ON-PREMISES / MCP SERVER EDITION

KlastroHeron BioMedical Configuration & Usage Manual

A practical guide for administrators deploying this container: how to configure it, what it does, what it must never be used for, and what to expect from its answers.

1. What This Application Is

KlastroHeron BioMedical is a literature-search and synthesis tool. Given a natural-language question, it searches **Europe PMC** for relevant biomedical research papers and, when a named drug is involved and the EMA feature is enabled on this deployment, cross-checks **the European Medicines Agency's (EMA) public regulatory database** (marketing authorisation status, ATC code, orphan/biosimilar/generic flags, and any safety alerts / DHPCs on record). It then returns one synthesized, cited answer — not a raw list of search results.

It exposes two tools over the Model Context Protocol (MCP):

Tool	Purpose
`search_literature`	Natural-language question → synthesized, cited answer drawn from Europe PMC (and EMA, for named-drug questions, if enabled).
`lookup_drug_regulatory_info`	Direct EMA lookup for one named drug — regulatory status and safety alerts only, no literature search. Only available when EMA is enabled on this deployment.

2. Not a Diagnostic Tool — Read Before Deploying

> ⚠ **IMPORTANT**
>
> This application is **not a medical device** and is **not intended for diagnosis, treatment decisions, or clinical decision-making for any individual patient**. It exists to help find and summarize *published research and public regulatory records* — nothing it returns should be treated as medical advice for a specific person.
>
> This holds **even if a response happens to sound like it is addressing your specific situation.** Language models can occasionally produce answers that read as personalized guidance. If that happens, it is a byproduct of language generation, not a clinical judgment — **do not act on it as one.** Always consult a qualified physician for any decision about diagnosis, treatment, or medication.

The application has guardrails built in specifically for this: questions that read as asking for a personal diagnosis, a treatment recommendation, or anything safety-critical (e.g. self-harm risk) are detected before an answer is generated and are routed to a fixed, non-diagnostic response instead — for self-harm-risk questions, that response is crisis-support resources, not medication information. Fully out-of-scope questions (non-medical topics) are rejected outright rather than answered. These guardrails reduce risk; they do not make the underlying language model infallible, which is why the rule above is unconditional regardless of what a given response contains.

3. What You Can Ask It

The intended use is research-oriented, not patient-specific. Good questions look like:

- Literature and treatment-trend questions — e.g. "*What are the recent treatment trends for type 2 diabetes?*" or "*Latest research on GLP-1 receptor agonists for obesity.*"
- Named-drug questions — e.g. "*What are the side effects of ibuprofen?*" or "*Ozempic EMA authorisation status.*" These also trigger the EMA regulatory lookup if enabled.
- Direct regulatory lookups via `lookup_drug_regulatory_info` — authorisation status, ATC code, safety alerts for one named medicine.

Each literature query retrieves the most relevant matching papers currently indexed in Europe PMC — not a fixed archive, so results reflect what is newly published as well as established research. This deployment returns up to **20 papers per query**, together with citation-metric enrichment (via OpenAlex) and an extracted methodology summary alongside the answer.

4. Configuring Supported Languages — `kh-config.yml`

The container auto-detects the language of each incoming question and answers in that same language — but only for languages your organization has enabled. This keeps memory usage down (only the enabled languages' detection models are loaded) and lets you scope the deployment to the languages your users actually use.

TO CHANGE THE ENABLED LANGUAGES

Edit the `languages:` list in `kh-config.yml`, then **restart the container** — this file is only read at startup, so changes do not take effect on a running container.

```
docker run -v ./kh-config.yml:/app/kh-config.yml ... <image>
```

The default configuration shipped with this deployment enables the languages most commonly requested by customers:

`en` `it` `fr` `de` `nl` `pt-PT` `es`

`en` is always included automatically as the fallback language, even if you remove it from your list.

All supported language codes

`en` `ko` `fr` `de` `es` `it` `pl` `cs` `ro` `hu` `hr` `bg` `pt` `pt-PT`
`sl` `nl` `el` `no` `sv` `fi` `da` `is` `et` `lt` `lv` `ja` `ru` `uk`
`tr` `ar` `he`

Example — a deployment enabling only English, Korean, and Japanese:

```
languages:
  - en
  - ko
  - ja
```

5. How Answers Are Grounded

- **Europe PMC** — the primary literature source for every query.
- **EMA regulatory database** (if enabled) — downloaded directly from EMA's own public data exports and refreshed automatically (on startup, then roughly monthly), so records reflect EMA's current published data rather than a static snapshot.
- **Citation metrics** — via OpenAlex, shown as citation counts and percentile badges alongside each reference.

Every answer includes source citations (DOIs) so results can be independently verified against the original papers or EMA's own records.

6. Building a Tabbed UI on `search_literature`

If you are building a UI (or any client that wants the answer broken into distinct parts rather than one flat block of text) on top of the `search_literature` tool, use the `sections` field in its response instead of trying to parse the `answer` string yourself:

```
{
  "answer": "... full answer, all sections concatenated ...",
  "sections": [
    {"key": "Regulatory",    "text": "..."},
    {"key": "Evidence",      "text": "..."},
    {"key": "Implications",  "text": "..."},
    {"key": "Limitations",   "text": "..."},
    {"key": "Summary",       "text": "..."},
    {"key": "Methodology",   "text": "..."},
    {"key": "References",   "text": "..."}
  ],
  "methodology_summary": "...",
  ...
}
```

THE SET OF SECTIONS VARIES PER QUERY

Do not assume a fixed list of tabs. `Regulatory` only appears when the question resolved to a named drug EMA has a record for; `Methodology` only appears when enough of the retrieved papers share a common method to summarize. A key that did not apply to a given query is simply absent from the array — build your UI to render only the tabs it receives, not a fixed set.

Keys you may see

Key	Suggested tab label	Appears when
`Overview`	Overview	Default opening context for non-drug questions, before the first detected section.
`Regulatory`	Regulatory Status	The question resolved to a specific drug and EMA lookup is enabled.
`Evidence`	Research Evidence	Relevant paper abstracts were found.
`Implications`	Clinical Implications	Paired with Evidence.
`Limitations`	Study Limitations	Paired with Evidence.
`Summary`	Summary	Paired with Evidence.
`Methodology`	Methodology	When enough retrieved papers share a common method.
`References`	References	Whenever at least one article was cited.
`HeronTalk`	Notes	Personal-health-style questions.

The `key` value is always this fixed English form, regardless of the answer's language — a French or Korean question still returns `"key": "Evidence"`, with only the `text` content translated. This is deliberate: it lets a UI map keys to tab labels in whatever language it displays, without parsing translated headings out of the answer text.

```
TAB_LABELS = {
    "Overview": "Overview", "Regulatory": "Regulatory Status",
    "Evidence": "Research Evidence", "Implications": "Clinical Implications",
    "Limitations": "Study Limitations", "Summary": "Summary",
    "Methodology": "Methodology", "References": "References",
    "HeronTalk": "Notes",
}

tabs = [
    {"label": TAB_LABELS.get(s["key"], s["key"]), "content": s["text"]}
    for s in result["sections"]
]
```

If you don't need a tabbed breakdown — a plain chat agent, for example — just use the flat `answer` string and ignore `sections` entirely. For rejection or error responses (out-of-scope questions, sensitive-content redirects, retry exhaustion), `sections` is an empty array; render `answer` as a single message in that case.

7. Full Response Schema — `search_literature`

The fields below appear in this exact order in every `search_literature` response. Field order is not meaningful to a JSON parser, but is listed here so the raw response is easy to read while debugging.

Key	Type	Meaning
`in_scope`	boolean	False only for out-of-scope (non-medical) or sensitive-content rejections. True otherwise, including error fallbacks.
`answer`	string	The full answer, all sections concatenated into one block of text (Markdown-formatted).
`sections`	array	The same answer, pre-split by section — see Section 6 above.
`methodology_summary`	string | null	Convenience duplicate of `sections[].text` where `key == "Methodology"`. Null when no Methodology section was produced.
`articles`	array	Structured data for every paper cited in the answer — see below. This is the canonical, parse-free source for reference data.
`drug_info`	object | null	Structured EMA record — see below. Null unless this was a named-drug question with EMA enabled.
`prisma`	object	Search-transparency metadata — see below. Null for rejection/error responses.

`articles[]` object fields

Field	Notes
`title`	Paper title.
`authors`	Author list as one string, comma-separated.
`doi`	Full DOI URL. May be an empty string if Europe PMC has no DOI for this paper.
`pmid`	PubMed ID.
`pmcid`	PubMed Central ID. Empty string if not in PMC.
`pmc_url`	Direct link to the free full text on PMC, when available.
`abstract`	Full abstract text, may include inline HTML tags (e.g. `<h4>`) as returned by Europe PMC.
`year`	Publication year, as a string.
`journal`	Journal name. Frequently an empty string — not reliably populated by Europe PMC for every record.
`source`	Source database and year, e.g. `"Europe PMC (2026)"`.
`origin`	Always `"europepmc"` currently.
`isOpenAccess`	Boolean.
`citation_count`	OpenAlex citation count. Present only when OpenAlex had data for this DOI.
`percentile`	OpenAlex citation percentile (0–100). Present only alongside `citation_count`.
`medals`	Array of badge strings (e.g. `"🥇 Top 5%"`). Present only alongside `citation_count`.
`citation_badge`	Pre-formatted display string combining count and medals (e.g. `"📚 4 | 🥉 Top 20%"`), or an empty string when no OpenAlex data was found for this paper.

`drug_info` object fields

Field	Notes
`name`	EMA's registered product name.
`substance`	Active substance(s), semicolon-separated for combination products.
`therapeutic_area`	MeSH therapeutic area.
`status`	Marketing authorisation status, e.g. `"Authorised"`.
`atc`	ATC code.
`biosimilar`	"Yes" / "No", per EMA's export.
`orphan`	"Yes" / "No", per EMA's export.
`generic`	"Yes" / "No" — EMA calls this field "Generic or hybrid".
`auth_date`	Marketing authorisation date, DD/MM/YYYY.
`monitoring`	Whether under EMA additional monitoring.
`url`	Link to the medicine's EMA page.
`safety_alerts`	Array of `{type, date, title}` — one entry per DHPC/safety notice on record. Empty array if none.

`prisma` object fields

Field	Notes
`database`	Always `"Europe PMC"` currently.
`query_original`	The question as submitted (after normalization).
`query_optimized`	The boolean search string actually sent to Europe PMC.
`total_identified`	Total matching records Europe PMC reports for the optimized query — often tens of thousands; this is a raw search-hit count, not a quality signal.
`after_screening`	Number of papers actually retrieved and passed to the answer (equals `articles.length`).
`excluded`	`total_identified − after_screening` — included for PRISMA-style reporting, not a per-paper relevance judgment.

Parsing the References section — use `articles`, not text parsing

The `sections[].text` entry with `key == "References"` is a human-readable rendering of the same data already available, structured, in the `articles` array. If you need reference data

programmatically — to build reference cards, deduplicate, sort by citation count, link out by DOI — read it from `articles` directly. There is no need to parse the References text at all.

IF YOU ONLY HAVE THE FLAT TEXT

Some integrations only retain the flat `answer` string (for example, a chat log that stores plain text). In that case, each reference entry inside the References block follows a fixed pattern: a line starting with `"• "` holding the title, followed by an indented `" Authors: ..."` line, then an optional `" DOI: ..."` line, then an optional citation-badge line (`" 📚 N | medal"`) — with a blank line between entries. The `Authors:` line is the one field always present on every entry, so it is the reliable anchor for splitting entries apart:

```
import re

entries = re.split(r"\n(?=• )", references_text)
for entry in entries:
    m = re.match(
        r"• (?P<title>.+?)\n\s+Authors: (?P<authors>.+?)"
        r"(?:\n\s+DOI: (?P<doi>\S+))?",
        entry.strip(), re.DOTALL,
    )
    if m:
        print(m.group("title"), m.group("authors"), m.group("doi"))
```

Treat this as a fallback only — the pattern is a rendering detail of the current prompt and is not a guaranteed stable wire format the way the `articles` array is.