

23 — TraxAI: Choosing and Connecting a Model

TraxProbe & Traxcope User Manual · Version 2.0.0 · Updated 2026-10-07

TraxAI is the assistant built into Traxcope. It answers questions about your trace — CPU load, a late task, a memory leak, a noisy signal — by calling Traxcope's own analysis tools, and it can build and flash your firmware with your approval. TraxAI does not come with a model of its own: you connect it to one. That can be a cloud provider (OpenAI, Anthropic, OpenRouter), or a model you run yourself — on this machine or on a server in your company — with Ollama, LM Studio, vLLM or llama.cpp.

There are three ways to use it:

Way	The conversation runs in	Needs
TraxAI panel, cloud model	Traxcope, with a provider's model	An API key from the provider; you pay per use.
TraxAI panel, your own model	Traxcope, with a model on this machine or a company server	Ollama, LM Studio, vLLM or llama.cpp. Works offline.
Your AI app	Claude, ChatGPT (Codex), GitHub Copilot, Cursor, Gemini CLI, Windsurf — the app you already use	That app and its subscription; no API key in Traxcope. See Using TraxAI from your AI app.

In every case the tools run in Traxcope, on what Traxcope shows, under your license, and anything that changes something asks you first.

This chapter explains what a model needs to work well with TraxAI, how to connect each kind, and what to do when it does not.

TraxAI is included in the Standard and Professional licenses, and in the trial. It is not part of the free Community Edition (see 04 — Licensing & Activation).

What TraxAI needs from a model

Need	Why
Tool calling (function calling)	TraxAI works by calling more than 40 tools — task statistics, spectra, triggers, the Scope and Trace views. A model that can only chat cannot drive them.
A long context — 32,000 tokens or more	The instructions and the tool list alone take about 21,000 tokens on a Qwen model (more with some other models), and each tool result adds more as a question is worked through.
Streaming	So the answer appears as it is written. Every provider and server in this chapter supports it.

The larger and newer the model, the better it picks the right tool and reads its results. Small local models (around 8 billion parameters and below) often call tools wrongly or stop half way. Use **Test model** (below) to find out before you rely on one.

Settings

Preferences → TraxAI:

Setting	Meaning
Enable TraxAI	Shows the assistant.
Backend	Local (Ollama) — a model run by Ollama on this machine. Cloud or your own server — a cloud provider, or any server that speaks the OpenAI-compatible API (LM Studio, vLLM, llama.cpp, a company gateway).
Provider	Fills the address for a known provider or server. Custom takes any OpenAI-compatible address.
API base URL	The server's address. Fixed for the cloud providers; editable for the server presets and Custom.
Model	Opening the list asks the server which models it offers. You can also type a name.
API key	Kept per provider: a key typed for OpenAI is never sent to another provider or to your own server. Leave it empty for a local server that needs none. On Windows, keys are stored encrypted for your Windows user account (DPAPI). On Linux they are only encoded, not encrypted: keep <code>~/ .config/Embedya/Traxcope.ini</code> readable by you alone (<code>chmod 600</code>).
Context window (Local only)	How much of the conversation the model reads at once. Default 32,768 tokens, never more than the model was trained for. Larger needs more memory.
Let the model think before it answers (Local only)	For models that can reason (Qwen3, DeepSeek-R1). Off by default: thinking can improve an answer but costs minutes on a local machine. Reasoning a model writes into its answer anyway is removed before you see it.
Temperature (under Advanced)	Low values keep answers analytical. Newer cloud models ignore it.
Max tool iterations (under Advanced)	A safety limit on tool calls per question; after it, TraxAI answers with what it has.

Two buttons check the connection:

- **Test connection** — can Traxcope reach the server, and which models does it offer?
- **Test model** — can the selected model drive TraxAI? It asks the model to call one tool with a given value, as TraxAI does all the time. It costs a few hundred tokens and takes a few seconds (a reasoning model may take a minute).
The result:

Result	What it means
✓ <i>works with TraxAI</i>	The model called the tool correctly.
⚠ <i>called the tool with the wrong arguments</i>	It may misuse TraxAI's tools. A stronger model is safer.
✗ <i>answered without calling the tool</i>	The model — or the way the server runs it — cannot use tools. See the server's tool-calling setting below, or choose another model.
✗ <i>(an error)</i>	The provider's reason: a wrong key, a model name that does not exist, no credits left.

Both tests use the fields as they are, before you press OK.

Privacy: where your data goes

When TraxAI answers, what it reads — trace statistics, variable values, parts of your source code it opens — goes to the model. Preferences says where, under the settings:

Address	Where requests go
Local (Ollama), or localhost / 127.0.0.1	Nowhere: the model runs on this machine.
A local-network address (192.168.x.x, 10.x.x.x, a server name without dots)	To that server, not over the internet — unless that server forwards them.
Anything else	Over the internet to that host, under that provider's terms.

Requests to OpenAI's Responses API are sent with storage off, like its chat API.

Cloud providers

Provider	Key from	Notes
OpenAI	platform.openai.com	Models that take tools only on OpenAI's newer Responses API (reasoning models, the -pro models) are switched to it automatically. -pro models can take several minutes per step.
Anthropic (Claude)	console.anthropic.com	Connected through Anthropic's own API.
OpenRouter	openrouter.ai	One key for many providers' models, including Qwen, Llama, Mistral and DeepSeek.

A provider may limit how many tokens a key can use per minute. TraxAI then waits and retries — for up to three minutes — and shows the wait on the status line. A used-up quota or an account with no credits ends the answer at once with the provider's message.

Using TraxAI from your AI app (MCP)

If you already use an AI app — Claude, ChatGPT, GitHub Copilot, Cursor — it can use TraxAI's tools directly, through the Model Context Protocol (MCP). You ask in that app, on your own subscription; the app calls Traxcope's tools, and the answers and views appear as they would from the TraxAI panel. No API key is entered in Traxcope.

Setting it up

Preferences → AI apps (MCP) lists the apps Traxcope knows, whether each is on this computer, and whether Traxcope is already added to it.

1. Tick **Let AI apps on this computer use TraxAI's tools**.
2. Press **Add** next to your app. Traxcope writes its entry into the app's settings (backing the file up first) and says what to do next.

3. Restart the app, or start a new session in it, as the message says.

App	What Add does	Then
Claude Desktop	Adds Traxcope to its MCP servers (claude_desktop_config.json).	Quit Claude Desktop fully, also from the tray, and start it again.
Claude Code	Runs <code>claude mcp add</code> for every folder.	A new Claude Code session has the tools (/mcp lists them).
Codex (CLI, IDE extension, and the Codex part of the ChatGPT desktop app)	Adds Traxcope to <code>~/.codex/config.toml</code> , shared by all three.	Start a new session, or restart the app. ChatGPT's ordinary chat cannot run local tools: use its Codex part.
VS Code (Copilot)	Adds Traxcope to VS Code's MCP servers, for every folder.	Copilot Chat in Agent mode has the tools; start the server from the MCP Servers view if asked.
Cursor	Adds Traxcope to <code>~/.cursor/mcp.json</code> .	Reload the window.
Gemini CLI	Adds Traxcope to <code>~/.gemini/settings.json</code> .	A new gemini session has the tools.
Windsurf (Devin Desktop)	Adds Traxcope to its MCP settings.	Refresh the MCP servers in its panel, or restart it.

Remove takes Traxcope out of an app again; **Update** appears when the app points at another Traxcope installation. For an app not listed, **Copy command** gives the program to enter as a local (stdio) MCP server, and **Copy JSON** the `mcpServers` entry most apps take.

Using it

Traxcope must be running. Ask the app as you would ask TraxAI — *"Which probes are in Traxcope?"*, *"Why was ctrlLoop late?"*. When a tool needs your decision — connecting a probe, recording, writing a file, building — Traxcope's window comes to the front and asks you there, exactly as from the panel.

License	The same as TraxAI's: Standard and Professional.
Privacy	The conversation, and what the tools return to it, go to the app's provider under that app's terms. Only your user account on this computer can reach Traxcope's tools.
Offline	Use the TraxAI panel with your own model; an AI app needs its provider, unless it runs a local model itself.

Reporting a problem

When TraxAI could not do what you asked because of Traxcope itself — a tool that fails, data it cannot read, a wrong result — you can send a report to Embedya:

- **Report a problem** under the TraxAI panel, at any time;
- the link under an answer when something went wrong in that turn;
- TraxAI itself can offer it when it is stuck, from the panel or from your AI app.

The report opens in Traxcope first. It holds TraxAI's summary of the problem (you can edit it), and, with **Include the conversation** ticked, your messages, the answers, and every tool call with what it read — trace data and source excerpts among them. **Preview...** shows exactly that text; untick the box to leave it out. API keys are removed. Nothing is sent until you press **Send Report**; **Export Report...** saves it as a file instead.

Running your own model

Ollama (this machine)

Ollama (<https://ollama.com>) downloads open-weight models — Qwen, Llama, Mistral, Gemma, DeepSeek and others — and runs them on your GPU or CPU. It cannot run closed models such as GPT or Claude.

1. Install Ollama and download a model that supports tools, for example:

```
ollama pull qwen3:14b
```

2. In Traxcope: **Backend: Local (Ollama)**, endpoint `http://localhost:11434`, pick the model.
3. Leave **Context window** at 32,768 unless memory is short.
4. Press **Test model**.

Traxcope asks Ollama for the context window with every request; Ollama's own default is much smaller and it drops what does not fit without an error. If a long conversation still fills the window, Traxcope's log says so — raise the setting or start a new conversation.

Hardware. What makes a local model fast or slow is whether it fits in the GPU's memory. TraxAI's instructions and tool list take about 21,000 tokens of a Qwen model's context before the question — so the context, not only the model, needs memory:

GPU memory	What to expect
4 GB	Too small. Measured: qwen3:4b with a 32k context on an RTX 2050 (4 GB) fit 14 of 37 layers on the GPU, wrote 1.8 tokens a second, and took about 25 minutes per question.
8 GB	A 4B–8B model with a 32k context fits.
12–16 GB	A 14B model with a 32k context.
24 GB or more	32B models, which use TraxAI's tools noticeably better.

On a CPU alone expect minutes per step; Traxcope waits up to 10 minutes for a first reply. Ollama can halve the context's memory with the environment variables `OLLAMA_KV_CACHE_TYPE=q8_0` and `OLLAMA_FLASH_ATTENTION=1`; on the 4 GB card above that moved 4 more layers to the GPU but made little difference to speed.

LM Studio (this machine)

1. In LM Studio, download a model that supports tool use and load it with a **context length of 32,768 or more** (set when loading; LM Studio decides the context, not Traxcope).
2. Start the local server (**Developer** tab). The default address is `http://localhost:1234/v1`.
3. In Traxcope: **Backend: Cloud or your own server, Provider: LM Studio (this machine)**, no API key.
4. Press **Test model**.

vLLM (a company GPU server)

vLLM parses tool calls only when told to. For Qwen models:

```
vllm serve Qwen/Qwen3-14B --max-model-len 32768 \
--enable-auto-tool-choice --tool-call-parser hermes
```

Other model families need their own `--tool-call-parser` (see the vLLM documentation). In Traxcope: **Provider: vLLM / company server**, the server's address (e.g. `http://gpu-server:8000/v1`), and the key if the server was started with `--api-key`.

llama.cpp server

Start it with the model's chat template enabled — without `--jinja` it cannot return tool calls:

```
llama-server -m qwen3-14b-q4_k_m.gguf --jinja -c 32768 --port 8080
```

In Traxcope: **Provider: llama.cpp server** (<http://localhost:8080/v1>).

Company gateways and other servers

A gateway such as LiteLLM, or any other OpenAI-compatible server: **Provider: Custom**, with the address and key your IT department gives you.

Which model?

Traxcope 2.0 was tested on 2026-09-30 with four questions on recorded traces — CPU and task timing, the worst delay of a task, memory, and a recording with gaps. ✓ means TraxAI answered correctly; the time is for the whole answer.

Model	Tasks	Worst delay	Memory	Gaps
OpenAI gpt-6-sol	✓ 21 s	✓ 19 s	✓ 14 s	✓ 41 s
OpenAI gpt-6.1-sol	✓ 39 s	✓ 22 s	✓ 26 s	✓ 60 s
OpenAI gpt-6-astra	✓ 38 s	✓ 33 s	✓ 28 s	✓ 94 s
OpenAI gpt-6-luna	✓ 37 s	✓ 33 s	✓ 32 s	✓ 50 s
Anthropic claude-sonnet-5-5	✓ 43 s	✓ 38 s	✓ 33 s	✓ 74 s
Anthropic claude-sonnet-5	✓ 53 s	✓ 37 s	✓ 53 s	✓ 86 s
Anthropic claude-opus-5	✓ 67 s	✓ 50 s	✓ 77 s	✓ 120 s
Anthropic claude-haiku-4-5	✓ 32 s — no CPU percentages	✓ 32 s	✓ 27 s	✓ 52 s
OpenAI gpt-5.5-pro	✓ 289 s	—	—	—

Each figure is a single run. A local qwen3:4b (Ollama, 4 GB GPU) answered the Tasks question in about 25 minutes, with the CPU load and the Control task's 46 ms block right but without naming the task that held it. Check your own model with **Test model**, then with a question you know the answer to.

Troubleshooting

Symptom	Cause and fix
<i>Connection failed with a key error</i>	The key is wrong or belongs to another provider. Keys are kept per provider: check the provider shown.
<i>Not found — check the model name (Anthropic), or the server's own "model not found" message (OpenAI-compatible providers and servers)</i>	The model name is misspelled or not offered to your key. Open the model list.
<i>Test model: answered without calling the tool</i>	The model cannot use tools, or the server does not parse tool calls: vLLM needs <code>--enable-auto-tool-choice --tool-call-parser ...</code> , llama.cpp needs <code>--jinja</code> .
<i>Answers that ignore the trace or forget the question</i>	The context window is too small. Ollama: raise Context window . LM Studio, vLLM, llama.cpp: set it on the server (32,768 or more).
<i>No reply from the local model for ... s</i>	The model is too slow for this machine. Try a smaller model or a GPU.
<i>The AI provider's rate limit was reached; retrying</i>	The key's per-minute budget is used; TraxAI waits and continues. A higher usage tier removes the wait.
<i>You have no credits remaining / quota</i>	Add credits or raise the quota with the provider.
<i>The AI app shows no Traxcope tools</i>	Traxcope is not running, or Let AI apps on this computer use TraxAI's tools is off. The app's tool call says so. Turn it on: the tools appear without restarting the app.
<i>traxcope-mcp is missing next to Traxcope</i>	The installation is incomplete. Reinstall Traxcope.
<i>The app asks, but nothing happens</i>	The confirmation is in Traxcope's window: look there (its taskbar button flashes if Windows kept it behind).