MH-4 PHP Services + DB-Cache-Tables — Plan-Review

Projekt: Steve-TradingBot · Phase: RECON-MH-4 · Author: claude-opus-4-7[1m]
Generated: 2026-05-12 18:52 UTC · master HEAD: `62a08f4` (MH-3b closed)
Status: **NO CODE** Plan-Review only — Operator-GO erforderlich vor MH-4 Code-Phase

1 — Live-State Verification

Check	Soll	Ist	OK
master HEAD	`62a08f4`	`62a08f4 mh-3b: add immutable proposal writer for risk proposal files`	✓
`git status`	clean	empty output	✓
Bot in-container PID	29984	`29984 python3 main.py --paper`	✓
Worker Host PID	338185	running (Healthcheck-Drift, kein Funktionsproblem)	⚠
cmd 13	cancelled	`13 | apply_baseline_holdings | cancelled`	✓
`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 (test-only via tempfile sandbox)	✓
Tracebacks last 200 lines	0	0 matches	✓

2 — Artefakt-Konsum (MH-0.5 → MH-3b)

Komponente	Status	Bedeutung für MH-4
MH-1: 8 `managed.*` CommandTypes in `CommandTypeRegistry`	`ea11637`	Services nutzen `CommandTypeRegistry::makeCommandPayload()` für Validierung
MH-2: `ManagedStateReader` + `ProposalReader` dormant	`50cc5c2`	Worker (MH-6) konsumiert; Service touched NICHT
MH-3a: `RiskProposalEngine` V1-Minimal, dry_run-only	`26c8ab3`	Worker (MH-6) calls engine; Service touched NICHT
MH-3b: `ProposalWriter` immutable atomic no-overwrite	`62a08f4`	Worker (MH-6) calls writer; Service touched NICHT
Worker-Handler `managed.*`	**nicht vorhanden**	MH-6
Filament `ManagedHoldings` Page / Wizard	**nicht vorhanden**	MH-5
`managed_proposals` / `managed_assets_history` Tables	**nicht vorhanden**	**MH-4 Migration**
`ProposalService` / `ManagedStateService` PHP	**nicht vorhanden**	**MH-4 Code**

Pattern-Referenz: `gui/app/Services/Apply/Baseline/ApplyBaselineService.php` (existing, RECON-2.3). Behält:

- `DB::transaction` Atomicity
- INSERT `commands` + INSERT `audit_events` Sequence
- Hard-restrict `environment = 'testnet'`
- Idempotency-Key-Pattern
- Boundary contract docstring (never touches state files, never spawns worker)
- `AuditMetadataScrubber` für Token/Secret-Filter

3 — MH-4 Scope aus Roadmap (verifiziert)

Canonical MH-Phase-Tabelle (aus `00_overview.md §4`)

MH-Phase	Inhalt	Restart?	Migration?	Mainnet?	Risk
MH-3 ✅	Engine (MH-3a) + Writer (MH-3b)	no	no	no	medium
MH-4	**PHP `ProposalService` + `ManagedStateService` + Tests**	**no**	**yes (DB-Cache-Tabellen)**	no	medium
MH-5	Filament Multi-Step Wizard	no	no	no	medium
MH-6	Bot-Worker Handler für 8 Command-Types	no	no	no	medium
MH-7	Bot-Side Wiring	**yes**	no	no	high

Was MH-4 GENAU liefert

1. **DB-Migration:** 2 neue Tabellen (`managed_proposals`, `managed_assets_history`)
2. **Eloquent Models:** `ManagedProposal`, `ManagedAssetHistory`
3. **`ProposalService`** mit 3 Builder-Methoden:
   - `createRequestCommand(asset, userId, intent)` → `request_managed_proposal`
   - `createApproveCommand(proposalId, userId, variant, overrides, hardConfirm)` → `approve_managed_proposal`
   - `createRejectCommand(proposalId, userId, reason)` → `reject_managed_proposal`
4. **`ManagedStateService`** mit 3 Builder-Methoden:
   - `createPauseCommand(asset, userId)` → `pause_managed_asset`
   - `createResumeCommand(asset, userId)` → `resume_managed_asset`
   - `createReleaseCommand(asset, userId, strategy)` → `release_managed_asset`
5. **~30 PHPUnit Tests** (Builder + Idempotency + Validator + Migration up/down)

Was MH-4 NICHT liefert (BACKLOG für andere MH-Phasen)

Tabu	Phase
Filament Page / Wizard / Actions	**MH-5**
Worker-Handler `_handle_request_managed_proposal` etc.	**MH-6**
Engine-Aufruf (ProposalEngine.generate)	**MH-6** (Worker)
JSON-Writes (managed_state.json, risk_proposals/)	**MH-6** (Worker)
`flag_managed_drift` (Bot-Self-Emit)	**MH-7** (Bot-Side per-cycle Hook)
`dry_run_promote_managed` (optional in MH-4 oder MH-5 Wizard)	tbd
audit_snapshot files `state/audit_snapshots/<id>.json`	**MH-6** (Worker SM-6)
Two-File-Atomic (managed_state + baseline_holdings)	**MH-6** (Worker)

Service-Verantwortlichkeiten (per `05_commandbus_worker §1` Pipeline)

```
Operator-Klick (GUI / MH-5)
   ↓
Filament Action (admin-only, MH-5)
   ↓
Service.createXxxCommand()                          ← MH-4
   ↓
DB::transaction:
   1. INSERT commands (status='pending', idempotency_key='mh:...')
   2. INSERT audit_events (managed.asset_proposal_requested, etc.)
   3. INSERT managed_proposals (für proposal-create: state='proposal_pending')
   ↓
Worker --once / Daemon                              ← MH-6
   ↓
claim_next() → _handle_<type>() →
   - load Readers (Baseline / Managed / Proposal)
   - call Engine.generate / Writer.apply / ProposalReader.read
   - UPDATE managed_proposals (state='risk_proposed', proposal_json=...)
   - INSERT managed_assets_history
   - emit audit_event managed.* (succeeded)
   - write audit_snapshot file
```

4 — Konfliktcheck Sonder-Anforderungen

Anforderung	MH-4-Touch	Verdict
Q-MH-15 Hybrid C (JSON=SoT, DB=Cache)	Service schreibt NUR DB (commands + audit_events + managed_proposals.initial_row). Niemals JSON. Worker (MH-6) macht JSON-first-then-DB-update.	**konform**
G-DR-14 Two-File-Atomic	Service touched NIE managed_state.json oder baseline_holdings.json. Pattern gilt für Worker (MH-6).	**kein Konflikt**
Proposal-Immutability (MH-3b)	Service INSERTs `managed_proposals` mit `proposal_json=NULL` (initial); Worker fillt es nach Engine-Run. Niemals UPDATE auf einer abgeschlossenen Proposal-Row mit `decided_at IS NOT NULL`. Service-Validator prüft das in `createApproveCommand` / `createRejectCommand`.	**konform**
Kein managed transfer	Service-Layer berührt nie baseline_holdings.json. `release_managed_asset` schreibt nur `commands.payload`; Worker setzt später `frozen` zurück.	**konform**
Kein synthetic entry apply	synthetic_entry ist im `proposal_json` Feld (DB-cache, JSON-Spalte). Service NICHT setzen; Worker setzt es nach Engine-Output.	**konform**
Kein Worker-Handler	MH-4 PHP-only. command_worker.py bleibt unverändert.	**konform**
Kein Bot-Wiring	main.py / paper_trade.py / live_trade.py bleiben unverändert.	**konform**
Kein Mainnet	Service hard-restricts `environment = 'testnet'` in jedem Builder (Pattern aus ApplyBaselineService).	**Defense-in-Depth Layer 5**

5 — Identifizierte Risiken

#	Risiko	Severity	Mitigation
R1	**Migration auf live GUI-DB** — `tradingbot_gui` hostet commands, audit_events, baseline, runtime; jede Migration berührt diese DB	**MEDIUM**	(a) Nur additive `CREATE TABLE` (keine ALTER). (b) `down()` mit `dropIfExists` rollback-able. (c) pg_dump Backup vor Migration-Run. (d) Migration läuft nur in **testnet-DB** (Mainnet-DB existiert nicht — Mainnet ist hard-blocked). (e) Schema-only, keine Backfill-Daten.
R2	**State-Enum-Drift PHP↔Bot** — die `state`-Spalte in `managed_proposals` muss enum-konsistent sein zur Bot-Side State-Machine	MEDIUM	Service definiert PHP-Konstante `ManagedProposalStates::ALL` parallel zu Bot-Side `KNOWN_STATES` in `managed_state_reader.py`. AST-Parity-Test in MH-4 prüft beide Listen stimmen überein (analog G6.5 Parity-Test).
R3	**Idempotency-Key approve/reject Race** — Operator klickt approve und reject simultan; beide schreiben mit gleichem Key `mh:decide:<proposal_id>`	LOW	DB-Unique-Constraint auf `commands.idempotency_key` (existiert bereits). Service-Code nutzt `INSERT ... ON CONFLICT (idempotency_key) DO NOTHING` Pattern. Test simuliert Race.
R4	**Proposal-not-found bei approve/reject** — User sieht stale GUI; Proposal ist bereits expired / decided	LOW	Service::createApproveCommand validiert `managed_proposals.state IN ('risk_proposed') + decided_at IS NULL` vor Command-Insert. Throws `ProposalAlreadyDecidedException` sonst.
R5	**Wizard Hard-Confirm-Pattern (Q-MH-11)** — `<asset>:<variant>` String muss matchen	MEDIUM	Service::createApproveCommand bekommt `hardConfirmString`-Param; validiert genau-match gegen `<asset>:<variant>` (case-sensitive). Throws `HardConfirmMismatchException`. Pattern analog G10-3b ApplyProfile.
R6	**Confidence-Override-Flag bei niedrigem Score** (G-SR-9)	LOW	Service::createApproveCommand prüft `confidence.overall_score < 0.50` → benötigt `confidence_override=true` Param. Sonst `ConfidenceOverrideRequiredException`.
R7	**Audit-Event-Prefix-Drift** (G-DR-11)	LOW	Alle managed.* audit-events nutzen `'managed.'` Prefix; Test pinnt das.
R8	**AuditMetadataScrubber-Bypass**	LOW	Service nutzt AuditMetadataScrubber (existing, G9-2) für metadata-Felder. Pattern aus ApplyBaselineService übernommen.
R9	**Migration-Rollback-Cost** wenn schon Daten drin sind	LOW	Initial roll-out hat 0 Rows in managed_proposals. Rollback = `dropTable`. Bei späterem Rollback nach Production-Use: separate Backup-Restore-Phase.
R10	**DB-Cache-Drift** wenn Worker JSON schreibt aber DB-INSERT fehlschlägt	MEDIUM (für MH-6)	NICHT MH-4-Scope — wird in MH-6 Worker-Handler via `managed.cache_drift_detected` audit + Retry-Logik gehandhabt. Service-side nur initial INSERT, immer in DB::transaction.

6 — Test-Plan

Test-Klasse	Anzahl	Inhalt
ManagedProposalsMigrationTest	3	up creates tables + indexes; down rolls back; idempotent re-migration
ManagedAssetsHistoryMigrationTest	3	up + down + index verification
ManagedProposalModelTest	4	model-cast (proposal_json as array); fillable; state-enum allowed values; relations to User
ManagedAssetHistoryModelTest	3	model-cast; fillable; relations
ProposalServiceCreateRequestTest	5	INSERT commands + audit + managed_proposals; idempotency; environment=testnet hardcoded; CommandType validator; payload schema
ProposalServiceCreateApproveTest	6	INSERT commands + audit; idempotency-key collision; proposal-not-found rejection; hard-confirm mismatch rejection; confidence-override required path; happy path
ProposalServiceCreateRejectTest	4	INSERT commands + audit; same idempotency key as approve; reason required; happy path
ManagedStateServiceCreatePauseTest	3	INSERT commands + audit; distinct key per click; happy path
ManagedStateServiceCreateResumeTest	3	analog pause
ManagedStateServiceCreateReleaseTest	4	strategy-param required; INSERT + audit; happy path; testnet-only
ManagedProposalStatesEnumTest	2	enum values pinned (10 states); parity with Bot-side KNOWN_STATES (via fixture / hardcoded mirror)
MainnetGuardTest	3	every Service hard-restricts `environment='testnet'`; Service rejects payload with environment='mainnet'
AuditPrefixTest	2	every audit_event has `managed.` prefix; no baseline.* / runtime_config.* / apply_profile_testnet.* prefix bleed
DbTransactionAtomicityTest	3	failed INSERT mid-transaction rolls back; no orphan commands without audit_event; no orphan managed_proposals without command
Total	**~48 Tests**	(etwas über Roadmap-Schätzung ~30; Coverage-Reserve für Edge-Cases)

7 — Betroffene Dateien (Final-Inventory)

Neue Files

Datei	Status	Kategorie
`gui/database/migrations/2026_05_xx_xxxxxx_create_managed_proposals_table.php`	**NEU**	Migration
`gui/database/migrations/2026_05_xx_xxxxxx_create_managed_assets_history_table.php`	**NEU**	Migration
`gui/app/Models/ManagedProposal.php`	**NEU**	Eloquent Model
`gui/app/Models/ManagedAssetHistory.php`	**NEU**	Eloquent Model
`gui/app/Enums/ManagedProposalState.php`	**NEU**	PHP Enum (10 states)
`gui/app/Services/Managed/ProposalService.php`	**NEU**	Builder Service
`gui/app/Services/Managed/ManagedStateService.php`	**NEU**	Builder Service
`gui/app/Exceptions/ProposalAlreadyDecidedException.php`	**NEU**	Custom Exception
`gui/app/Exceptions/HardConfirmMismatchException.php`	**NEU**	Custom Exception
`gui/app/Exceptions/ConfidenceOverrideRequiredException.php`	**NEU**	Custom Exception
`gui/tests/Feature/ManagedProposalsMigrationTest.php`	**NEU**	Test
`gui/tests/Feature/ProposalServiceTest.php`	**NEU**	Test
`gui/tests/Feature/ManagedStateServiceTest.php`	**NEU**	Test
`gui/tests/Unit/ManagedProposalStatesEnumTest.php`	**NEU**	Test

Unverändert

- `gui/app/Services/CommandTypeRegistry.php` (8 managed.* Types bereits aus MH-1)
- `gui/app/Services/Apply/Baseline/ApplyBaselineService.php` (Pattern-Source)
- `gui/app/Services/Apply/ApplyProfileService.php` (Hard-Confirm Pattern-Source)
- `gui/app/Services/ConfigProfile/AuditMetadataScrubber.php` (gewieder-verwendet)
- `gui/app/Models/Command.php` / `gui/app/Models/AuditEvent.php`
- **0 Bot-Side Touches** (trading/*.py)
- **0 docker cp**
- **0 Bot/Worker Restart**

Erwartete LOC: ~600 PHP services + ~150 migrations/models + ~50 enum + ~900 tests = **~1700 LOC total**.

8 — DB-Migrations-Plan

Migration 1: `managed_proposals`

```
CREATE TABLE managed_proposals (
    proposal_id           UUID PRIMARY KEY,
    asset                 VARCHAR(32) NOT NULL,
    state                 VARCHAR(32) NOT NULL,
    proposal_json         JSON,
    generated_at          TIMESTAMP,
    expires_at            TIMESTAMP,
    decided_at            TIMESTAMP NULL,
    requested_by_user_id  BIGINT NULL REFERENCES users(id) ON DELETE SET NULL,
    decided_by_user_id    BIGINT NULL REFERENCES users(id) ON DELETE SET NULL,
    decision              VARCHAR(16) NULL,
    created_at            TIMESTAMP DEFAULT now(),
    updated_at            TIMESTAMP DEFAULT now()
);
CREATE INDEX idx_managed_proposals_state    ON managed_proposals(state);
CREATE INDEX idx_managed_proposals_asset    ON managed_proposals(asset);
CREATE INDEX idx_managed_proposals_expires  ON managed_proposals(expires_at);
```

Migration 2: `managed_assets_history`

```
CREATE TABLE managed_assets_history (
    id            BIGSERIAL PRIMARY KEY,
    asset         VARCHAR(32) NOT NULL,
    state_from    VARCHAR(32),
    state_to      VARCHAR(32) NOT NULL,
    actor         VARCHAR(64) NOT NULL,
    event_type    VARCHAR(64) NOT NULL,
    metadata_json JSON,
    ts            TIMESTAMP NOT NULL
);
CREATE INDEX idx_managed_history_asset_ts ON managed_assets_history(asset, ts);
CREATE INDEX idx_managed_history_event    ON managed_assets_history(event_type);
```

Rollback Plan

- `php artisan migrate:rollback --step=2` → `Schema::dropIfExists('managed_proposals'); Schema::dropIfExists('managed_assets_history')`
- Pre-Migration pg_dump-Backup → Restore-Point falls Rollback nicht reicht

Migration-Workflow

1. **Pre-Migration Backup**: `pg_dump -Fc -d tradingbot_gui > /srv/shares/backups/mh-4-pre-migration-<UTC>.pgdump`
2. `php artisan migrate --pretend` → SQL preview
3. `php artisan migrate --path=database/migrations/2026_05_xx_xxxxxx_create_managed_proposals_table.php`
4. `php artisan migrate --path=database/migrations/2026_05_xx_xxxxxx_create_managed_assets_history_table.php`
5. Verifizieren: `\d managed_proposals` + `\d managed_assets_history` + INDEX-Check
6. Rollback-Test in **Staging-DB** (falls vorhanden) oder unittest-suite-Sandbox

9 — Stop-Regeln MH-4

ID	Stop wenn…
MH-4-SR-1	Service schreibt direkt nach `managed_state.json` / `baseline_holdings.json` / `risk_proposals/*.json`
MH-4-SR-2	Service spawned Worker oder triggert `command_worker.py --once`
MH-4-SR-3	Service akzeptiert `environment != 'testnet'`
MH-4-SR-4	Migration verändert eine existierende Tabelle (ALTER auf commands / audit_events / baseline_holdings)
MH-4-SR-5	Idempotency-Key fehlt bei approve_managed_proposal oder reject_managed_proposal
MH-4-SR-6	audit_event-Prefix ist nicht `managed.*` (Bleed in baseline.* / runtime_config.*)
MH-4-SR-7	Migration läuft auf Mainnet-DB (es gibt keine; Mainnet hard-blocked)
MH-4-SR-8	Service ruft Engine.generate() oder Writer.apply() direkt auf (das ist Worker-Job)
MH-4-SR-9	Eloquent-Model erlaubt `state` ausserhalb der 10 KNOWN_STATES
MH-4-SR-10	State-Enum-Parity-Test PHP ↔ Bot bricht
MH-4-SR-11	Service-Methode NICHT in DB::transaction (atomic INSERTs gebrochen)
MH-4-SR-12	Hard-Confirm-String nicht case-sensitive validiert

10 — Backup / Restart-Bedarf

Aktion	Pflicht?	Begründung
pg_dump GUI-DB	**JA**	durable rule `feedback_backup_before_live_actions.md` — Migration läuft auf live DB
`live_portfolio.json` snapshot	**NEIN**	kein State-Touch
`.env` snapshot	**NEIN**	keine Mutation
`state/` snapshot	**NEIN**	kein State-Touch
Memory-Sicherung	**NEIN**	Closure-Pin reicht
Bot-Restart	**NEIN**	kein Bot-Code-Touch
Worker-Restart	**NEIN**	command_worker.py unverändert
docker cp	**NEIN**	PHP läuft in GUI-Container; Files werden via Live-Volume gemounted (Laravel hot-reload)
GUI-Container Cache-Clear	**JA (sicherheitshalber)**	`php artisan config:clear && php artisan route:clear && php artisan view:clear` nach Migration

→ **MH-4 = Code+Migration+Test+Commit auf master + pg_dump-Backup.** Bot bleibt komplett unbehelligt.

11 — Kleinste sichere Code-Phase + GO/NO-GO Empfehlung

Subschnitt-Optionen

Variante	Inhalt	LOC	Tests	Risk	Migration
MH-4-monolithic	Alles zusammen: 2 Migrationen + 2 Services + 6 Methoden + Tests + Enum + Exceptions	~1700	~48	MEDIUM	ja, atomar
MH-4a ← empfohlen	NUR Migration + Eloquent Models + Enum + Migration-Tests + Model-Tests	~350	~13	LOW	ja
MH-4b (folgt)	ProposalService (3 Methoden) + Tests	~450	~15	LOW	nein
MH-4c (folgt)	ManagedStateService (3 Methoden) + Tests	~350	~10	LOW	nein
MH-4d (folgt)	DbTransactionAtomicity + MainnetGuard + AuditPrefix Tests	~150	~10	LOW	nein

Empfehlung: MH-4a — Migration + Models + Enum atomar zuerst

Begründung:

1. **Maximaler Scope-Lock**: Schema-additiv-only, kein Business-Logic-Code. Klar auditierbar.
2. **Migration ist riskiest sub-step**: separates Commit → vor Service-Logic isoliert verifizieren
3. **Eloquent Models + Enum** = passive data-shape Layer; keine `INSERT INTO commands` etc.
4. **State-Enum-Parity-Test** pinnt PHP↔Bot Konvention sofort beim ersten Commit
5. **Rollback-Cost minimal**: `migrate:rollback` + `git revert`
6. **MH-4b/c folgen sequenziell**: Services bauen auf existing Schema auf, einfacher zu reviewen

Pre-Implementation-Q an Operator

1. **MH-4a Subschnitt oder MH-4-monolithic?** ← **Empfehlung MH-4a** (max scope-lock, Migration isoliert verifizierbar)
2. **`ManagedProposalState` als PHP-Enum** (Laravel 9+ feature) oder als Plain-String-Constants Class? ← **Empfehlung PHP Enum** (typsicher, IDE-Autocomplete, Eloquent cast)
3. **Foreign Keys auf `users(id)`**: `ON DELETE SET NULL` oder `RESTRICT`? Roadmap-Schema sagt SET NULL für `requested_by_user_id` + `decided_by_user_id`. ← **Empfehlung SET NULL** (Audit-Trail durabel, User-Löschung erlaubt)
4. **Migration-Run via `php artisan migrate`** in GUI-Container oder via `--pretend` + manuell? ← **Empfehlung**: zuerst `--pretend`, dann live nach Operator-Approval auf pg_dump-Backup
5. **`managed_proposals.proposal_json` Type**: `JSON` (Postgres native) oder `JSONB`? Schema-Doc sagt `JSON`. ← **Empfehlung JSON** wie spec sagt; `JSONB` wäre nice-to-have für Indexing aber MH-4 hat keine JSON-path Queries

Scope-Größe + Risiko-Bewertung MH-4a

Aspekt	Bewertung
LOC	~350
Tests	~13
Komplexität	Pattern-Mirror existing migrations + ApplyBaselineService
Blast-Radius	LOW (additive Schema only)
Roll-Back-Cost	minimal: `migrate:rollback + git revert`
Dependencies-Klärung	Q-MH-15 + State-Enum-Liste (10 states)
Backup-Pflicht	**JA** pg_dump
Restart-Pflicht	NEIN
Mainnet-Touch	NEIN
docker cp	NEIN

STOP vor Implementierung. Erwarte Operator-GO mit Subschnitt-Wahl (MH-4a empfohlen) + Q1-Q5-Approval.