans la version précédente.

[@openlatch/client@0.1.18](#)

Client 0.1.14

23 avril 2026 · Client

Correctif de l'adresse d'API par défaut.

- L'adresse d'API par défaut ne répète plus /api.

[@openlatch/client@0.1.14](#)

Client 0.1.13

23 avril 2026 · Client

Le paquet npm se résout avec `npx @openlatch/client init`.

- Le paquet npm gagne un point d'entrée `client`, si bien que `npx @openlatch/client init` se résout.
- L'aide n'affiche plus de bannière.

[@openlatch/client@0.1.13](#)

Client 0.1.12

23 avril 2026 · Client

Maintenance de la publication.

- Changements dans la façon dont le client est construit et publié. Le client se comporte comme dans la version précédente.

[@openlatch/client@0.1.12](#)

Client 0.1.8

23 avril 2026 · Client

Maintenance de la publication.

- Changements dans la façon dont le client est construit et publié. Le client se comporte comme dans la version précédente.

[@openlatch/client@0.1.8](#)

Client 0.1.7

23 avril 2026 · Client

Le paquet npm fournit le point d'entrée du hook.

- Le paquet npm installe le point d'entrée openlatch-hook.

[@openlatch/client@0.1.7](#)

Client 0.1.6

23 avril 2026 · Client

Maintenance de la publication.

- Changements dans la façon dont le client est construit et publié. Le client se comporte comme dans la version précédente.

[@openlatch/client@0.1.6](#)