Endpoint Suite

The desktop app: zero-click governing, updates, MDM, restrictions.

- Overview
- The Tools tab
- The MCP tab
- The Account tab
- Running in the background
- macOS and Windows differences
- Deploying with MDM

Overview

Sentilai Endpoint (desktop app)

The Sentilai Endpoint is the small desktop app that makes governance zero-touch: it discovers the AI tools on the machine, routes them through the Gateway, wraps local MCP servers, and reports device compliance.

Install & sign in

Download the signed installer from your invite or the downloads page, drag to Applications (macOS) or run the setup (Windows), open, and sign in with your work identity (passkey or your company SSO). The device registers automatically and gets its own credential.

Zero-click governing

With **Automatic governing** on (the default), the app governs every detected tool on launch — no clicks. Admins can pin this on for the whole fleet via managed configuration.

Updates

The app updates itself: it checks for signed updates, downloads in the background, and applies on restart. Versions are cryptographically verified end-to-end.

Managed configuration (MDM)

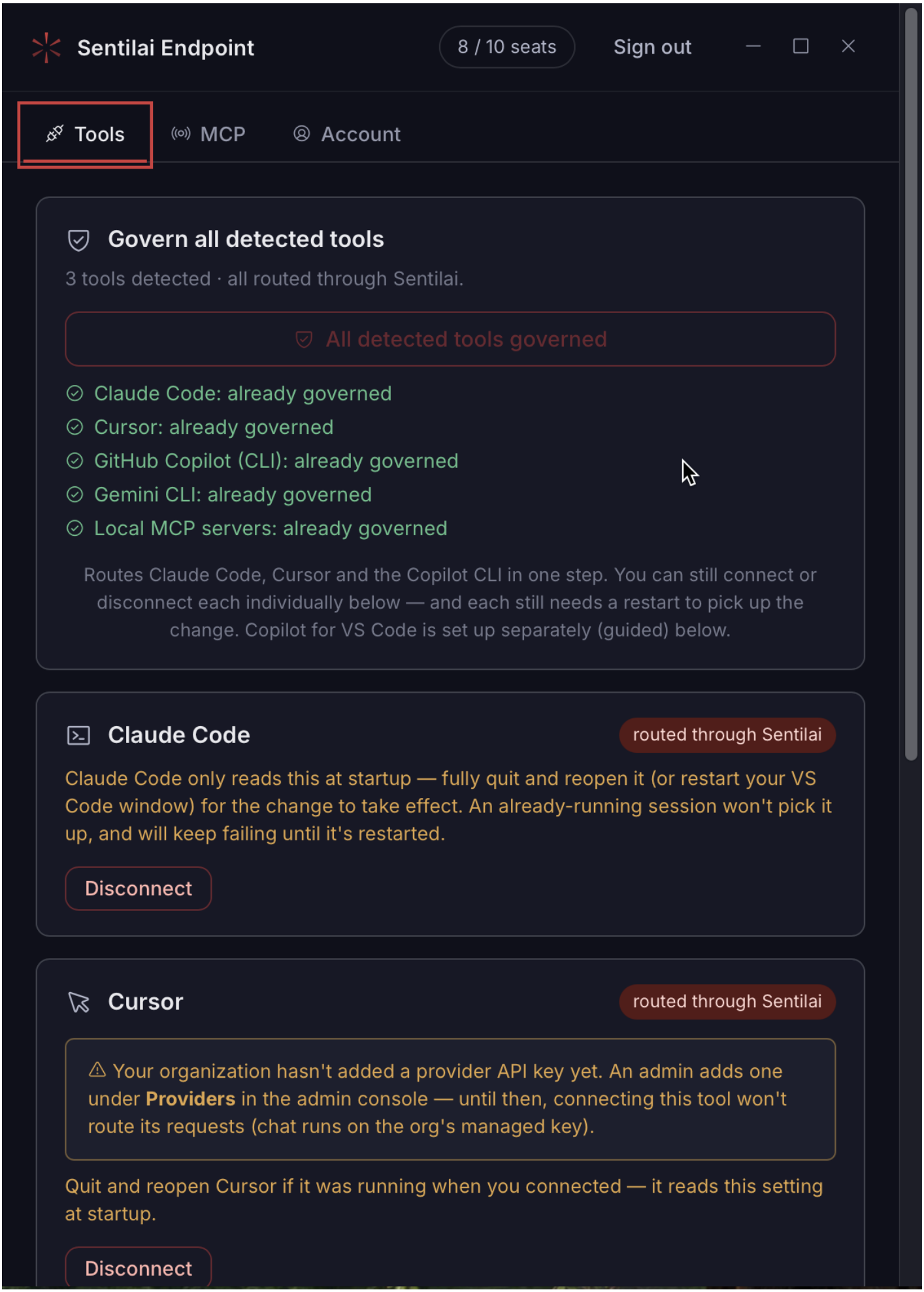
Fleet deployments can pre-set behavior via a managed config file (`/Library/Application Support/Sentilai/managed-config.json` on macOS, `C:\ProgramData\Sentilai\managed-config.json` on Windows): environment, autostart, pinned auto-govern, and **user restrictions** — disable disconnect, sign-out, or un-governing MCP servers, so a developer cannot quietly step outside governance.

Device compliance

Each device reports whether every installed tool is actually governed. Non-compliant devices are visible in the Admin Console's Devices screen, so drift (e.g. a manually removed setting) surfaces instead of hiding.

Local event log

The app keeps a local, exportable event log. Admins can request a device's logs from the console; with user-visible consent flows, support can be granted access for troubleshooting.


Sentilai Endpoint
8 / 10 seats
Sign out
Tools
MCP
Account
Govern all detected tools
3 tools detected · all routed through Sentilai.
All detected tools governed
Claude Code: already governed
Cursor: already governed
GitHub Copilot (CLI): already governed
Gemini CLI: already governed
Local MCP servers: already governed
Routes Claude Code, Cursor and the Copilot CLI in one step. You can still connect or disconnect each individually below — and each still needs a restart to pick up the change. Copilot for VS Code is set up separately (guided) below.
Claude Code
routed through Sentilai
Claude Code only reads this at startup — fully quit and reopen it (or restart your VS Code window) for the change to take effect. An already-running session won't pick it up, and will keep failing until it's restarted.
Disconnect
Cursor
routed through Sentilai
⚠ Your organization hasn't added a provider API key yet. An admin adds one under Providers in the admin console — until then, connecting this tool won't route its requests (chat runs on the org's managed key).
Quit and reopen Cursor if it was running when you connected — it reads this setting at startup.
Disconnect

The Endpoint Suite: three tabs, the seat counter, and every AI tool it found on this machine.

The Tools tab

Everything about which AI tools on this machine are governed.

Tool cards

One card per supported tool, each showing one of three states:

- **routed through Sentilai** — governed; its requests appear in the admin console.
- **detected** — installed, but still talking directly to the AI provider.
- **not detected** — not installed, or not where the app looks for it.

Each detected tool has **Connect** and, once governed, **Disconnect**.

Govern all detected tools

The card at the top connects everything detected in one action and reports the result per tool. See Govern all detected tools.

The provider-key warning

If your organization has not added a provider key yet, a warning sits above the tools: connecting a tool that runs in managed mode will not route its requests anywhere useful. Your admin adds the key on **Providers** in the console.

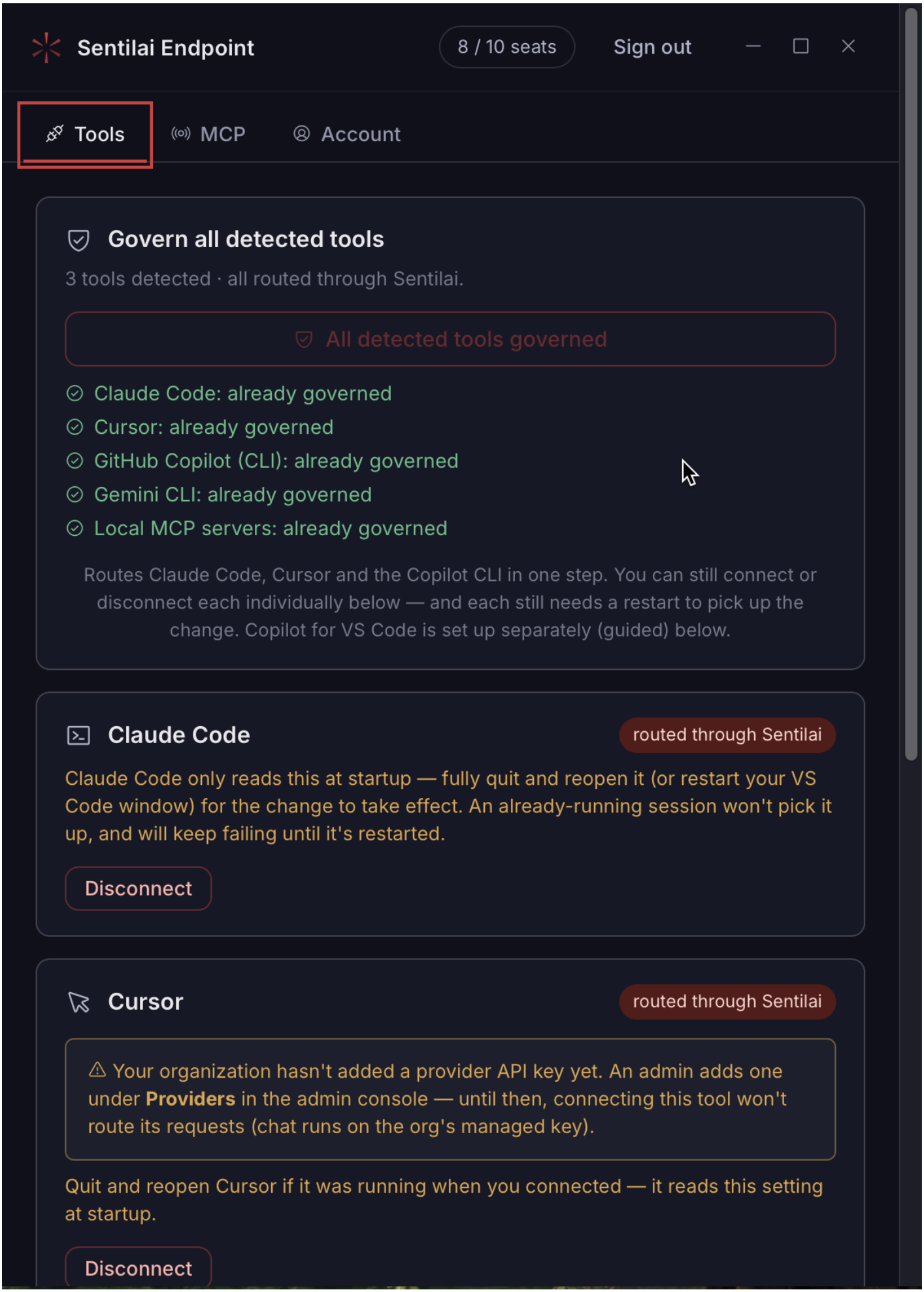
Restart reminders

Every card that needs it says so. All of these tools read their configuration at startup only, so a connect without a restart changes nothing.

When buttons are missing

If your organization deploys a managed configuration, it can prevent disconnecting, signing out, or ungoverning MCP servers. Where that applies, the button is absent and the card says the setting is managed by your

organization.


Sentilai Endpoint
8 / 10 seats
Sign out
Tools
MCP
Account
Govern all detected tools
3 tools detected · all routed through Sentilai.
All detected tools governed
Claude Code: already governed
Cursor: already governed
GitHub Copilot (CLI): already governed
Gemini CLI: already governed
Local MCP servers: already governed
Routes Claude Code, Cursor and the Copilot CLI in one step. You can still connect or disconnect each individually below — and each still needs a restart to pick up the change. Copilot for VS Code is set up separately (guided) below.
Claude Code
routed through Sentilai
Claude Code only reads this at startup — fully quit and reopen it (or restart your VS Code window) for the change to take effect. An already-running session won't pick it up, and will keep failing until it's restarted.
Disconnect
Cursor
routed through Sentilai
⚠ Your organization hasn't added a provider API key yet. An admin adds one under Providers in the admin console — until then, connecting this tool won't route its requests (chat runs on the org's managed key).
Quit and reopen Cursor if it was running when you connected — it reads this setting at startup.
Disconnect

The Tools tab — one card per detected tool, each with its own connect/disconnect and the restart it needs.

The MCP tab

Model Context Protocol servers are the tools your AI assistant can call — file access, issue trackers, databases, browsers. They are also the most direct route from a prompt to something happening in the real world, which is why they get their own governance.

Local MCP governance

The card at the top shows how many local servers are governed. **Govern local MCP servers** puts them behind the Sentilai shim.

Before it edits anything it **backs up the configuration file** and it reverts exactly that file if you stop governing. Claude Code must be restarted afterwards.

Stop governing restores the backup — unless your organization has locked this down.

The server list

Every MCP server found on this machine, with its name, its project path, and how it is started. Servers configured in Cursor are listed separately, because Cursor keeps its own configuration.

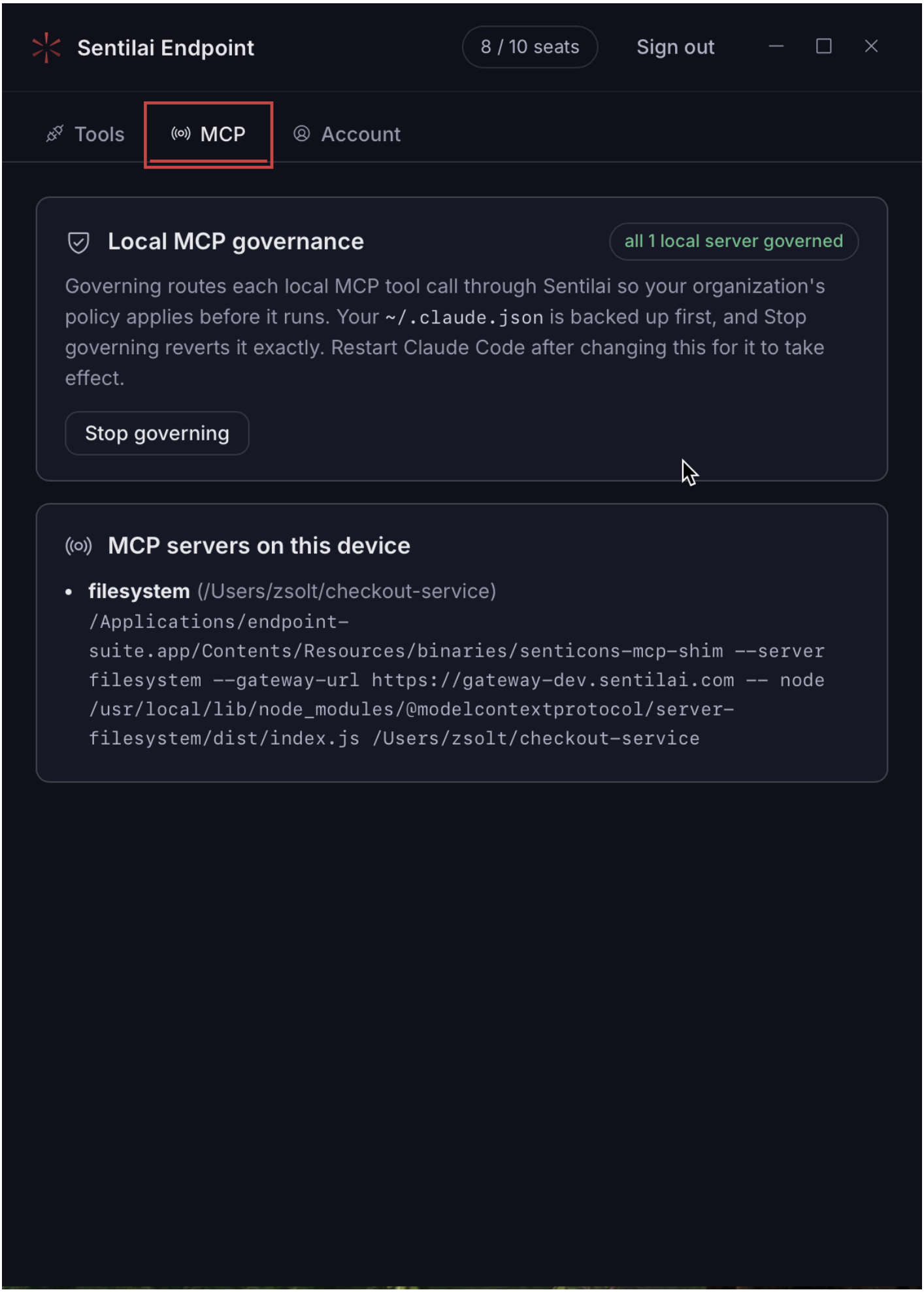
"Scanning…" means the sweep is still running; "None found" means this machine has no local MCP servers configured.

What the shim actually does

Once governed, an AI tool calling an MCP server goes tool ? Sentilai shim ? real server. The shim only inspects **tool calls**; everything else passes through untouched.

For each call it asks the Gateway for a decision and caches the answer for as long as your admin's **Local MCP decision cache** setting allows. When a call is blocked, the agent gets a clean error naming the server and tool, so the assistant can tell the developer what happened instead of failing mysteriously.

The shim also watches the server's advertised tool list, which is what makes it possible to notice when a tool's description or schema changes underneath you.


Sentilai Endpoint
8 / 10 seats
Sign out
Tools
MCP
Account
Local MCP governance
all 1 local server governed
Governing routes each local MCP tool call through Sentilai so your organization's policy applies before it runs. Your ~/.claude.json is backed up first, and Stop governing reverts it exactly. Restart Claude Code after changing this for it to take effect.
Stop governing
MCP servers on this device
filesystem (/Users/zsolt/checkout-service)
/Applications/endpoint-suite.app/Contents/Resources/binaries/senticons-mcp-shim --server filesystem --gateway-url https://gateway-dev.sentilai.com -- node /usr/local/lib/node_modules/@modelcontextprotocol/server-filesystem/dist/index.js /Users/zsolt/checkout-service

The MCP tab — local MCP governance, and every MCP server this device has been seen using.

The Account tab

Your session, this device, and the app's own behaviour.

Signed in

Shows who you are signed in as and your organization's seat usage.

Signing out ends the session only. Your governed tools keep pointing at the Gateway and start failing closed rather than quietly reverting to the direct provider. This is deliberate: silently ungoverning a machine because someone signed out would be the worst possible failure mode for a governance product. Removing the device entirely is an admin action, from the console.

Device identity

The ECDSA P-256 key pair for this machine, stored in the OS secure storage. Not exportable, not copyable to another machine.

Automatic governing

On by default. New tools and new local MCP servers are governed in the background as they appear, and tools that drift back to the direct provider are re-governed.

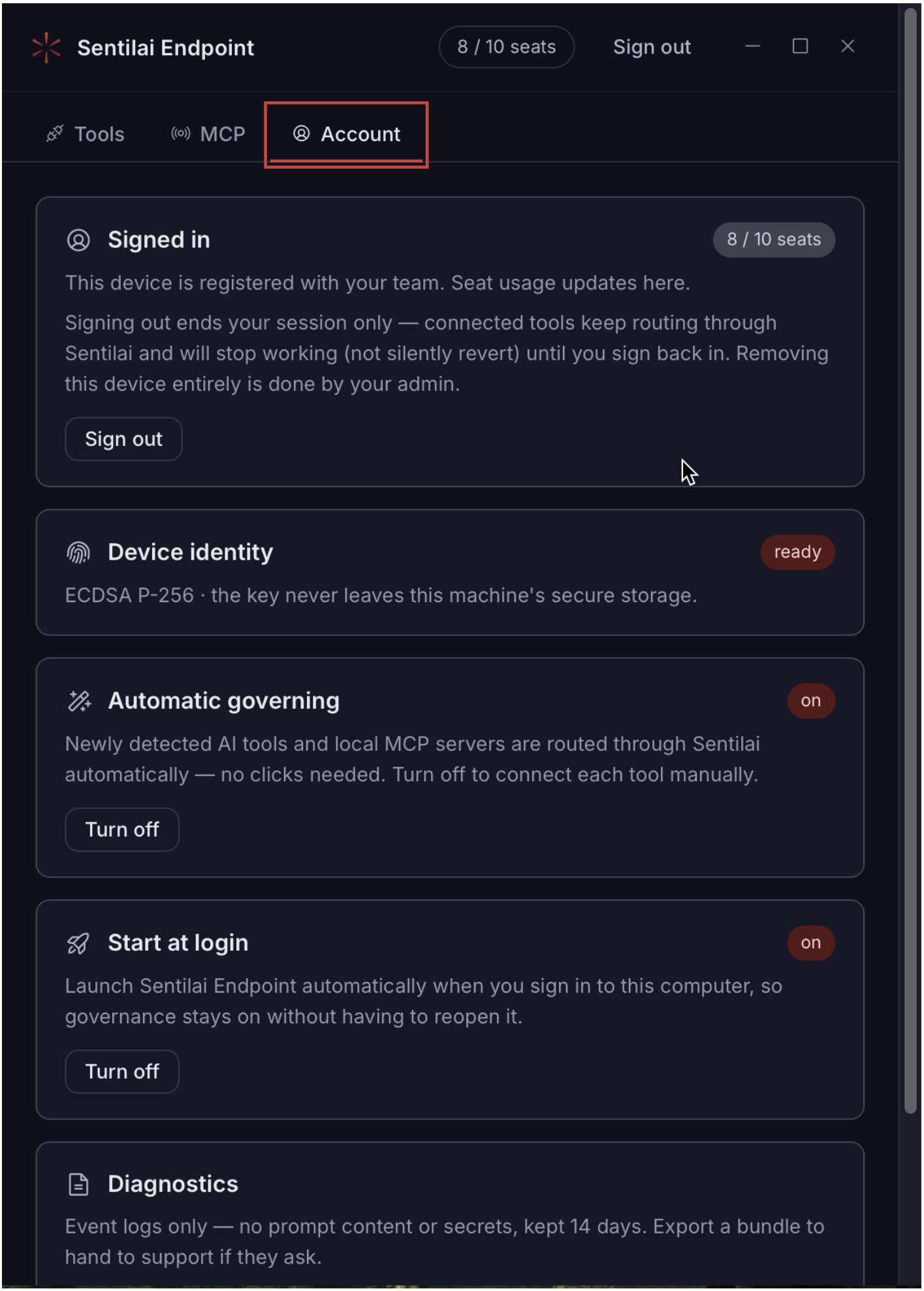
Start at login

Whether the app launches when you sign in to the machine. Both this and automatic governing can be pinned by a managed configuration, in which case they show as managed by your organization.

Diagnostics

Export logs writes a bundle and shows you where it is. **Open logs folder** opens the directory.

These are **event logs only — no prompt content and no secrets** — and they are kept for 14 days. This is the same material an admin receives when they request logs from your device.


Sentilai Endpoint
8 / 10 seats
Sign out
Tools
MCP
Account
Signed in
8 / 10 seats
This device is registered with your team. Seat usage updates here.
Signing out ends your session only — connected tools keep routing through Sentilai and will stop working (not silently revert) until you sign back in. Removing this device entirely is done by your admin.
Sign out
Device identity
ready
ECDSA P-256 · the key never leaves this machine's secure storage.
Automatic governing
on
Newly detected AI tools and local MCP servers are routed through Sentilai automatically — no clicks needed. Turn off to connect each tool manually.
Turn off
Start at login
on
Launch Sentilai Endpoint automatically when you sign in to this computer, so governance stays on without having to reopen it.
Turn off
Diagnostics
Event logs only — no prompt content or secrets, kept 14 days. Export a bundle to hand to support if they ask.

The Account tab — the signed-in session, this device's identity key, and automatic governing.

Running in the background

The app is designed to be invisible once set up.

The system tray

Closing the window hides the app to the tray rather than quitting it. The tray menu has **Show Sentilai Endpoint** and **Quit**; left-clicking the icon reopens the window.

The connectivity badge

If a host the app needs is unreachable, the tray icon gains an amber badge and the window shows a banner naming the host. It keeps retrying — this is usually a VPN that has just come up, or a firewall rule that has not been widened yet.

Your network team may need the hostnames; see Hostnames and firewall rules.

Automatic updates

The app checks for updates, verifies the download's signature before installing it, and replaces itself. Verified on macOS; on Windows the updater ships but has not yet been proven end to end, so treat Windows updates as manual for now.

What it does periodically

- **Adapter drift check** — is each governed tool still pointing at the Gateway? A tool that has drifted is re-governed if automatic governing is on, and either way the console shows the device as **Non-compliant** .
- **Log request poll** — roughly every two minutes, so that an admin's request for logs is fulfilled promptly.
- **Posture scan** — read-only inspection of your AI tools' own autonomy settings, sent to the console as posture findings. It reads configuration, never prompts or code.



The Suite lives in the menu bar. Closing the window does not stop it — governance keeps running.

macOS and Windows differences

The two platforms are not at the same maturity, and it is better to know where before you roll out.

macOS

- Signed and notarized; no security warning on first launch.
- **Install by dragging to Applications.** Running from the mounted disk image leaves macOS executing it from a translocated read-only path where it cannot keep its device key.
- The device key lives in the **Keychain**.
- Environment changes are applied in the way that actually reaches applications launched from the Dock, not only terminal sessions.
- The app can compare an installed AI application's code-signing identity against the vendor's real one, and flag an imposter. This check is macOS-only.

Windows

- Installs **per user** via an NSIS installer — no administrator rights needed.
- **Not yet Authenticode-signed**, so SmartScreen warns on first run. This is the main outstanding gap on Windows.
- The device key lives in **Credential Manager**.
- Command-line tool detection understands Windows executable extensions, so `claude.cmd` is found the same way `claude` is on macOS.
- Automatic updates ship but are unproven; treat them as manual.
- There is no MSI and no Group Policy deployment path yet — the per-user installer is the only supported route.

Linux

The app builds, but Linux is not a supported distribution track and there is no signed package. If you need it, tell us — it is a matter of demand, not of feasibility.

Deploying with MDM

For fleets, the Endpoint Suite reads a managed configuration file that your device management tool can place on each machine. It pins behaviour so that individual users cannot undo your rollout.

Where the file goes

- **macOS**: under `/Library`, in the location your MDM normally uses for managed app configuration.
- **Windows**: `C:\ProgramData\Sentilai\managed-config.json`.

What you can pin

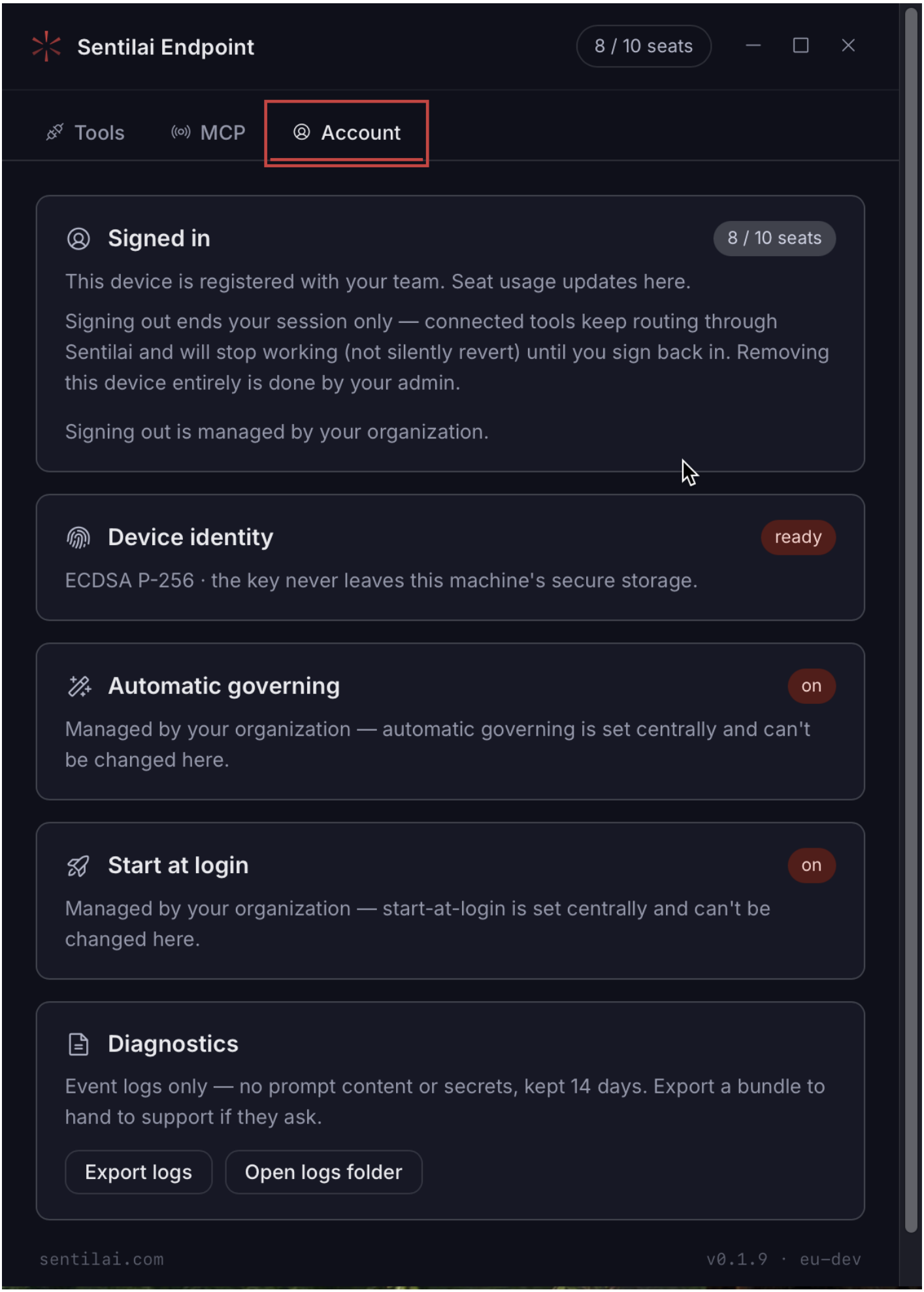
- **Environment** — which Sentilai environment this fleet belongs to.
- **Start at login** — force the app to launch with the session.
- **Automatic governing** — force new tools and MCP servers to be governed as they appear.
- **Restrictions**, three separate switches:
  - `allow_signout` — whether the user may sign out.
  - `allow_disconnect` — whether the user may disconnect a governed tool.
  - `allow_ungovern_mcp` — whether the user may stop governing local MCP servers.

How restrictions behave

Each is enforced in two places: the underlying command refuses, and the button disappears from the interface with an explanation that the setting is managed by your organization. Enforcing only in the UI would be theatre; enforcing only in the command layer would leave buttons that fail confusingly.

A sensible rollout

1. Pilot on a handful of machines with nothing pinned, so people can undo things and tell you what broke.
2. Then pin **start at login** and **automatic governing** — the two that make governance stick without taking anything away.
3. Only pin the restrictions once your policy is stable. A developer who cannot disconnect a tool that is genuinely misbehaving will open a ticket with you instead, so make sure you want that traffic.



The same Account tab on a fleet-managed machine: sign-out, automatic governing and start-at-login are all set

centrally, and the controls are gone rather than disabled.