Noma CLI Reference

The `noma` CLI parses, renders, and validates `.noma` source files. This page is itself written in Noma — the same source produces the rendered HTML you're reading and the deterministic LLM context an agent would consume.

Install

```
npm install -g @ferax564/noma-cli
# or one-off
npx @ferax564/noma-cli render path/to/file.noma --to html
```

> The CLI auto-detects the `themes/default.css` shipped with the package. To use a custom theme, pass `--theme path/to/theme.css` (coming in v0.2).

Commands

`noma parse <file>`

Print the parsed AST as JSON. Useful for debugging the parser or building tools that consume Noma documents.

```
noma parse examples/thesis.noma
noma parse examples/thesis.noma --out ast.json
```

Returns the full typed AST defined in `src/ast.ts`. Block IDs are stable across re-parses of unchanged content, so AST diffs map cleanly to source diffs.

`noma render <file> [--to <target>]`

Render a `.noma` file to one of the supported targets.

Target	Output
`html`	Standalone HTML document with the default theme
`llm`	Deterministic plain-text context for LLM consumption
`json`	The parsed AST (alias of `noma export`)

```
noma render docs/spec.noma --to html --out dist/spec.html
noma render docs/spec.noma --to llm
noma render docs/spec.noma --to json --out dist/spec.json
```

Use `--no-standalone` to emit just the HTML body (for embedding inside an existing page). Use `--title "..."` to override the document title.

`noma check <file>`

Validate a Noma document. Exits 1 if any errors are present, 0 otherwise.

```
noma check examples/thesis.noma
```

Catches: duplicate block IDs, broken `for=` references on evidence blocks, broken internal links, invalid frontmatter, plot blocks missing a dataset or `data=` attribute, and (in v0.2) claims missing supporting evidence.

`noma export <file>`

Alias for `noma render <file> --to json`. Kept as a separate command because it's the most common scripted use case (CI pipelines, agent context, RAG indexing).

Architecture

◆ **Parser**

Hand-written recursive descent. Tracks fence depth by counting leading colons. Never throws on malformed input — produces a best-effort AST and lets the validator complain.

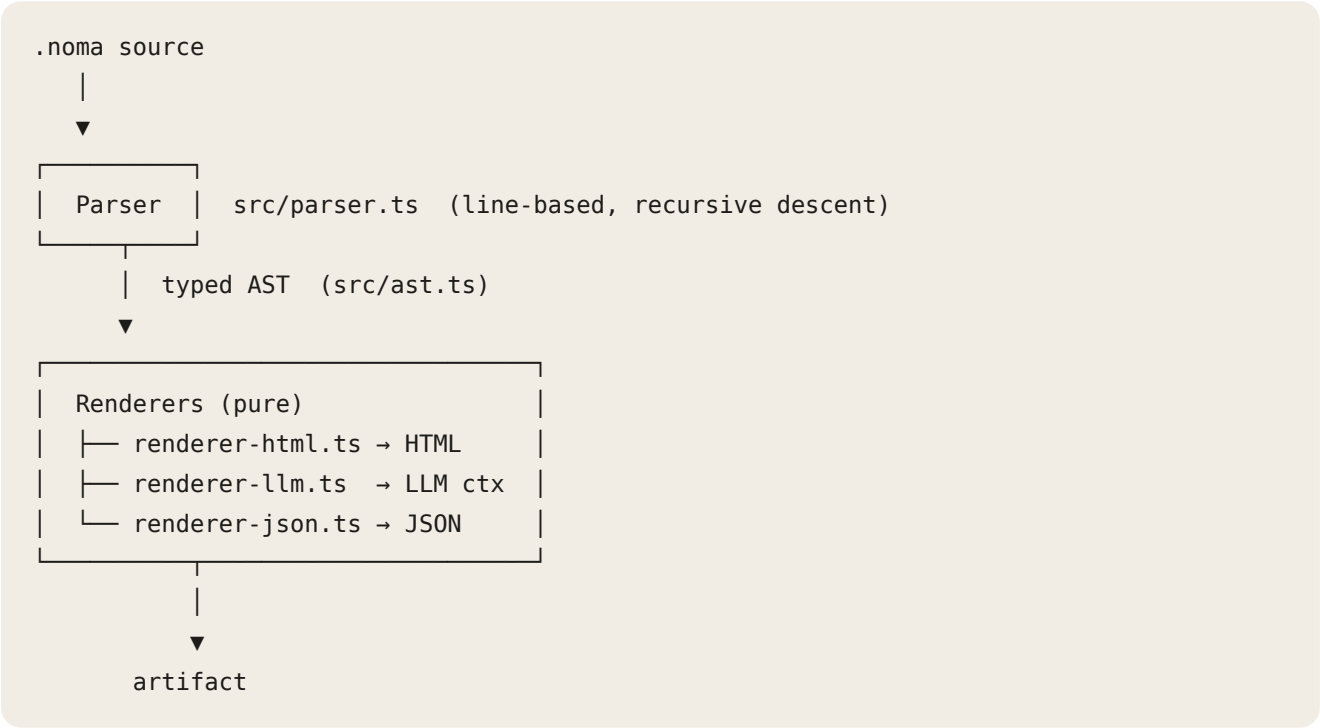
◆ AST

Discriminated union in `src/ast.ts`. Single source of truth. Adding a node type is a one-line change that the TypeScript compiler propagates to every renderer's switch.

◆ Renderers

Pure functions. `AST → string`. No I/O, no globals. Three in core today (HTML, LLM, JSON); PDF is a wrapper around the HTML renderer + Puppeteer.

Data flow



Common errors

Stray triple-colon at top level. `:::card` only opens a child block one level deeper. At the top of a document it has no parent and the parser will emit a `parser/orphan-fence` diagnostic. Fix: wrap in `::grid` or use `::card`.

Duplicate block ID. Block IDs are user-facing API. The validator emits `validator/duplicate-id` and `noma check` exits non-zero. Fix: rename one of the blocks. If they were intentionally duplicated, they shouldn't have been — split or merge.

Plot without data. A `::plot` block without a `data=` or `dataset=` attribute renders as a placeholder SVG. The validator emits `validator/plot-missing-data` as a warning, not an error, so the build still passes — but the artifact won't have real data in it.

Programmatic use

```
import { parse, renderHtml, validate } from "@ferax564/noma-cli";

const source = await fs.readFile("doc.noma", "utf8");
const ast = parse(source, { filename: "doc.noma" });
const diagnostics = validate(ast);
if (diagnostics.some(d => d.severity === "error")) {
  throw new Error("invalid Noma document");
}
const html = renderHtml(ast, { standalone: true });
```

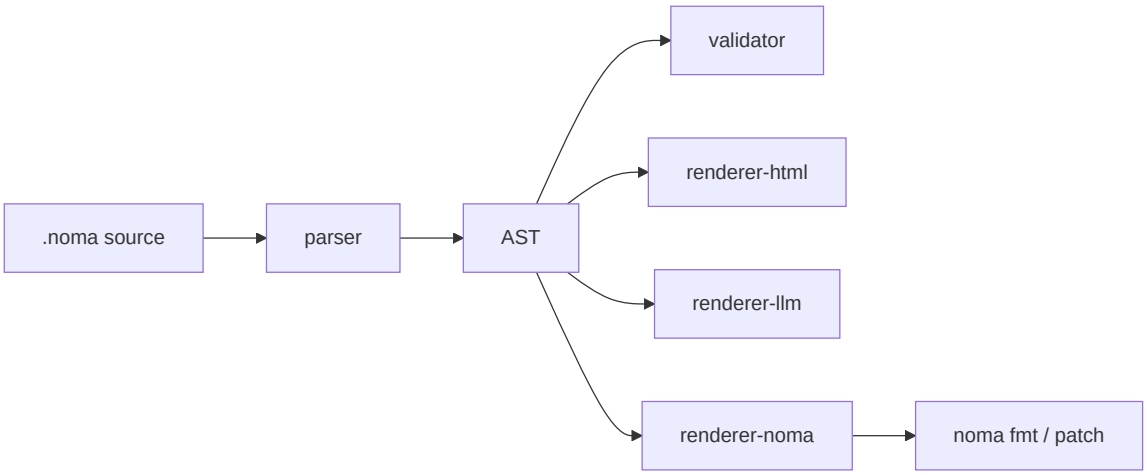
Patch protocol (preview, v0.3)

Agents should not rewrite full files. They propose block-level operations that the CLI applies safely.

```
{
  "op": "replace_block",
  "id": "asml-euv-moat",
  "content": "ASML's moat rests on EUV optics, mechanics, and supply chain — not just excl
}
```

Operations: `add_block`, `replace_block`, `delete_block`, `move_block`, `update_attribute`, `add_evidence`, `add_comment`, `resolve_comment`, `rename_id`. See `docs/agent-protocol.noma` for the full schema.

Architecture (v0.5 — interactive diagrams)



Cross-links

- See `docs/spec.noma` for the full block reference.
- See `docs/agent-protocol.noma` for the patch protocol.
- See `docs/direction.noma` for the product positioning.
- See `examples/research-thesis.noma` for a reasoning-heavy demo.