

enterprise-okf-ai Handbook

Operational and technical manual for onboarding, running, validating, and maintaining this repository.

Start Here

Who this handbook is for

- AI/ML engineers implementing or extending the pipeline

- con

****New to OKF itself, not just this repo?**** This handbook assumes you

alrea

Role-based navigation

- New contributor path: [Tutorial lane](#tutorial-lane-first-successful-run) -> [Reference lane](#reference-lane-llookup-details)

- Ope
g)

Tutorial Lane (first successful run)

Goal: run the project end-to-end and produce validated artifacts.

Step 1: install dependencies

```
uv sync --extra dev --frozen
```

cp .e

Step 2: run quality gates

```
UV_CACHE_DIR=.uv-cache make check
```

Expected result:

- ruff

Step 3: run strict end-to-end pipeline

```
UV_CACHE_DIR=.uv-cache make run-e2e
```

Expected summary output:

- `val

Artifacts produced:

- `arti

How-to Lane (common operational tasks)

Build an OKF bundle from source docs

```
uv run enterprise-okf-ai build-okf examples/enterprise_docs okf_bundle
```

`retrieve-search`/`agent-ask` below always read the default paths from

`Setti

Validate bundle integrity

```
uv run enterprise-okf-ai okf-validate --okf-dir okf_bundle
```

Build graph artifacts

```
uv run enterprise-okf-ai graph-build --okf-dir okf_bundle
```

Build the vector index

```
uv run enterprise-okf-ai index-build --okf-dir okf_bundle
```

Required before `retrieve-search`/`agent-ask` ? they read from this

persis

Run retrieval query with traces

```
uv run enterprise-okf-ai retrieve-search "Which API updates order status?" --with-trace
```

Run agentic Q&A

```
uv run enterprise-okf-ai agent-ask "Which API updates order status and what does it depend on?"
```

Run agent benchmark evaluation

```
uv run enterprise-okf-ai agent-eval \
```

--b

Start runtime services

```
make run-api
```

make

Generate handbook PDF

```
make handbook-pdf
```

Reference Lane (lookup details)

Glossary

- OKF-style bundle: markdown concept files with YAML frontmatter.

- Con

Core frontmatter contract (this repo's enterprise profile)

The OKF v0.1 spec itself requires only ``type``. This repo's own

gener

See [``docs/05-frontmatter-and-fields.md``](docs/05-frontmatter-and-fields.md)

for th

Configuration reference

Runtime settings are loaded through ``enterprise_okf_ai.core.settings.Settings``.

Important settings:

- ``okf``

Config files:

- ``con``

Module map

Canonical package: ``src/enterprise_okf_ai/``

- ``ingestion``: multi-format parsing + normalization

- ``okf``

Compatibility package:

Output artifacts and interpretation

Primary outputs from strict E2E:

Key current metrics (from latest strict run):

Community and repository health files

- CONTRIBUTING.md

Explanation Lane (why the system is designed this way)

Why OKF-style markdown

- portable and versionable in git

Why strict validation before retrieval/agent

- prevents broken links from degrading graph and retrieval quality

Why hybrid retrieval instead of vector-only

- vector retrieval captures semantic similarity

Why explicit unsupported-answer logic

- reduces confident hallucinations in weak-evidence scenarios

- prov

Troubleshooting

CLI command not found

If `enterprise-okf-ai` is unavailable in shell path:

```
PYTHONPATH=src uv run --no-sync python -m enterprise_okf_ai.cli.main --help
```

Failing E2E due to stale artifacts

Delete target artifact folder and rerun strict pipeline:

```
rm -rf artifacts/e2e_real_run
```

UV_CA

GitHub CLI authentication issues

```
gh auth status
```

gh au

Non-fatal dependency warning in tests

Current tests may show a `chromadb` deprecation warning (`asyncio.iscoroutinefunction`). The run is still successful.

Documentation Maintenance Policy

- Treat `README.md` as onboarding + quick-run surface.

- Trea

Official References and Sources

Project references:

- Goo

Official docs for stack used here:

- uv: [uv](#)

Documentation quality guidance used in this handbook rewrite:

- Gith