UHB — User Manual

Version 0.1.0 · Universal Harness Backend
This manual covers installation, the commands and the guardrails of the reference implementation. It ships inside the package and is served by `uhb run` at `/docs/manual.pdf`.

---

1. What UHB is, in three sentences

UHB is the **execution backend** that sits under a control plane (Harness Maxxing, OpenClaw, Hermes Agent, or your own CLI). It does not think for you: it decides **what a turn is allowed to do** (modes), **which model answers** (router) and **where knowledge is written** (compiled wiki) — and it refuses by default anything that is not explicitly authorised.

---

2. Installation

Requires Python 3.11 or newer. One runtime dependency: `httpx`.

2.1 Package

```
pip install uhb-harness          # CLI, panel and the embedded manual
pip install "uhb-harness[yaml]"  # additionally enables `uhb config set`
```

The PyPI name is `uhb-harness` because `uhb` belongs to an unrelated project. The command is `uhb`.

2.2 From a checkout

```
git clone https://github.com/Antoine-Thomas/UHB-Universal-Harness-Backend.git
cd UHB-Universal-Harness-Backend
./install.sh               # creates .venv, installs, verifies uhb --version
./install.sh --with-yaml   # adds pyyaml, needed by `uhb config set`
```

Option	Effect
`--dry-run`	print the commands, change nothing
`--with-yaml`	install pyyaml (writes `config.yaml`)
`--help`	recall these options

The installer never uses `sudo`, never writes outside the checkout, and is a no-op reinstall on a second run.

2.3 Verify

```
uhb status
```

It prints the version, the git commit, the providers with `key_env` and a `loaded` / `MISSING` state, the routing policy and the available modes.

---

3. Commands

3.1 Read-only

Command	Purpose
`uhb status`	version, repository, providers, policy, modes
`uhb status --json`	the same snapshot as JSON (for a script or a UI)
`uhb modes list`	name, kind, tier, write scope, turn budget
`uhb modes show <name>`	the full profile of one mode
`uhb config list`	every effective key, defaults included
`uhb config get <key>`	one value, dotted path (`routing.default`)
`uhb providers`	provider table (protocol, URL, variable **name**)
`uhb health`	backend health probe
`uhb docs`	path of the manual PDF and its download URL
`uhb ui`	what to paste into a dashboard, and where the guide is
`uhb brain status`	RAG, SiYuan, MARI: up or down, with detail

3.2 Acting

Command	Purpose
`uhb run [--mode chat] [--port 8101]`	start the HTTP entrypoint
`uhb execute "<prompt>" --mode dev`	one full turn through the configured harness
`uhb config set <key> <value>`	write one configuration key
`uhb update [--dry-run]`	fast-forward from the remote

`uhb run` serves five surfaces:

- `POST /v1/chat/completions` — OpenAI shape (streaming supported)
- `POST /v1/messages` — Anthropic shape
- `GET /` — the UHB control panel (status badge, quick commands, manual download)
- `GET /status` — state as JSON
- `GET /docs/manuel.pdf` / `GET /docs/manual.pdf` — this manual, French and English

---

4. Modes

A mode is a JSON file: read scope, write scope, allowed tools, system prompt, model tier, turn budget.

Mode	Reads	Writes	Tier
`chat`	`wiki/**`, `raw/**`	nothing	eco
`dev`	everything	`uhb/**`, `tests/**`, `docs/**`, memory files	eco
`review`	everything	nothing	eco
`observe`	`wiki/**`, `logs/**`, `data/**`, `*.log`	`wiki/log.md`, `reports/**`	free

Non-negotiable rules:

1. `deny` always wins over `allow`.
2. An unknown key in a profile is a load-time error, not a silent no-op.
3. An absolute path or a `..` in a pattern is refused at load time.
4. Every write goes through `check_write` **before** the file is opened.
5. A write path with no mode raises `ModeRequired`: there is no "implicit envelope".

Adding a mode means dropping a file into `uhb/hub/modes/profiles/`, or into the directory named by `UHB_MODES_DIR` (which replaces by name and adds new ones).

---

5. The router

```
from uhb.hub.modes import get_mode
from uhb.hub.router import route

decision = route("Refactor the router and add tests", get_mode("dev"))
print(decision.reason)
```

Three steps: **classify** (length, code blocks, multi-step verbs, file paths, several questions), **plan** the chain (the mode's tier, then `eco` → `free` → `paid`, filtered by the complexity each model declares), **execute** (first candidate that answers).

Fallback, in order: a remote provider whose key variable is absent is **skipped**, not called; a provider that raises is **recorded then skipped** (the exception class is kept, never the payload); if everything fails, `AllProvidersFailed` is raised with the attempts — never an empty string that reads like an answer.

`paid` is only reachable when the mode carries `allow_paid: true`. No shipped profile allows it: an expensive model remains a manual decision.

---

6. The compiled wiki

Three layers: `raw/notes/` (immutable sources), `wiki/` (compiled pages), `schema.md` (the conventions). Three operations:

Operation	What it does
`ingest`	one source → N pages, **a single call** to the model, atomic write
`query`	reads the index and returns candidate pages; never answers for you
`lint`	orphans, dead links, index gaps, invalid frontmatter, contradictions

Ingestion is idempotent: the sha256 of the source is stored in the frontmatter of every page compiled from it, so re-ingesting an unchanged source writes nothing.

---

7. Configuration

File: `uhb/config.yaml` (override with `UHB_CONFIG`). When absent, the built-in defaults apply — which is why `uhb config list` works on a fresh clone.

```
uhb config get routing.default
uhb config set routing.default deepseek/deepseek-flash
uhb config set routing.retries 3
```

Two guardrails: only the `providers` and `routing` sections are writable, and any key whose last segment looks like a secret (`api_key` , `token` , `password` , `secret`) is **refused**. A credential value never enters a file: the configuration carries the **name** of the variable, the value stays in the environment.

Recognised variables:

Variable	Effect
`UHB_CONFIG`	path of the configuration file
`UHB_LLM_BASE_URL`	OpenAI-compatible endpoint for direct calls (default: local Ollama)
`UHB_LLM_MODEL`	model used by those direct calls
`UHB_LLM_TOKEN`	optional token (named, never stored)
`UHB_LLM_REASONING`	`none` by default: extraction does not need deliberation
`UHB_MODES_DIR`	extra mode-profile directory
`UHB_ROUTER_CATALOG`	alternative routing catalog
`UHB_HERMES_BIN`	harness binary (name, path, or argv prefix)

---

8. The second brain (Hermes Home Vision)

UHB reads the local infrastructure when it is there. Three services, four calls, no new dependency:

Service	Call	Measured answer
RAG (8200)	`GET /sante` · `POST /search {"question","k"}`	4,434 fragments, `intfloat/multilingual-e5-base` · ranked passages
SiYuan (6806)	`POST /api/system/version` · `POST /api/export/exportMdContent {"id"}`	v3.8.6 · a document's markdown
MARI (55553)	`GET /status`	`{"status":"ok","signalsEmitted":2}`

```
uhb brain status            # ● / ○ per service, with detail and latency
uhb brain status --strict   # exit 1 when a service is down (cron-friendly)
uhb brain search "How does the MARI bridge work?" -k 3
uhb brain read 20261109131511-donr63p   # export one SiYuan document as markdown
```

Rules of conduct:

- `brain status` never raises: a stopped service is a **state** (○), not a command failure.
- `brain search` and the SiYuan read **raise** (`SecondBrainError`) instead of returning an empty list — an empty list would read as "your second brain has nothing", which is false when the service is merely unreachable.
- Every call has its timeout (2 s for a probe, 30 s for a search): a hung service does not hang the caller.

- SiYuan needs a token to read content: `SIYUAN_TOKEN`, read by name, sent as a header, never logged nor kept on the object.
- Paths returned by the RAG are **raw** (they come from its index); UHB shortens them to `~` for display but does not normalise them. To be handled at the source the day an index would be published.

Installing the services

Path	Command	What is fetched
Docker	`docker compose up -d`	`uhb`, `rag` and `b3log/siyuan:v3.8.10` images
Script	`./install_services.sh` (or `install_services.ps1`)	SiYuan, sha256 verified against `SHA256SUMS.txt`
Manual	`python -m uhb.services.install --with rag --yes`	the RAG dependencies (~1.5 GB: torch, faiss, flashrank)
MARI	`… --with mari`	nothing to fetch: the bridge ships in `uhb/services/mari/bridge`, its framework is a submodule (MIT) — `git submodule update --init`

The repository is a monorepo: the RAG code and the MARI bridge live inside it; the MARI framework and SiYuan do not (an MIT submodule and a verified download).

`uhb brain start --dry-run` shows what would be launched and from where; `uhb brain start` launches what is installed and prints the pids. Nothing downloaded enters git (`uhb/bin/` is ignored, never packaged), and the RAG index is not shipped: it is made of your notes.

To put the `9+ UHB` badge in a dashboard, `uhb ui` prints the markup and the path of `UI_INTEGRATION.md`: scoped CSS, no hard-coded origin, loopback-only CORS.

9. Updating

```
uhb update --dry-run   # fetch and report, move nothing
uhb update             # git pull --ff-only, then prove it
```

`uhb update` refuses a dirty tree, has no `--force` and no `reset`, and stops if the branch has diverged: it does not merge, it does not rebase. It reports the commit before and after.

It **restarts no service**: UHB ships no supervisor. systemd, Task Scheduler or your terminal owns the process.

---

10. Troubleshooting

`uhb status` shows `MISSING` for a provider

The named environment variable is not loaded in this shell. Nothing is broken: the router will skip that provider and continue down the chain.

`AllProvidersFailed` during a compile

Every candidate failed. Read the attempts: `skipped` means "key absent", `failed` carries the exception class. With no key configured, the chain ends on the local model.

`ModeRequired`

A write path was called without a mode. Pass one explicitly (`get_mode("dev")`): that is the guarantee that no write happens without a policy.

`ModeViolation: read-only`

The chosen mode does not write. Use `dev` to compile, or a profile you wrote yourself.

• `siyuan` but reading a document fails

SiYuan answers `/api/system/version` without authentication, but refuses content: `SIYUAN_TOKEN` is missing in this shell. The probe stays green because it only checks the version; `uhb brain read` then fails with `Auth failed [session]` .

The PDF does not download

`GET /docs/manuel.pdf` only exists if the package was installed with its data (`pip install uhb-harness` , or `pip install -e .` from a checkout). `uhb docs` prints the real local path.

---

11. Known limits

- The compile was measured on a three-sentence note, not on a corpus.
- Only DeepSeek is verified against its vendor's real catalog; Kimi, OpenAI and Anthropic remain declared, not proven.
- The Harness Maxxing and OpenClaw adapters declare the contract; they have not been run against those products.
- A harness started as a subprocess sits outside the gate: UHB protects its own writes, not those of a third-party process.

---

12. Licence

Two licences, one per layer:

- the specification (`SPEC.md` , `ARCHITECTURE.md` , `ADAPTERS.md` , `NON-GOALS.md` , `README.md`) is **MIT**;
- the implementation (`uhb/`) is **Fair Source, FSL-1.1-MIT**: read it, run it, modify it, use it commercially; you may not resell it as a competing hosted service. Each release becomes MIT two years after publication.

Author: Thomas Leroyer (Searching Murphy).