Reference

Lookup tables: detectors, event types, policy precedence, glossary.

- Glossary
- All policy settings at a glance
- The four policy actions
- What Sentilai does not do

Glossary

Admin — someone who uses the console. Manages policy, people, providers and billing. Consumes a seat.

Agent — an AI assistant that can act, not only answer, by calling tools.

Blocked — a request or tool call your policy stopped. Not an error.

Developer — someone whose AI tools are governed. Uses the Endpoint Suite, never the console. Consumes a seat.

Device — one enrolled machine, identified by a key pair in its secure storage.

Drift — a governed tool that no longer points at the Gateway, because its configuration was changed outside the app. Surfaces as **Non-compliant**.

Endpoint Suite — the desktop app that enrolls a machine and points its AI tools at the Gateway.

Evidence pack — the point-in-time report from **Compliance**.

Gateway — the proxy every governed AI request passes through.

Lethal trifecta — private data access, untrusted content, and an exfiltration channel present together. See Lethal-trifecta enforcement.

Managed API key mode — requests billed to your organization's provider key.

MCP — Model Context Protocol, how an assistant calls tools.

Posture finding — something about an AI tool's *own* autonomy settings on a device. Report-only.

Prompt capture — the opt-in setting that stores conversation content.

Routed endpoint — a remote MCP server reached through the Gateway, so policy applies.

Rug pull — an MCP tool whose description or schema changes after you approved it.

Seat — one person, admin or developer. Shared pool.

Shim — the small process between an AI tool and a local MCP server that asks the Gateway for a decision.

Slopsquatting — registering package names that AI models commonly hallucinate.

Subscription mode — requests billed to the developer's own AI plan.

Ungoverned agent — an autonomous AI system on a device whose traffic does not pass through the Gateway, so it can be detected but not routed, enforced or audited. See Ungoverned AI agents.

All policy settings at a glance

Everything on the **Policy** screen, with its default.

MCP

Setting	Default	Notes
MCP default action	Warn	Allow / Warn / Report / Block. Per-server rules override it
Require approval for new MCP servers	Off	Unreviewed servers get the pending action
Pending action	Warn	Applies only when approval is required
Local MCP decision cache	5 seconds	0–3600 seconds. 0 means ask every time

Risk

Setting	Default
AI risk classifier sensitivity	Medium
Secret scanning	Warn
Credentials in context	Report
Personal data (PII)	Report
Lethal-trifecta enforcement	Warn

Detected secrets are handled by their detector's action regardless of the sensitivity dial.

Data

Setting	Default	Notes
Prompt capture	Off	Redacted before storage; follows retention
Log retention	Set by support	Open a ticket to change it

Devices

Setting	Default	Notes
Idle timeout (hours)	Off	Blank disables
Max age (days)	Off	Blank disables
Ungoverned AI agents	Report it only	Report / Mark non-compliant / Block the device's access
Grace period (hours)	24	Blocking only. 0 blocks immediately

All of these take effect within about 30 seconds, like manual revocation. Blocking on an ungoverned agent is the one setting here that can withdraw a developer's access without an admin acting at the time — see Ungoverned-agent policy before changing it.

Per tool

Connection modes — subscription or managed API key, per tool. Cursor is locked to managed.

Also on this screen

Per-server and per-tool MCP rules, real-time alert channels, the egress blocklist, and the blocked-conversations list.

The four policy actions

Every rule in Sentilai resolves to one of four actions. They mean the same thing wherever they appear.

Allow

Nothing beyond the ordinary audit row. Use it to carve an exception out of a stricter default — for example, a default of Block with explicit Allow rules for the MCP servers you have approved.

Report

Recorded as a finding. Visible in **Activity** and counted in **Compliance**. Nothing is interrupted and the developer sees nothing.

This is the setting you want while you are learning. It gives you the same visibility as blocking without changing anyone's day. Most organizations should spend their first weeks almost entirely in Report.

Warn

Recorded more prominently and, where the path allows it, surfaced to the developer. Useful when you want the person to know without stopping them — a nudge, not a wall.

Block

The request or tool call does not proceed. In Activity the row is tinted with a red rail and its outcome is **Blocked**.

How they combine

- **MCP rules**: the most specific rule wins — tool, then server, then pending, then default.
- **Across servers in one request**: the most restrictive wins. The Gateway cannot partially block a single API call.
- **Detectors**: independent of MCP rules; the strongest action any detector takes determines the outcome.

What Sentilai does not do

Reading a list of a security product's limits is a better use of five minutes than reading its feature list. Here is ours.

It does not cover ungoverned machines

A personal laptop with a personal API key never touches the Gateway and is invisible to Sentilai. What we give you is a governed path that is easy, drift detection when a governed machine slips out of it, and an egress blocklist so your network can close the direct route. That is control of the managed path, not of every possible path.

It does not read your repositories or your CI

Sentilai governs traffic between AI assistants and AI providers, plus the MCP servers those assistants call. It is not a code scanner, not a secrets scanner for your git history, and not a CI gate.

It does not stop a determined insider

Someone who wants to exfiltrate data has simpler routes than an AI assistant. Sentilai raises the floor and creates a record; it is not an insider-threat programme.

Detection is imperfect

The classifier will miss things and will occasionally flag innocent work. The detectors are tuned to be useful rather than exhaustive, which is why the trifecta detector exists — it constrains the *consequence* of an injection rather than relying on recognizing it.

Some paths are only partly covered

- **Cursor** — chat only; Tab autocomplete is not routed. Requests also still transit Cursor's servers.
- **GitHub Copilot** — chat only through the custom endpoint; inline completions are not routed, and requests still transit GitHub's servers.
- **MCP per-tool rules** — enforced where the Gateway knows a tool name: the local shim and routed remote endpoints. The chat-request side-channel supports server-level policy.

We hold no certifications yet

No SOC 2, no ISO 27001 today. When that changes it will be stated here, with the report available. Ask us for the current status; you will get a straight answer.