MH-3b ProposalWriter — Plan-Review

Projekt: Steve-TradingBot · Phase: RECON-MH-3b · Author: claude-opus-4-7[1m]
Generated: 2026-05-12 17:47 UTC · master HEAD: `26c8ab3` (MH-3a closed)
Status: **NO CODE** Plan-Review only — Operator-GO erforderlich vor MH-3b Code-Phase

1 — Live-State Verification

Check	Soll	Ist	OK
master HEAD	`26c8ab3`	`26c8ab3 mh-3a: add dry-run managed proposal engine v1`	✓
`git status`	clean	empty output	✓
Bot-Container	running	`clawbot Up 35h healthy`	✓
Bot in-container PID	29984	`29984 python3 main.py --paper`	✓
Worker-Container	running	`clawbot-worker Up 18h (unhealthy)` — Healthcheck-Drift, kein Funktionsproblem	⚠
`BINANCE_TESTNET`	true	`BINANCE_TESTNET=true`	✓
`runtime_config.json`	absent	not present	✓
`baseline_holdings.json`	absent	not present	✓
`managed_state.json`	absent	not present	✓
`state/risk_proposals/`	absent	not present	✓
Tracebacks last 300 lines	0	0 matches	✓

2 — MH-3a-Konsumiertes Surface

Komponente	Status	Bedeutung für MH-3b
`RiskProposalEngine` V1-Minimal	**committed `26c8ab3`**	Engine bleibt Source-of-Truth für payload-Format
`dry_run=True` returns `ProposalResult`	✓	Writer kann `result.payload` direkt konsumieren
`dry_run=False` raises `NotImplementedError("ProposalWriter not available until MH-3b")`	✓	**Aktivierungsfrage** für MH-3b: bleibt diese Meldung oder wird sie gelifted? Siehe §5
`risk_model_version="phase1-min-v1"` / `proposal_version=1`	**pinned**	Writer-Validator prüft Anwesenheit (G-DR-10)
`confidence.overall_score` Pflicht	**enforced**	Engine garantiert; Writer dupliziert defensive Check
`proposal_id` als UUID-String erzeugt	**via `uuid_factory`**	Writer validiert Filename-Safe-Pattern
`STABLE_PEG_BLOCKLIST` / `MainnetBlockedError`	**Engine raises vor Compute**	Writer eigene Mainnet-Defense-Layer (Defense-in-Depth)

Pattern-Referenz: `trading/baseline_holdings_writer.py` (600 LOC, RECON-2.1). Behält:

- `tempfile.mkstemp` + `fsync` + `os.replace` atomic-write
- `MainnetBlockedError` Exception
- `ValidationError` Exception
- `ApplyResult` frozen dataclass
- `ALLOWED_ENVIRONMENTS = frozenset({"testnet"})`
- `restore_from_backup` static helper

Unterschiede zum Baseline-Pattern:

- Proposal-Files sind **immutable** (per `01_foundation §3`: „NIE überschrieben, NIE gelöscht"). Kein `apply()`-with-idempotency-skip — Writer **refuses** overwrite.
- Kein `clear()` Method — Proposals werden nur via separater TTL-Expiry-Cleanup-Job (MH-9) archive-moved, nicht direkt gelöscht.
- Kein `canonical_hash` für idempotency — `proposal_id` ist der natürliche Primärschlüssel.
- Per-File-Modell statt single-config-file: jeder Apply schreibt eine neue Datei `<proposal_id>.json`.
- Kein Backup-before-mutate (keine Mutation; nur erste Erzeugung).

3 — ProposalWriter Scope-Definition

3.1 — Target Layout

```
state/risk_proposals/
  └── <proposal_id>.json     ← eine Datei pro Engine-Output, immutable
```

Eigenschaft	Wert
Default target_dir	`DEFAULT_STATE_DIR / "risk_proposals"` = `<trading>/state/risk_proposals/`
Filename-Pattern	`<proposal_id>.json` (UUID-String + `.json`)
In Container	`/home/node/.openclaw/workspace/trading/state/risk_proposals/<id>.json` (passt zu `ProposalReader.DEFAULT_PROPOSALS_DIR`)

3.2 — `proposal_id` Validierung (anti-path-traversal)

Validator-Regex (vorgeschlagen): `^[a-zA-Z0-9_-]{1,128}$` (akzeptiert UUID4 default + Engine custom-IDs).

Reject	Reason
`""` / `" "` (empty/whitespace)	invalid
`"../etc/passwd"` (path traversal)	contains `/` und `..`
`"a/b"` (path traversal)	contains `/`
`"a\\b"` (path traversal Windows)	contains `\\`
`".hidden"` (leading dot)	filesystem-hidden semantics
`"file\\x00trick"` (null byte)	injection
`"this-is-129-chars-long-..."` > 128 chars	bound length
`"id with space"`	non-canonical

Defense-in-Depth: nach Regex-Pass zusätzlich `Path(filename).name == filename` Check (PathLib normalisiert path; Identity-Check fängt Edge-Cases).

3.3 — Atomic Write Sequenz

1. **Validate payload-dict shape** (raises before IO):
   - `proposal_id` present + matches arg
   - `proposal_version == 1`
   - `risk_model_version` ∈ Whitelist (mindestens `phase1-min-v1`)
   - `confidence.overall_score` ∈ [0..1]
   - `asset`, `generated_at`, `expires_at`, `generated_by` present
2. **Mainnet defense layer** (Defense-in-Depth, unabhängig von Engine):

- Settings.BINANCE_TESTNET != True → raise `MainnetBlockedError`

3. **Ensure target_dir exists:** `os.makedirs(target_dir, exist_ok=True)` (auto-create OK — keine Mutation existierender Files)
4. **Collision check:** if `target_path.exists()` → return `ApplyResult(ok=False, error="proposal_id_collision")`. **Kein Overwrite**, **kein Backup-Move**.
5. **Atomic write:**
   - `tempfile.mkstemp(prefix=f"{proposal_id}.json.tmp.", dir=target_dir)` (same FS → `os.replace` ist atomic)
   - `json.dump(payload, f, indent=2, sort_keys=True, ensure_ascii=False)`
   - `f.flush() + os.fsync(f.fileno())` (Daten durable)
   - `os.replace(tmp, target)`
6. **sha256 berechnen** für `ApplyResult` (audit-relevant; analog Baseline-Writer)
7. **Return** `ApplyResult(ok=True, target_path, sha256, written_at)`

3.4 — Detailfragen

Frage	Antwort
Temp-file + rename?	Ja — `tempfile.mkstemp` in same dir + `os.replace`. Pattern aus Baseline-Writer übernommen.
fsync nötig?	Ja — `f.fsync()` nach `json.dump`. Schützt gegen Crash zwischen Write und Power-Loss. Directory-fsync optional (Baseline-Writer macht's nicht, wir matchen).
Permissions?	Default umask. **0644** für JSON, **0755** für Dir. Kein chmod explicit — Files sind nicht secret (kein Token-Material), Audit-Trail. Stricter chmod ist BACKLOG (operator-optional).
Backup bei overwrite?	**Nein** — keine Overwrite-Möglichkeit (Immutable per spec).
Darf overwrite vorkommen?	**Nein**. Writer refuses mit `proposal_id_collision` Error.
canonical_hash nötig?	**Nein** für Filing-Idempotency (`proposal_id` ist unique key). **sha256** des geschriebenen Body **ja** für Audit-Result.
created_at / written_at metadata?	`written_at` wird im `ApplyResult` zurückgegeben (separate Spur vom `generated_at` im Payload). **Kein** in-payload-mutation — payload bleibt 1:1 wie Engine es liefert (immutable).

4 — Boundary-Contract

4.1 — Forbidden Imports (AST-pinned)

```
main · command_worker · live_trade · paper_trade · risk_manager
```

`proposal_engine` darf NICHT importiert werden (Writer ↛ Engine — sonst Zyklus). Writer akzeptiert plain `dict`-payload.

4.2 — Forbidden Source-Tokens (Grep-pinned)

- ccxt Order-API: `create_market_*`, `create_order`, `cancel_order`, `edit_order`, `create_limit_order`, `cancel_all_orders`
- File-Targets außerhalb risk_proposals: `managed_state.json`, `baseline_holdings.json`, `runtime_config.json`, `live_portfolio.json`, `.env`
- DB-Kopplung: `INSERT INTO`, `execute(`, `cursor`
- Network: kein ccxt-Import, kein `requests`, kein `urllib`

4.3 — Was Writer NICHT macht

Tabu	Begründung
managed_state.json schreiben	MH-4 Scope
baseline_holdings.json schreiben	RECON-2.1 hat eigenen Writer
risk_proposals/*.json löschen / archivieren	MH-9 TTL-cleanup-Job
DB-Insert	MH-6 Worker-Handler
audit_events Insert	MH-6 Worker-Handler
command_worker.py touchen	MH-6
main.py / live_trade.py / paper_trade.py touchen	MH-7
Bot-Wiring	MH-7
Exchange/API write	aus Prinzip — Writer ist filesystem-pure
Mainnet	`ALLOWED_ENVIRONMENTS = {"testnet"}` + `MainnetBlockedError` raise

5 — Integration mit MH-3a

Option A — Engine unverändert, Writer standalone (empfohlen)

- `proposal_engine.py` **bleibt byte-identisch**. dry_run=False raised weiter `NotImplementedError`.
- `proposal_writer.py` ist unabhängig instanziierbar.
- Caller-Pattern (für MH-6 worker handler):

```
result = engine.generate(asset, operator_id, dry_run=True)
if result.ok:
    apply_result = writer.apply(result.payload)
```

- **Pro:** Maximaler Scope-Lock; Engine bleibt byte-identical; Writer ist isoliert testbar; MH-3a Closure-Pin bleibt 1:1 gültig.
- **Con:** dry_run=False ist weiterhin „inaktiv". Erst MH-6 worker handler aktiviert den Full-Flow.

Option B — Engine bekommt optional `writer`-Konstruktor-Param

- Engine bekommt 1 neuen optionalen Kwarg: `proposal_writer: Optional[ProposalWriter] = None`.
- `dry_run=False` Pfad: wenn `writer` injected → `writer.apply(payload)`; sonst weiter `NotImplementedError`.
- ProposalResult bekommt befüllte `proposal_path` + `written_audit_event=False` (audit-event bleibt Worker-Job).
- **Pro:** dry_run=False wird aktiv. Engine + Writer composable.
- **Con:** MH-3a Engine wird angefasst (kleine Mutation: 1 ctor-Param + 1 if-Branch). Engine-Tests müssen angepasst werden. AST-Boundary-Liste muss `proposal_writer` zulassen.

Option C — Engine ungenagelt, neue Compose-Methode in Writer

- Writer bekommt `apply_from_engine_result(result: ProposalResult)` Convenience-Methode.
- Engine bleibt unverändert.
- **Pro:** MH-3a Closure stays.
- **Con:** Subtile zirkuläre Type-Abhängigkeit (Writer kennt ProposalResult), Kohärenz schlechter als Option A.

Empfehlung: Option A

1. Klare Trennung von Computation vs. Persistence
2. Engine MH-3a closure bleibt zementiert
3. MH-6 worker handler ist die natürliche Orchestrierungs-Stelle
4. Tests für „dry_run=True schreibt nichts" bleiben byte-identisch — keine Re-Run-Diskussion

Dry-Run-Persistenz-Guarantee Test (Option A):

- `test_dry_run_true_engine_does_not_call_writer`: Mock-Writer in Test; Engine mit dry_run=True ruft Writer nie auf (Writer.apply call_count == 0). Bestätigt MH-3a Contract bleibt.
- Engine-Tests bleiben unverändert; kein NotImplementedError-Test-Anpassen nötig.

6 — Test-Plan

Test-Klasse	Anzahl	Inhalt
ProposalWriterAtomicWriteTests	4	tempfile + replace; partial-write-on-crash recoverable; sha256 stable; round-trip Reader→Writer
ProposalWriterDirCreationTests	2	absent dir auto-create; existing dir reused
ProposalIdValidationTests	8	empty / whitespace / `..` / `/` / `\\` / leading dot / null byte / 129-chars rejected
ProposalWriterCollisionTests	3	existing file refuses overwrite; refused even on identical content; refused with sha256-different content
PayloadValidationTests	8	proposal_id mismatch; missing proposal_version; wrong proposal_version; missing risk_model_version; missing confidence.overall_score; score out of range; missing asset; missing generated_at
MainnetBlockedTests	3	BINANCE_TESTNET=False raises; missing setting raises; raised before any file IO
EnvironmentAllowlistTests	2	only `testnet` ALLOWED_ENVIRONMENTS pinned; future-mainnet refused
ApplyResultTests	4	ok=True populates target_path + sha256; ok=False populates error; frozen dataclass; written_at ISO-8601 UTC
RestoreFromBackupTests	3	optional helper for symmetry with Baseline-Writer; tests existence + no-raise contract
ReaderRoundTripTests	3	Writer.apply(payload) → ProposalReader.read(id) yields equivalent ProposalSnapshot; identity check via raw-payload-comparison
IntegrationTests (Option A)	2	end-to-end: Engine.generate(dry_run=True) → Writer.apply(result.payload); Mock-Writer test pins dry_run=True never calls writer
BoundaryASTTests	5	forbidden imports (5 modules); no order API in source; no foreign state file refs; no DB tokens; no proposal_engine import
SourceGrepTests	4	no `.write_text` outside atomic helper; no `os.unlink` (proposals immutable); no `shutil.copy` of risk_proposals (no backups); no path concat with `..`
NoRealNetworkTests	1	sanity — Writer doesn't import network modules
Total	**~52 Tests**	

7 — Betroffene Dateien (Final-Inventory)

Datei	Status	Phase
`trading/proposal_writer.py`	**NEU**	MH-3b
`trading/tests/test_proposal_writer.py`	**NEU**	MH-3b
`trading/proposal_engine.py`	unverändert	MH-3a closure intact
`trading/tests/test_proposal_engine.py`	unverändert	MH-3a tests intact
`trading/proposal_reader.py`	unverändert	MH-2 reader intact
`trading/tests/test_proposal_reader.py`	unverändert	MH-2 tests intact
`trading/baseline_holdings_writer.py`	unverändert	RECON-2.1 reference (pattern source)
0 PHP touches		
0 DB-Migration		
0 docker cp		
0 Bot/Worker restart		

Erwartete Größenordnung: ~350-450 LOC Writer + ~700-850 LOC Tests = ~1100-1300 LOC total (analog MH-3a 1307 LOC).

8 — Stop-Regeln MH-3b

ID	Stop wenn…
MH-3b-SR-1	Writer schreibt eine andere Datei als `state/risk_proposals/<proposal_id>.json`
MH-3b-SR-2	Writer überschreibt eine bestehende Proposal-Datei
MH-3b-SR-3	Writer löscht / verschiebt eine bestehende Proposal-Datei
MH-3b-SR-4	Writer schreibt `managed_state.json` oder `baseline_holdings.json` oder `runtime_config.json`
MH-3b-SR-5	Writer emittet `INSERT INTO commands` oder `INSERT INTO audit_events`
MH-3b-SR-6	Writer akzeptiert `proposal_id` mit `/`, `\\`, `..`, NUL-Byte, > 128 chars, oder leading dot
MH-3b-SR-7	Writer schreibt mit Settings.BINANCE_TESTNET != True (Mainnet-Block-Layer-Bypass)
MH-3b-SR-8	Writer ruft ccxt-API auf (`create_*`, `fetch_*`, `cancel_*`, etc.)
MH-3b-SR-9	AST-Boundary-Test schlägt fehl (Import von main / command_worker / live_trade / paper_trade / risk_manager / proposal_engine)
MH-3b-SR-10	Tests treffen echtes Filesystem außerhalb von `tempfile.mkdtemp` Sandbox
MH-3b-SR-11	Atomic-Write-Test schlägt fehl (tmpfile sichtbar nach erfolgreichem replace)
MH-3b-SR-12	Reader-Round-Trip-Test schlägt fehl (Writer-Output nicht von ProposalReader.read() parsbar)

9 — Backup / Restart-Bedarf

Aktion	Pflicht?
pg_dump GUI-DB	**NEIN — kein DB-Touch**
`live_portfolio.json` snapshot	**NEIN**
`.env` snapshot	**NEIN**
`state/` snapshot	**NEIN — neue Files only, kein Mutation an existierenden**
Health-Snap	**NEIN**
Memory-Sicherung	**NEIN — Closure-Pin reicht**
Bot-Restart	**NEIN — kein Wiring, kein Bot-Code-Touch**
Worker-Restart	**NEIN — Worker-Handler ist MH-6**
docker cp	**NEIN**

→ **MH-3b ist Code+Test+Commit auf master.** Analog MH-2 / MH-3a.

10 — Identifizierte Risiken

#	Risiko	Severity	Mitigation
R1	**Filesystem-Atomarität nicht garantiert wenn target_dir auf anderem Filesystem als TMPDIR** — `tempfile.mkstemp` mit `dir=target_dir` löst das (gleiche FS)	LOW	Override `dir=` explicit auf target_dir; identisches Pattern zu Baseline-Writer
R2	**proposal_id collision** — UUID4 hat 2^122 Entropie, Kollision praktisch unmöglich, aber theoretisch möglich	LOW	Collision-Check vor Write + klarer Error; Tests pinnen das Verhalten
R3	**Concurrent writes mit gleicher proposal_id** — race condition: zwei Caller validieren `not exists`, beide schreiben	MEDIUM	`tempfile.mkstemp` hat `O_CREAT | O_EXCL`; `os.replace` ist atomic. Pragmatisch: in MH-3a sind nur 1 Worker-Pfad, single-writer, kein Concurrency-Risiko in Praxis. Test mit Mock-Concurrency dokumentiert Vorsicht.
R4	**JSON-Serialisierungs-Fehler bei unbekannten Datentypen im Payload**	LOW	Engine erzeugt nur JSON-safe Werte (str/int/float/dict/list/bool). Writer prüft `json.dump` Exception → Failure-Path mit klarer Meldung
R5	**Path-Traversal-Bypass via Unicode** — z.B. RTL-Override-Zeichen, normalized vs raw NFC	MEDIUM	Strikte ASCII-Regex `[a-zA-Z0-9_-]{1,128}`; alle non-ASCII reject. + `Path(filename).name == filename` als zweite Schicht
R6	**Disk-Space-Erschöpfung** beim Write	LOW	Tempfile cleanup im except-Branch; OS raise OSError; tests können das mit Mock-Filesystem prüfen
R7	**fsync Performance auf Container-Storage** — write+fsync kann auf Bind-Mounts langsam sein	LOW	Akzeptabel: Engine-Output erfolgt selten (Operator-getriggered, nicht per Cycle). Performance-Tests nicht in Scope MH-3b
R8	**Pattern-Mirror-Drift** vom Baseline-Writer — wenn Baseline-Writer später ein Bug-Fix bekommt, Proposal-Writer driftet	LOW	Pattern-Reference-Comment im Doctring; CI-Lint-Rule wäre BACKLOG (MH-3b-FU-1)

11 — GO/NO-GO-Empfehlung

Empfehlung: GO Option A — Engine bleibt unverändert, ProposalWriter standalone.

Aspekt	Bewertung
Scope-Größe	medium (~1100-1300 LOC code+tests, ~52 Tests)
Komplexität	Pattern-Mirror Baseline-Writer; gut bekannte Atomic-Write-Semantik; **keine** State-Interdependenz
Blast-Radius	NULL — keine existierenden Files modifiziert; Engine MH-3a closure bleibt byte-identisch
Roll-Back-Cost	minimal: `git revert <commit>` + `rm -rf state/risk_proposals/` (kein DB / runtime touch)
Dependencies-Klärungen	keine — alle Pflicht-Schemas/Felder bereits in MH-3a Engine-Output
Risiko-Bewertung	LOW
Backup-Pflicht	nein
Restart-Pflicht	nein
docker cp Pflicht	nein

Pre-Implementation-Q an Operator

1. **Option A oder B?** ← **Empfehlung A** (Engine untouched, max scope-lock)
2. **proposal_id-Regex-Schwere:** strict ASCII `[a-zA-Z0-9_-]{1,128}` oder erlaubt z.B. dots in der Mitte (für proposal_id-Namespacing wie `org.eth.20260512`)? ← **Empfehlung strict ASCII no-dots**, einfacher pin
3. **fsync Directory** zusätzlich zum file-fsync? ← **Empfehlung nein** (matches Baseline-Writer)
4. **`restore_from_backup` Helper** mit liefern (Symmetrie zu Baseline-Writer) obwohl proposals immutable sind und nie restored werden müssten? ← **Empfehlung weglassen** (YAGNI)
5. **MH-3c Real-Testnet-Drill** als separate Phase oder erst Teil von MH-6? ← **Empfehlung MH-6** (Drill braucht Worker-Handler-Pfad, der Engine+Writer orchestriert)

Subschnitt

Phase	Inhalt	Scope
MH-3b (jetzt)	`proposal_writer.py` + Tests, **Engine untouched**	atomic
MH-3c (später)	Real-Testnet-Drill — empirisch Engine+Writer gegen echtes Testnet	manual operator-drill
MH-4 (später)	`ManagedStateWriter` + Two-File-Atomic Promote/Release	G-DR-14
MH-5 (später)	Filament Multi-Step-Wizard	UX-2
MH-6 (später)	Worker-Handler für `managed.*` Commands (Engine + Writer + DB-Cache + audit_events)	orchestration
MH-7 (später)	Bot-Wiring (universe / balance_provider / execute_buy / drift-detection)	**Restart-Pflicht**

STOP vor Implementierung. Erwarte Operator-GO mit Option-A/B Wahl + Pre-Implementation-Q1-Q5-Approval.