API reference

The management API: OAuth client credentials, scopes, endpoints.

Overview

API

Everything the Admin Console does is an API — the console is just a client.

Authentication

Create an **API credential** in the console (**API Credentials** in the left nav): you get a client ID/secret for the OAuth 2.0 **client_credentials** grant. Each credential carries explicit **scopes**, an optional IP allow-list, and an expiry; every use is attributable in the audit log.

```
curl -X POST https://<your-console-host>/api/oauth/token \
  -d grant_type=client_credentials \
  -d client_id=... -d client_secret=... -d scope="audit:read"
```

Typical uses

- Pull audit events into your own tooling (or use the SIEM export).
- Manage policy (MCP rules, detector actions) from infrastructure-as-code.
- Automate provider-key rotation.

Conventions

JSON in/out; standard HTTP statuses; errors carry a human-readable message. Credentials are revocable instantly from the console.

Create API credentials

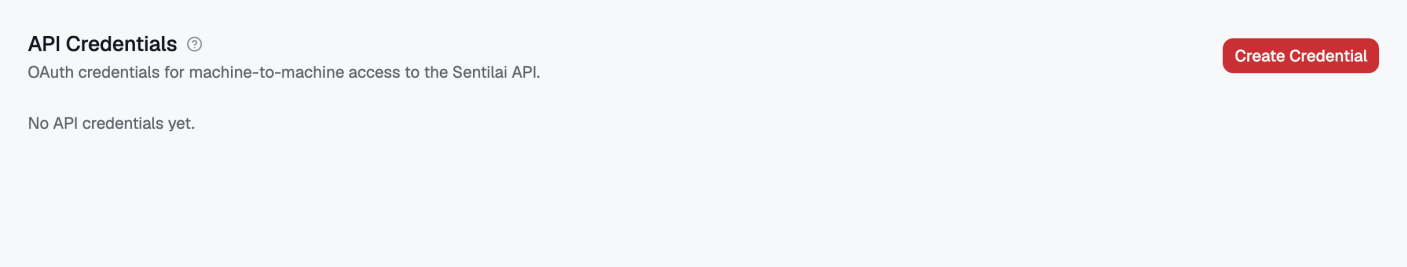
API Credentials issues OAuth client-credential pairs for machine-to-machine access — your own automation reading the audit trail, or exporting policy.

1. **Create credential**, name it for the thing that will use it.
2. Choose **scopes** — read-only ones like `audit:read` where possible.
3. Optionally restrict it to an **IP allowlist** and set an expiry.
4. The **secret is shown exactly once.** Store it in your secret manager before closing the dialog.

Rotating and revoking

- **Reissue secret** generates a new secret; the old one stops working immediately.
- **Revoke** disables the credential for good. It can't be undone — create a new one.

Both actions ask for confirmation, and name the credential in the prompt, because both break whatever is currently using it.



API Credentials — create a scoped, expiring credential. The secret is shown exactly once.

Authenticating to the API

Sentilai's API uses OAuth 2.0 **client credentials**. You create a credential in the console, exchange it for a token, and call the API with the token.

Creating a credential

API Credentials ? Create Credential:

- **Name** — after the system that will use it, not the person creating it.
- **Scopes** — see Scopes and least privilege.
- **Expires in** — 30, 90 or 180 days, or a year.
- **IP allowlist / denylist** — one entry per line or comma-separated; CIDR ranges work.

The **client secret is shown exactly once**, in a dialog that asks you to confirm you have saved it. There is no way to retrieve it afterwards — only to reissue, which invalidates the old one.

Using it

Exchange the client id and secret for an access token at `POST https://admin.sentilai.com/api/oauth/token`, then send it as a bearer token. The request body is **form-encoded** (standard OAuth), not JSON:

```
curl -X POST https://admin.sentilai.com/api/oauth/token \
  -d "grant_type=client_credentials" \
  -d "client_id=cid_..." \
  -d "client_secret=..."
```

The response carries `access_token` (valid for an hour) and `token_type: Bearer`. Send it as `Authorization: Bearer <token>`. The token carries your credential's scopes; a call outside them is refused.

Rotating

Reissue secret generates a new secret and the old one stops working immediately. Plan for a moment of downtime in whatever uses it, or create a second credential, migrate, and revoke the first.

Revoke is permanent.

Expiry

Set one. An expiring credential forces a rotation you would otherwise never do, and the console shows the expiry date in the table so it does not surprise you.

The **Last used** column tells you which credentials are actually in service — "never" is usually a credential someone created, mislaid the secret for, and quietly recreated.

Create a credential

The secret is shown once, right after creation — it can't be retrieved again.

Name

e.g. CI pipeline

Scopes

- [ ] `audit:read`
- [ ] `risk:read`
- [ ] `policy:read`
- [ ] `policy:write`
- [ ] `users:read`
- [ ] `users:write`
- [ ] `gateway:invoke`

Expires in

90 days

IP allowlist (optional — one per line or comma-separated)

e.g. 203.0.113.0/24

IP denylist (optional)

Cancel Create

The create-credential dialog — name, scopes, expiry and IP restrictions. The secret is shown once, right after creation.

Scopes and least privilege

Seven scopes, granted per credential.

Scope	What it allows
`audit:read`	Read audit events — the SIEM pull endpoint
`risk:read`	Read risk findings
`policy:read`	Read your policy configuration
`policy:write`	Change your policy configuration
`users:read`	Read admins and developers
`users:write`	Invite, approve and offboard people
`gateway:invoke`	Send AI requests through the Gateway

Choosing

Grant the minimum. Three patterns cover almost everything:

- **A SIEM integration** — `audit:read` only.
- **A compliance or reporting script** — `audit:read`, `risk:read`, `policy:read`.
- **Provisioning automation** — `users:read`, `users:write`.

`policy:write` deserves particular care: a credential holding it can turn your enforcement off. If something needs it, restrict that credential by IP as well.

IP restrictions

The allowlist and denylist are cheap and effective. A credential that only ever calls from your SIEM's egress address should say so — a leaked credential is then also useless.

`gateway:invoke`

This is what a tool needs to send AI requests through the Gateway. The Endpoint Suite manages this for developers automatically; you only need it explicitly for something you are building yourself.

Create a credential

The secret is shown once, right after creation — it can't be retrieved again.

Name

e.g. CI pipeline

Scopes

- [ ] `audit:read`
- [ ] `risk:read`
- [ ] `policy:read`
- [ ] `policy:write`
- [ ] `users:read`
- [ ] `users:write`
- [ ] `gateway:invoke`

Expires in

90 days

IP allowlist (optional — one per line or comma-separated)

e.g. 203.0.113.0/24

IP denylist (optional)

Cancel Create

The seven scopes, chosen per credential at creation. A SIEM integration needs exactly one: audit:read.

Gateway endpoints

What the Gateway exposes, for when you are integrating something yourself rather than using the Endpoint Suite.

Chat endpoints

Path	Shape
`POST /v1/messages`	Anthropic — Claude Code
`POST /v1/chat/completions`	OpenAI — Cursor, Copilot
`POST /azure/openai/deployments/{deployment}/chat/completions`	Azure OpenAI
`POST /gemini/v1beta/models/{model}`	Gemini

They mirror the upstream providers' own shapes, which is what makes governing existing tools possible without modifying them.

Supporting endpoints

- `GET /v1/models` — an OpenAI-compatible model list. VS Code Copilot's custom-endpoint feature probes this to populate its dropdown. It is a curated list, not a statement of what your organization is entitled to.
- `GET /v1/status` — per-provider health.
- `GET /healthz` — public liveness.

MCP endpoints

- `POST /mcp/{alias}` — a routed remote MCP server, registered on MCP Inventory.
- `POST /mcp/decide` — the policy decision the Endpoint Suite's shim asks for.
- `POST /mcp/observations` — the shim reporting observed tool schemas and protocol versions.

The public Observatory

`GET /v1/slopsquat/observatory` needs no authentication. It returns the anonymized feed of commonly hallucinated package names. Use it in your own supply-chain checks if it is useful.

Two ways to authenticate

A normal `Authorization: Bearer` or `x-api-key` header, **or** a device token embedded in the path. The Endpoint Suite uses the second form, because it can be written into a tool's base URL setting where there is nowhere to put a header.

Gateway error codes

What the Gateway returns when something is wrong, and what each one means for you.

Code	Meaning	What to do
401	No token, or an invalid one	The device may be revoked or its session expired — sign in again in the Endpoint Suite
403	The token lacks `gateway:invoke`	Grant the scope, or use a credential that has it
409	Ambiguous provider key — the model is in more than one key's enabled-models list	Make the model lists disjoint on **Providers**
424	No provider key configured for this request	Add a matching key on **Providers**, or move the tool to subscription mode

Upstream errors

Errors from the AI provider itself are passed through unchanged and appear in **Activity** with the **Upstream error** outcome. Rate limits, provider outages and model deprecations all arrive this way. The provider's own message tells you which.

Blocked is not an error

A request stopped by your policy is not a failure of the Gateway. It shows in Activity as **Blocked**, with the rule that stopped it in the Signals column. If a developer reports "an error", check the outcome column before assuming something is broken — it may be working exactly as configured.

Revocation timing

Revoking a device, or a device session expiring, takes effect within roughly 30 seconds rather than instantly. The Gateway notices at the next token refresh.