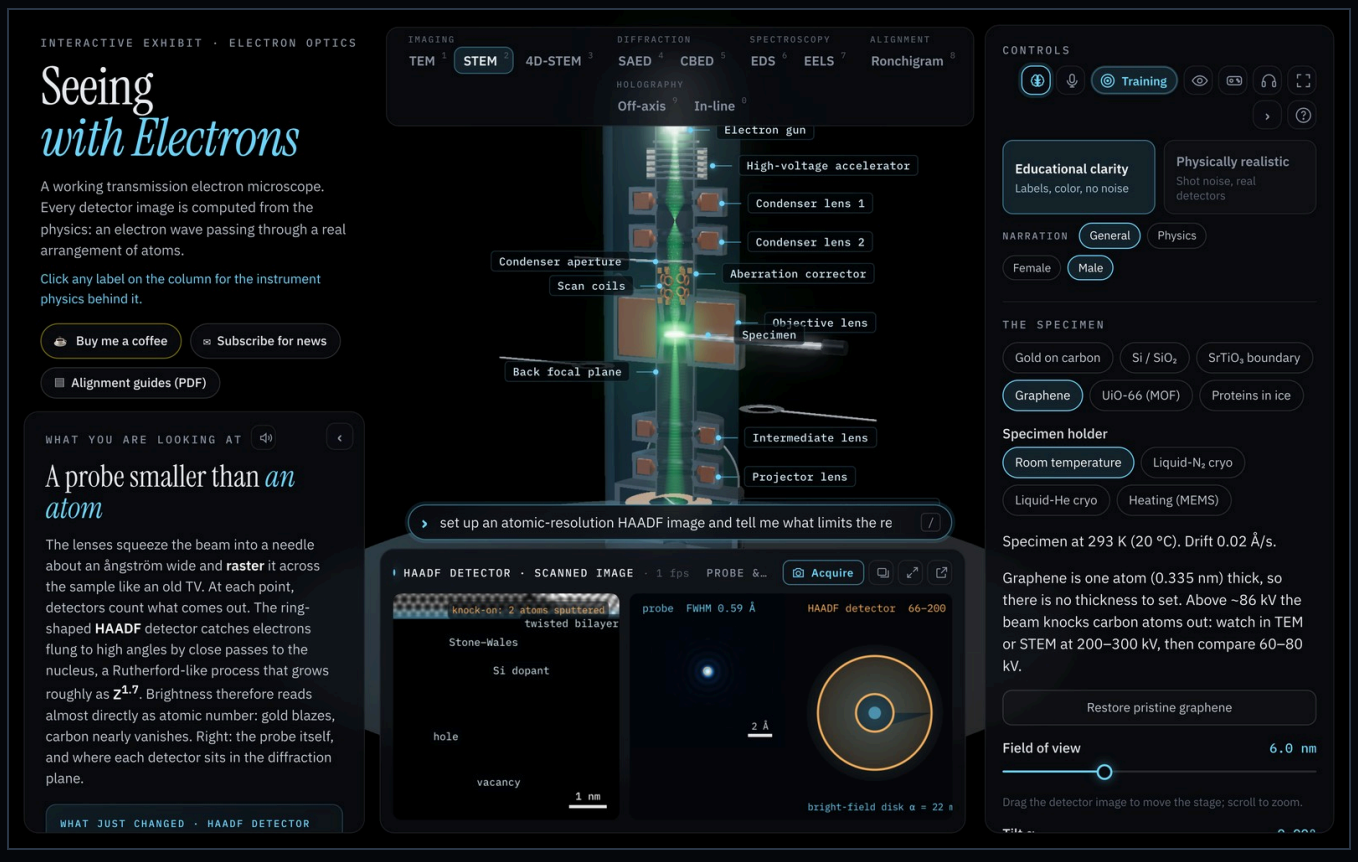
AI agents, *MCP and skills*

Connect Claude, OpenAI Codex, Ollama or another MCP client to TEM-view; use emskills skills with an agent; contribute a skill; and test commands beyond the built-in ones.

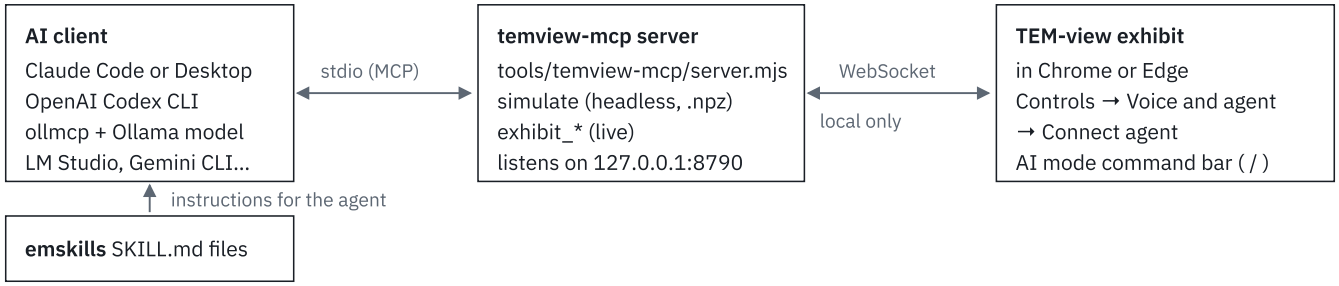


SOP-03, version 1.0, October 2026
For users and developers of TEM-view and emskills
Simulation only: nothing here controls a real microscope

Seeing with Electrons
TEM-view · emskills · Advanced-EM

Scope and overview

The Model Context Protocol (MCP) lets an AI client call tools on your computer. TEM-view ships an MCP server, **temview-mcp**, with tools to simulate data headlessly and to drive an open exhibit. emskills skills describe analysis tasks that an agent can run on that data. The steps: install the server, register it with an AI client, connect the exhibit, test commands, use skills, and contribute a skill.



Requirements

- Node.js 20 or later, and git.
- A copy of the TEM-view repository: `git clone https://github.com/Advanced-EM/TEM-view`
- An MCP client (Step 2).
- Chrome or Edge for the exhibit. Some browsers block a web page from connecting to a local server.
- For contributing skills: Python 3.10 or later and a GitHub account.

Safety and privacy

- The server listens on 127.0.0.1 only and accepts connections only from the exhibit's site and localhost.
- The exhibit connects only when you select **Connect agent**, and it shows when an agent is connected.
- While a training task is running, the exhibit refuses changes from the agent, so recorded practice stays your own.
- With a cloud model (Claude, OpenAI), your prompts and the tool results, including images, go to that provider. With Ollama, everything stays on your computer.
- Speech recognition in Chrome is processed by Google. Typed commands are not sent anywhere unless an agent is connected.

Caution. Do not expose port 8790 to a network or tunnel it to the internet. Anyone who can reach it could drive the exhibit.

Step 1

Install the server

```
git clone https://github.com/Advanced-EM/TEM-view
cd TEM-view/tools/temview-mcp
npm install
node test.mjs          # headless check: all lines should start with "ok"
```

Note the full path of `server.mjs`; the client configurations below need it. In the examples it is written `/PATH/TO/TEM-view/tools/temview-mcp/server.mjs`.

Step 2

Register the server with your AI client

The client starts the server itself when it starts. Register it with one client and restart that client.

Claude Code

```
claude mcp add --scope user temview -- \
  node /PATH/TO/TEM-view/tools/temview-
mcp/server.mjs
```

Claude Desktop

Edit `claude_desktop_config.json` (macOS: `~/Library/Application Support/Claude/`; Windows: `%APPDATA%\Claude\`), keeping any servers already listed:

```
{
  "mcpServers": {
    "temview": {
      "command": "node",
      "args": ["/PATH/TO/TEM-view/tools/temview-
mcp/server.mjs"]
    }
  }
}
```

The same `mcpServers` block works in Cursor (`~/.cursor/mcp.json`), LM Studio (`mcp.json`) and Gemini CLI (`~/.gemini/settings.json`).

OpenAI Codex CLI

Add to `~/.codex/config.toml`:

```
[mcp_servers.temview]
command = "node"
args = ["/PATH/TO/TEM-view/tools/temview-
mcp/server.mjs"]
```

Note. The ChatGPT app connects only to remote MCP servers at an HTTPS address. temview-mcp is a local server, so use Codex CLI or another local client with OpenAI models. Do not tunnel the server to the internet to make it reachable.

Ollama (local models)

Ollama has no MCP client of its own. Use **ollmcp** (MCP Client for Ollama) with a model that supports tool calls, such as `qwen2.5:7b` or `qwen3`. Save the Claude Desktop block above as `servers.json`, then:

```
pip install ollmcp
ollama pull qwen2.5:7b
ollmcp --model qwen2.5:7b --servers-json
servers.json
```

Small local models call tools less reliably. Give short, direct requests, one step at a time.

Result. The client lists the `temview` server with nine tools. In Claude Code, run `claude mcp list`; in other clients, look in the tools or connectors menu.

Step 3

Connect the exhibit and verify

1. Start the AI client from Step 2. This also starts temview-mcp.
2. Open the exhibit in Chrome or Edge: `https://advanced-em.github.io/TEM-view/`, or a local copy (`python3 -m http.server 8765` in the repository folder, then `http://localhost:8765`).
3. In the controls panel, scroll to **Voice and agent** and select **Connect agent**. The button changes to **Agent connected · Disconnect** (Fig. 9).
4. In the AI client, ask: *"What is the exhibit showing?"* The agent calls `exhibit_status` and reports the mode, specimen and optics.
5. Ask for a change: *"Switch to STEM and load graphene."* The exhibit changes, and the agent reports what it did.

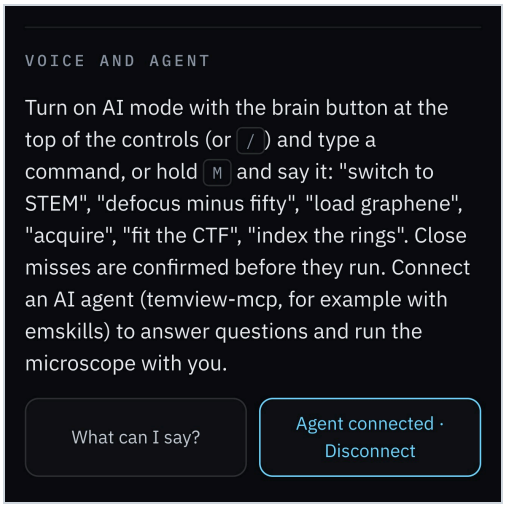


Fig. 9. Controls → Voice and agent, with an agent connected.

Result. The exhibit shows **Agent connected**, and the agent can read and change it.

Live test from the command line

```
cd TEM-view/tools/temview-mcp
node test.mjs --live    # then select Connect agent within 90 s
```

This drives an open exhibit without an AI client: it changes settings, acquires, and checks the answers.

Ports. The exhibit tries `ws://127.0.0.1:8790`, then `8791`. Each AI client starts its own copy of the server, and only one copy can use a port. If two clients are running, the exhibit connects to whichever server holds 8790. Run one client at a time, or start a second one with `TEMVIEW_PORT=8791` in its environment.

Variable	Default	Meaning
`TEMVIEW_PORT`	8790	Local WebSocket port for the exhibit
`TEMVIEW_ORIGINS`	exhibit site	Other web origins allowed to connect, besides localhost
`TEMVIEW_OUT`	./temview-out	Where simulations and saved acquisitions are written

Step 4

Test built-in and beyond-built-in commands

Open the command bar with the brain button at the top of the controls, or press `/`. Hold `M` to say a command instead of typing it. The exhibit handles each sentence in one of four ways.

Sentence	What happens	Agent needed
Exact built-in command	Runs at once (Fig. 10a)	No
Close miss or misspelling	“Did you mean…?” with buttons; answer yes, no, or “the second one” (Fig. 10b)	No
Anything else, with an agent connected	Passed to the agent: “Asked the agent…” (Fig. 10c). The agent receives it through `exhibit_wait_for_voice` and answers with `exhibit_say`, acting with the other tools if asked	Yes
Anything else, no agent	Not understood; the exhibit lists examples (Fig. 10d)	No



Fig. 10a. Built-in commands run directly.

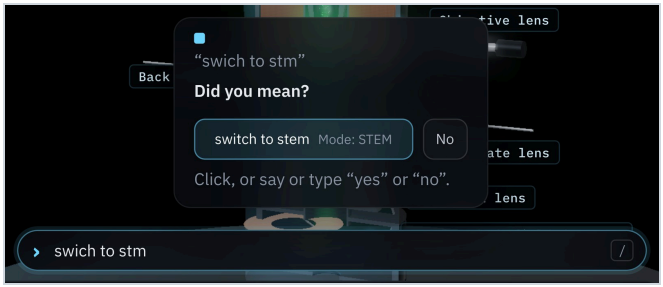


Fig. 10b. A misspelled command is confirmed before it runs.

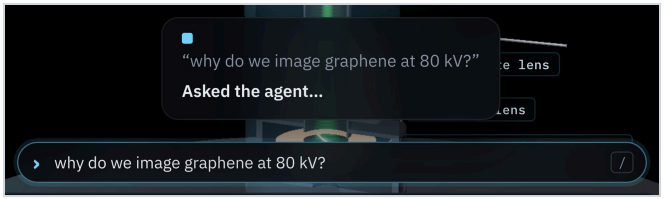


Fig. 10c. A question goes to the connected agent.

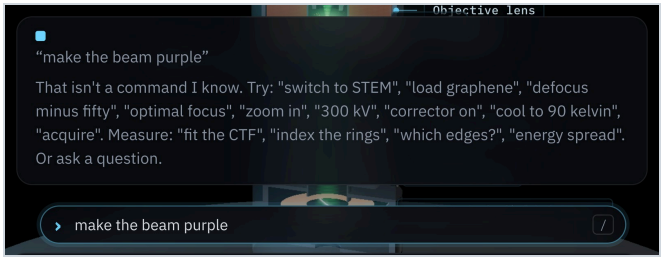


Fig. 10d. Without an agent, unknown sentences are not run.

Starting the agent's listening loop

The agent receives sentences only while it is waiting for them. Tell it once, in the AI client:

```
Listen to me in the exhibit: call exhibit_wait_for_voice, answer each question with exhibit_say in two or three sentences, use the other temview tools when I ask for something, and keep listening until I say stop.
```

Test plan

Run these in order after Step 3, with the listening loop running. Tick each row that behaves as expected.

	Type or say	Expected
☐	switch to STEM	Mode changes at once; caption “Mode: STEM”
☐	load graphene and acquire	Specimen changes and the detectors are recorded; caption lists both
☐	swich to stm	“Did you mean switch to stem?”; “yes” runs it
☐	defocus minus fifty	Defocus set to −50 nm
☐	why do we image graphene at 80 kV?	“Asked the agent…”, then a spoken answer about knock-on damage
☐	set up an atomic-resolution HAADF image of the grain boundary at 300 kV	The agent calls `exhibit_set` (STEM, $SrTiO_3$, 300 kV, HAADF) and says what it changed
☐	take an image and tell me the lattice spacing	The agent calls `exhibit_acquire`, then `exhibit_analyze` with `fit_lattice`, and reports the spacing with the true value
☐	compare 200 and 300 kV for this specimen	Several tool calls in sequence, then a summary; this tests multi-step planning
☐	Start a training task, then ask the agent to change the defocus	The exhibit refuses the change, and the agent reports the refusal
☐	stop	The agent ends the loop

Analyses and emskills

Measurement commands run a built-in analysis and compare it with the simulator's true value. The caption names the matching emskills skill, or says the analysis is not in emskills yet (Fig. 11).

Say	Analysis	emskills
fit the CTF	`fit_ctf`	ctf-fitter
index the rings	`fit_lattice`	lattice-dspacing
which edges are there?	`identify_edges`	eels-edge-id
what is the energy spread?	`fit_zlp`	not yet
fit the EDS peaks	`fit_eds`	not yet
measure the fringes	`fit_hologram`	not yet

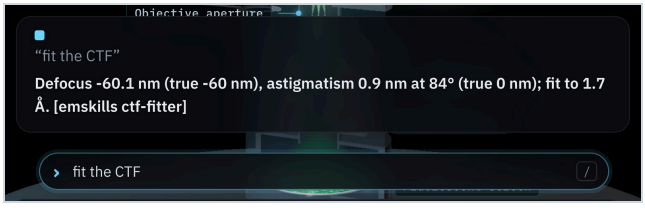


Fig. 11a. CTF fit on gold at −60 nm defocus: fitted −60.1 nm against a true −60 nm.

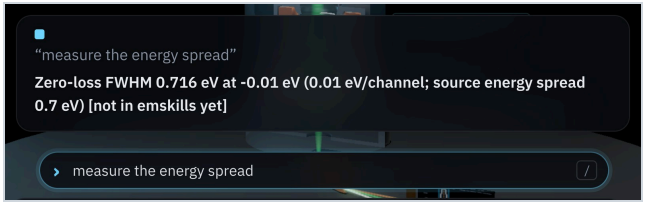


Fig. 11b. An analysis that is not in emskills yet. Candidates for a new skill (Step 6).

Step 5

Use emskills skills with the agent

An emskills skill is a folder with a `SKILL.md` manifest: front matter that names the skill and its tool call, and sections that tell an agent what the skill does, its limits, and how to call it. The registry is at `github.com/emskills-mcp/em-skills` and listed at emskills.org.

Claude Code

Claude Code loads skills from `~/.claude/skills/<name>/SKILL.md`. emskills manifests include the `name` and `description` fields it needs.

```
git clone https://github.com/emskills-mcp/em-
skills
mkdir -p ~/.claude/skills
cp -r em-skills/skills/tem-view-twin
~/.claude/skills/
cp -r em-skills/skills/lattice-dspacing
~/.claude/skills/
```

Start a new session and ask, for example: *"Simulate a gold diffraction pattern at 300 kV with temview, then measure the d-spacings with the lattice-dspacing skill and compare with 2.355 Å."*

Other clients

Give the agent the skill's `SKILL.md` as instructions: add it to a Claude Desktop project, reference it from `AGENTS.md` for Codex, or paste it into the system prompt for an Ollama model.

Typical workflow

1. The agent produces data: `simulate` (headless) or `exhibit_acquire` with `save` (from the exhibit). Both write `.npz` with `metadata.json` holding the calibrations.
2. The agent runs the skill's tool or reference implementation on that file, following the skill's Usage section.
3. The agent compares the result with the truth recorded in the metadata, which is known because the data is simulated.

Note. Simulated data with known truth is the main reason to pair TEM-view with emskills: a skill can be tested on exact ground truth before it is trusted on real data.

Caution. Skills run code on your computer. Read a skill's `SKILL.md` and its reference implementation before installing it, as you would any software.

temview tool reference

Tool	Purpose
`simulate`	Headless acquisition to .npz with metadata; no browser needed
`exhibit_status`	Is an exhibit connected, and what it shows
`exhibit_set`	Change mode, specimen, kV, defocus, field of view, dose, holder
`exhibit_action`	Optimal focus, auto-tune, blank beam, wobbler, narrate
`exhibit_command`	Run a sentence through the exhibit's command grammar
`exhibit_acquire`	Record the detectors; optionally save .npz and .png
`exhibit_analyze`	Measure the live data and compare with the truth
`exhibit_say`	Show a caption and speak it in the exhibit
`exhibit_wait_for_voice`	Wait for the visitor's next question

Step 6

Contribute a skill to emskills

1. Fork `emskills-mcp/em-skills` on GitHub, clone your fork, and create a branch.
2. Create `skills/<slug>/` with the files in the table at right.
3. Write `SKILL.md`: the front matter below, then the sections **Scope**, **Safety**, **Usage** (a JSON block whose `tool` equals `tool_call`) and **Reproducibility** (one command that runs the check).
4. Make a small, deterministic fixture with known ground truth, and a reference implementation that recovers it.
5. Add one line for the skill under **## Skills** in the repository README.
6. Run the validator. It must pass with `--run`:

```
pip install -r requirements-dev.txt
python3 scripts/validate_skills.py --run
```

7. Commit, push, and open a pull request. The same validator runs in CI on the pull request.

Front matter

```
---
name: eels-zlp-width
slug: eels-zlp-width
technique: EELS
category: Spectroscopy
kind: SKILL.md
summary: Energy spread from the zero-loss peak
FWHM
license: MIT
version: 0.1.0
runtime: python 3.11
tool_call: fit_zlp
description: >
  One paragraph for agents: what it does and
when to use it.
---
```

File	Contents
`SKILL.md`	Manifest and instructions
`examples/invoke.json`	A sample tool call, matching the Usage block
`fixtures/sample/`	Small test input with known truth
`fixtures/generate.py`	How the fixture was made
`verify.py` + `requirements.txt`	Reference implementation (Python runtime)
`verify.js` + `package.json`	Reference implementation (Node runtime)

Required front-matter keys: `name`, `slug`, `technique`, `category`, `kind`, `summary`, `license`, `version`, `runtime`, `tool_call`. Techniques: SEM, TEM, STEM, EDS, EELS, cryo-EM, FIB. The slug, name and folder must match, and the version is semver.

Tip. TEM-view can generate fixtures with exact ground truth: `simulate` writes calibrated `.npz` data for any mode. `skills/tem-view-twin` in the registry is a worked example of a Node skill whose check indexes simulated gold rings.

Open candidates. The exhibit's `fit_zlp`, `fit_eds` and `fit_hologram` analyses have no emskills skill yet (Fig. 11b).

Adding a command to TEM-view

New sentences for the built-in grammar go in `parseCommand` in `js/voice.js`. New agent tools go in `tools/temview-mcp/server.mjs` (`registerTool`), with the matching method in `js/agent.js`. Add a check to `tools/temview-mcp/test.mjs` and open a pull request on `Advanced-EM/TEM-view`.

Troubleshooting

Observation	Likely cause	Action
“No agent server at ws://127.0.0.1:8790 or 8791”	The AI client is not running, or the server failed to start	Start the client; check that it lists temview; run `node test.mjs`
Connected, but the agent says no exhibit is connected	Two clients are running; the exhibit joined the other one's server	Close the other client, or use `TEMVIEW_PORT` (Step 3)
Connect agent fails on the public site only	The browser blocks a web page from reaching a local server	Use Chrome or Edge, or a local copy of the exhibit
Questions show “Asked the agent…” but nothing is said	The agent is not in its listening loop	Give the listening instruction in Step 4 again
The agent stops listening after a few questions	Common with small local models	Restart the loop; use a larger model; keep requests short
The agent calls the wrong tool or invents arguments	Small model, or a long multi-part request	Ask for one change at a time; name the setting and value
Changes are refused	A training task is running	Finish or give up the task first
Validator fails on Usage	The Usage JSON or `invoke.json` does not match `tool_call`	Make the three agree
Validator fails with `--run`	The reference implementation does not recover the fixture's truth	Run the Reproducibility command locally and fix the check

Record

	Item	Value or criterion	Initials
☐	Server installed, `node test.mjs` passes	All lines “ok”	
☐	AI client and model	_____	
☐	Exhibit connected	“Agent connected” shown; port _____	
☐	Built-in commands	Test plan rows 1 to 4 pass	
☐	Beyond built-in commands	Test plan rows 5 to 10 pass	
☐	Skill used with the agent	Skill: _____	
☐	Skill contributed (optional)	Pull request: _____	

Resources

TEM-view is a physics-based TEM and STEM simulator that runs in a web browser. emskills is an open registry of AI agent skills for electron microscopy. Both are open source.

The temview-mcp server and its README

github.com/Advanced-EM/TEM-view/tree/main/tools/temview-mcp

Scan the code with a phone.

TEM-view

advanced-em.github.io/TEM-view
github.com/Advanced-EM/TEM-view

emskills

emskills.org
github.com/emskills-mcp/em-skills

Model Context Protocol

modelcontextprotocol.io

SOP-03 AI agents, MCP and skills, version 1.0, October 2026
Client names and configuration paths change between versions: check your client's documentation if a step does not match.
Simulation only. Nothing here controls a real microscope.

Seeing with Electrons

TEM-view · emskills · Advanced-EM