Using Nokta with an AI assistant

Nokta exposes its notebooks via the **Model Context Protocol (MCP)** — an open standard that lets AI assistants read and write external data. Once your MCP-capable client (Claude Desktop, Claude Code, Cursor, Cline, and others) knows about Nokta, you just talk to the assistant — no extra steps per conversation.

Getting started

1. Open the **Configuration** tab. For Claude Desktop, click *Configure*. For other clients, copy the snippet shown there into the client's MCP config.
2. Restart your client (or reload its config) so it picks up the new server.
3. Start a chat and ask the assistant to read, search, or edit your notes.

Two ways to use it

- **Live mode (app running).** If Nokta is open, the assistant connects to the notebook you currently have open and can **read and write**. This is the default — no file path needed. Changes appear in the app immediately.
- **Read-only mode (any .nokta file).** When Nokta is closed, or you want the assistant to look at a different notebook without opening it, give it the path: *"Open D:\Notes\Worldbuilding.nokta and summarise it."* The assistant can read and search the file, but cannot make changes.
- **Peeking.** The assistant can also briefly inspect another `.nokta` file by path without switching its current connection — handy for cross-notebook lookups.

What the assistant can do

- **Browse** — list the note tree, get notebook info and per-note details.
- **Read** — fetch a note by id or path, or one specific block within it.
- **Search** — full-text search across all notes.
- **Filter** — list notes carrying a specific flag emoji.
- **Write** — create, update, rename, move, or delete notes *(live mode only)*.
- **Annotate** — set or clear a note's flag or synopsis *(live mode only)*.

Your AI client asks permission before actions on your notes, so you stay in control of what actually happens.

Precise references

Right-click a note in the left panel (or a block, via its drag handle in the editor) and choose **Copy path**. You get a `nokta://` link like `nokta://notes/12/my-note` or `nokta://notes/12/my-note#b3` for one specific paragraph. Paste these into a prompt to point the assistant at exactly one note or one block — no ambiguity, no wasted reading.

Built-in safety

- **Your edits win.** Every write requires the assistant to know the note's current version. If you changed the note in Nokta in the meantime, the assistant's write is rejected and it must re-read first — it can never silently overwrite your work.
- **Visible activity.** In live mode, the note tree briefly highlights notes the assistant is reading or has just modified.
- **One notebook per connection.** The assistant works on one notebook at a time — the live one, or the file you pointed it at. Ask it to switch when needed.
- **Folder mode is off-limits.** MCP connects to notebooks only; while a folder is open, live connections are refused.

Examples

MCP is a toolbox that lets an AI read from and write to Nokta. The most rewarding way to use it is **collaborating on documents**: you discuss, the assistant keeps the note updated. When you come back tomorrow, the state is in Nokta — not stuck in a chat history.

The pattern

1. Ask the assistant to *create or read* a working note for the topic at hand.
2. Tell it explicitly: *"we'll work in Nokta — keep the note updated as we go."*
3. Discuss, decide, change your mind. The assistant edits the note in place at each step.
4. Close the chat whenever you like. The note holds the state.

Working documents

> *List all modules in my project under "Modules" as a checklist in a new note called "Module sanity check". Then we go through them one by one. Keep track of the updates.*

> *Read the design document at D:\Notes\Design.nokta and set up the current notebook for implementation: open questions, decisions still to make, and a checklist of concrete steps.*

The same recipe works for governance docs, worldbuilding bibles, and design documents.

> *Summarise our conversation so far into a new note and turn the action items into a checklist.*

> *I'll keep talking. Append a one-line entry to nokta://notes/45/decision-log every time we make a decision, with date and short rationale.*

Quick one-shots

For when you don't need a working document — just an answer or a small change:

> *Search my notes for "budget" and list the matches with a one-line summary each.*

> *Read nokta://notes/12/my-note and rewrite the paragraph at #b3 to be more concise.*

> *Set the ❝❝ flag on every note that mentions "pricing".*

> *Give every note in the "Chapters" branch a one-line synopsis.*

Across notebooks

> *Open D:\Notes\Archive.nokta read-only and tell me which characters from there also appear in my current notebook.*

The assistant opens the other file by path; your live notebook stays the active write target.

Tips

- **Point, don't describe.** Pasting a `nokta://` link (right-click → *Copy path*) beats describing a note by title.
- **Use flags as a queue.** Flag notes 🚧, then ask the assistant to "process everything flagged 🚧" — and let it clear the flag on each note it finishes.
- **Frontmatter counts.** The assistant sees each note's frontmatter and flag, so it can filter and update metadata too.

Tool reference

The assistant uses these tools under the hood. You never call them yourself — but knowing what exists helps you phrase requests, and helps you judge the permission prompts your AI client shows.

Connecting

Tool	What it does
`nokta_connect`	Connect to a notebook. Without a path: the notebook currently open in Nokta (live, read & write). With a path: that `.nokta` file, read-only.
`nokta_read_file`	Peek into a *different* `.nokta` file by path without switching the current connection.
`nokta_info`	Info about the connected notebook: title, path, mode, note counts, metadata.

Reading & searching

Tool	What it does
`nokta_tree`	The note tree: titles, ids, content types, synopses, flags.
`nokta_read`	Read one note — full content, or a single block (`#b3`). Returns the note's version, needed for safe writes.
`nokta_search`	Full-text search across all notes, with context snippets.
`nokta_flag`	List all notes carrying a specific flag emoji.

Writing (live mode only)

Tool	What it does
`nokta_write`	Update a note's content — the whole note or a single block. Requires the note's current version; if you edited the note in the app meanwhile, the write is rejected and the assistant must re-read.

Tool	What it does
`nokta_create`	Create a new note (optionally under a parent, at a position). New notes get your default frontmatter template.
`nokta_delete`	Move a note and its children to the recycle bin.
`nokta_rename`	Rename a note.
`nokta_move`	Move a note to a different parent and/or position.
`nokta_synopsis`	Read or update a note's synopsis.

Good to know

- Every response names the notebook it came from, so multi-notebook conversations stay unambiguous.
- Write operations are transactional — a change either fully lands (content, search index, activity signal) or not at all.
- Deletes go to the recycle bin, so even an approved-too-quickly delete is recoverable in the app.

Troubleshooting

The assistant doesn't see Nokta

Make sure you restarted your AI client after configuring — most clients read their MCP config only at startup. For Claude Desktop, check the **Status** on the Configuration tab; it should say "Configured".

Status says "Configured, but pointing at a different install"

You moved or reinstalled Nokta since configuring. Click **Re-apply configuration** on the Configuration tab.

Config file is invalid JSON

Click **Show config file** to open it, fix the JSON manually, then try *Configure* again.

The assistant connects read-only while the app is running

- Check that the notebook is actually open in the app (not the start screen, not folder mode — MCP is disabled while a folder is open).
- The assistant may have been given a file path — with a path it always opens read-only. Ask it to connect *without* a path to reach the live notebook.

A write failed with a version error

That's the safety net doing its job: the note changed (you edited it in the app) between the assistant's read and its write. The assistant should simply re-read the note and apply its change again — no action needed on your side.

Other clients don't pick up the snippet

Check that you pasted it inside an `mcpServers` block (not at the JSON root) and that the surrounding JSON is still valid. Then restart the client.