Review the system. Evaluate the source.

Python trading framework and TA strategy engine.
MetaTrader FX/CFD execution through MetaApi; historical replay and account-level controls.

Start at falbi.xyz/review/.
Inspect a completed synthetic replay, select a recorded trade, follow its lifecycle and explore the implementation map. No login, broker access or installation is needed for the public review.

WHAT IS OFFERED

FALBI is a privately owned, pre-revenue software asset. The source edition includes the built-in TA Trader layer, historical replay, account-aware risk and execution, durable operational state and reconciliation. The proposed acquisition covers the agreed proprietary source and transferable rights; third-party rights remain separately identified.

A USEFUL REVIEW SEQUENCE

1. Inspect output	View the 9 generated plans and 8 closed simulated executions. Check entry, stop, staged exits and recorded events. Read the classification before interpreting any result.
2. Check technical fit	Use the architecture map to choose the modules your team wants to inspect. Compare the implemented MetaTrader FX/CFD scope with your instruments, account models and integration requirements.
3. Evaluate source	Agree a controlled source inspection with the owner. With the complete agreed archive, verify its manifest, reproduce the offline demo and define any additional deployment qualification.

Contact: Vladyslav, independent FALBI owner
vvlad0066@gmail.com | falbi.xyz

A complete replay. Inspectable outputs.

Recorded offline run from the FALBI 1.1.0 buyer archive using a synthetic fixture. All frozen acceptance assertions passed.

SYNTHETIC ENGINEERING FIXTURE. The input is project-generated non-market data. This checks functionality; it is not a historical performance study, live trading result or broker-execution qualification.

Input / derived data	2,304 synthetic 5m candles; 192 derived closed 1h candles; 2,496 processed candles in total. Fixture interval: 1-9 January 2026 UTC, end-exclusive.
Strategy lifecycle	9 opportunities; 9 plans; 9 persistent setups. 8 routed orders; 8 fills; 8 closed outcomes. The ninth opportunity remains WAITING_FOR_ENTRY at the fixture cutoff.
Evidence integrity	Journal COMPLETED; 0 reported TA/risk integrity violations; 100% linkage across 8 filled signal groups. The completed DB was inspected through a read-only connection.

FIRST RECORDED ACCOUNT EXECUTION: T01

EURUSD LONG; 1h source context, 5m execution.
Entry: **1.435075** at 6 January, 08:00 UTC; stop: **1.428950**.
Stored targets: 1.448000 / 1.454350 / 1.460700.
Three simulated partial exits; final close at 09:20 UTC.
Weighted account exit: **1.452688**. Volume rounding and replay slippage distinguish account fills from the canonical plan.

STRUCTURAL RESULT SHA-256

`6224626ca6f5801f6d5d5f468993936a0fb3c788bb8233f7aa56661cc0b5729e`

The website displays a read-only projection of this completed run. Selecting a trade changes the recorded view; the browser does not execute Python, connect to a broker or place orders. Download the selected JSON/CSV outputs from the review page.

Shared strategy. Independent account checks.

The framework separates context and strategy decisions from account policy, broker transport and operational persistence. These are source locations for an informed inspection.

Configuration	**run.py; falbi.py; core/runtime/config.py** Resolve the operating mode, strategy graph and account policy. falbi.py also exposes supervision and read-only replay statistics.
TA engine	**core/pro_trader/** Closed-candle context, market structure, playbooks, arbitration, persistent setups and entry/SL/TP planning. The included strategy uses the configured base timeframe, normally 5m.
Historical replay	**core/replay/; scripts/run_replay.py** History loading, higher-timeframe resampling, replay clock, simulated ReplayBroker and canonical evidence. Inspect coverage, warmup and cost assumptions.
Routing / risk	**core/runtime/account_router.py; core/execution/** A shared signal is evaluated separately for eligible account aliases. Sizing, risk, Warden, heat and margin checks can accept different sizes or reject independently.
Execution / adapter	**core/runtime/live_broker.py; core/adapters/metaapi.py** Execution ownership and position lifecycle. Hedging child-position legs and netting partial closes are implemented paths; qualify the buyer account model explicitly.
Operational state	**core/runtime/live_state.py** Transactional SQLite, execution journals and reconciliation. The controlled runtime uses WAL, FULL synchronization and explicit transactions.

Integration boundary: exact broker symbols/specifications are configured. Independent APIs need an adapter. Automatic cross-broker discovery, exchange futures/options/rollover, shared-write horizontal deployment and hosted multi-tenant operation require separately scoped engineering.

Reproduce the demo from the agreed archive.

Use the documented Python 3.12 BASE profile from the FALBI_1.1.0/ root. Select a fresh dependency target; no broker account or credentials are required for this offline check.

1 / INSTALL THE BASE DEPENDENCIES

```
python3.12 -m pip install --target runtime/deps -r requirements.txt -c constraints-py312.txt
```

2 / VERIFY THE EXACT DEPENDENCY TARGET

```
PYTHONPATH=runtime/deps python3.12 scripts/verify_release.py --check-dependencies
runtime/deps --dependency-profile BASE
```

3 / RUN THE FROZEN SYNTHETIC FIXTURE

```
PYTHONPATH=runtime/deps python3.12 scripts/run_demo.py
```

ACCEPTANCE OUTPUTS

Require exit 0, `DEMO | COMPLETED | status=PASS`, the structural fingerprint on page 2 and Journal **COMPLETED**. The runner compares observed structural counts with a locked contract; mismatches fail the demonstration.

Inspect the result	**runtime/demo/falbi-demo-1-0-0/** demo_summary.json; demo.log; canonical_trade_journal.sqlite3; TA decision/evidence files and replay integrity reports.
Read the procedure	**FIRST_START.TXT; INSTALLATION.md; DEMO.md** Check the archive/manifest with the packaged verification tool. Use METAAPI dependencies and the LIVE guide only for separately agreed external evaluation.

The browser review is a presentation. The downloadable buyer source is the Python runtime. Full source inspection/delivery, confidentiality and allowed use follow the agreed review and transaction terms; viewing this document does not transfer IP or grant commercial use.