MH-4b-3 ProposalService::createRejectCommand — Plan-Review

Projekt: Steve-TradingBot · Phase: RECON-MH-4b-3 · Author: claude-opus-4-7[1m]
Generated: 2026-05-12 20:51 UTC · master HEAD: `be9cab7` (MH-4b-2 closed)
Status: **NO CODE** Plan-Review only — Operator-GO erforderlich vor MH-4b-3 Code-Phase

1 — Live-State Verification

Check	Soll	Ist	OK
master HEAD	`be9cab7`	`be9cab7 mh-4b-2: add proposal approve command service`	✓
`git status`	clean	empty output	✓
Bot in-container PID	29984	`29984 python3 main.py --paper` (unverändert seit MH-3a)	✓
Worker Host PID	(egal)	**2486135** (vorher 338185 — Watchdog-Respawn 22:04 UTC, nicht durch MH-4b-2 verursacht)	⚠
cmd 13	cancelled	`13 | apply_baseline_holdings | cancelled`	✓
`managed_proposals` rows	0	**0**	✓
`managed_assets_history` rows	0	**0**	✓
`BINANCE_TESTNET`	true	`BINANCE_TESTNET=true`	✓
`runtime_config.json`	absent	not present	✓
`baseline_holdings.json`	absent	not present	✓
`managed_state.json`	absent	not present	✓
Tracebacks last 200 lines	0	0 matches	✓

Worker-PID-Hinweis: Container `clawbot-worker` ist `restart: unless-stopped` in docker-compose. Wechsel auf 2486135 deutet auf Watchdog-Respawn (Healthcheck-Probe-Fail). Funktional kein Problem — Worker pollt weiter. **Nicht MH-4b-3-blocking**; durabel BACKLOG für separates Worker-Healthcheck-Hardening.

2 — Konsumierte Artefakte

Komponente	Status	Konsumiert in MH-4b-3
MH-4a: `managed_proposals` table + `ManagedProposalState` enum + Model	`83dafca`	**READ-only** auf `managed_proposals` für state-validation
MH-4b-1: `createRequestCommand`	`3fd7846`	Service-Skelett unverändert; Validation-Helpers wiederverwendet
MH-4b-2: `createApproveCommand` + 5 Exceptions + Hard-Confirm-Builder + Parity-Pins	`be9cab7`	**massive Wiederverwendung**: 4 Exceptions reusable, validate-Helpers reusable, idempotency-Pattern symmetrisch
MH-1: `reject_managed_proposal` CommandType-Stub	`ea11637`	`$registry->get('reject_managed_proposal')` Existence-Check
Worker-Handler `_handle_reject_managed_proposal`	**MH-6**	nicht in MH-4b-3
Filament Wizard	**MH-5**	nicht in MH-4b-3
`ManagedStateService` (pause/resume/release)	**MH-4c**	nicht in MH-4b-3

Pattern-Source: `ProposalService::createApproveCommand` (MH-4b-2). MH-4b-3 ist nahezu spiegelsymmetrisch — gleicher Idempotency-Key, gleiche state-allowlist, kürzerer Validation-Stack.

3 — MH-4b-3 Scope-Definition

Welche States dürfen rejected werden?

Erlaubt:

- `risk_proposed` — gleicher single-State wie approve. Operator kann immer ablehnen, was Engine ihm vorgelegt hat.

Block-Verhalten (jeweils mit Exception):

State	Rea
`frozen`	Kein Prop exist zum Able
`proposal_pending`	Worl hat Engi noch nicht laufe lasse
`proposal_rejected`	bere abge
`proposal_aborted`	Engi selbs abor
`managed_active` / `managed_paused` / `managed_drift_alert` / `managed_released` / `exit_executed`	bere prom

Wichtig: Bei reject ist `proposal_json/sha256/file_path = NULL` theoretisch *kein* harter Reject-Blocker. ABER: Hard-Confirm braucht `proposal_sha256` für die sha8-Komponente — ohne sha8 ist kein Hard-Confirm möglich. → Reject ohne Worker-Cache ist konzeptionell auch geblockt mit `ProposalNotCachedException`. **Symmetrie zu approve** für Operator-Mental-Model.

Reject-Payload-Schema

```
{
  "environment":     "testnet",
  "proposal_id":     "<pid>",
  "asset":           "<asset>",
  "reason":          "<operator-text>",
  "hard_confirm":    "<asset>:reject:<sha8>",
  "proposal_sha256": "sha256:abc...",
  "decided_by":      <userId>,
  "expires_at":      "<iso8601-z>"
}
```

Felder die approve hat aber reject NICHT braucht:

- `variant` — Reject ist variant-agnostisch
- `overrides` — kein Override bei Reject
- `confidence_override_used` — kein Confidence-Check
- `proposal_file_path` — optional

Variante A — Minimal-Reject (7 Felder oben, ohne confidence_score / file_path)
Variante B — Symmetrisch-Reject (plus `confidence_score_at_decide` + `proposal_file_path` für identische Audit-Trail-Shape zu approve)

Empfehlung Variante B — Reject und Approve sollten identische Forensik-Trail-Struktur haben. Hilft Filament-Wizard + Worker-Handler MH-6 mit gleichem Payload-Schema arbeiten.

Hard-Confirm-Format

Operator-Frage: Braucht Reject einen Hard-Confirm?

Empfehlung JA, mit Format: `<asset>:reject:<sha8>`

Begründung:

1. Symmetrie zu approve `<asset>:<variant>:<sha8>` — derselbe Wizard-Pattern
2. Verhindert versehentliches Reject des falschen Proposals (z.B. nach Browser-Tab-Wechsel oder stale URL)
3. sha8 bindet an konkrete Proposal-Version — wenn Worker einen neuen Proposal generiert hat, ist alter Reject-Versuch ungültig
4. "reject" als String-Literal in der Mitte ist intentional menschenlesbar — Operator tippt bewusst "reject"

Build-Helper: `ProposalService::buildRejectHardConfirmString($asset, $proposalSha256): string`

Reason-Regeln

Pflicht: ja. Operator muss begründen, warum er ablehnt — Audit-Trail-Anforderung (G-DR-5 audit-heavy).

Validierung:

- Type: `string` (nicht null)
- Non-empty nach trim
- Max-Length: **256 chars** (per MH-4b Plan-Review §3.c)
- Whitespace-only → reject mit `InvalidCommandPayloadException`
- Keine Format-Beschränkung (free-form text)

Default: KEIN default. Reason muss explizit übergeben werden.

Idempotency-Verhalten

Existing Command	Verhalten
`reject_managed_proposal` (any status)	return existing, kein 2nd audit
`approve_managed_proposal` mit gleichem Key	`ProposalAlreadyDecidedException(detail='already_approved')`
Anderer command_type mit gleichem Key	`ProposalAlreadyDecidedException(detail='idempotency_key_conflict'` defensive STOP
Keine existing Command	INSERT new + audit

Symmetrisch zu MH-4b-2 approve-Verhalten, nur mit gespiegelten Rollen.

Service-API

```
ProposalService::createRejectCommand(
    string $proposalId,    // ^[a-zA-Z0-9_-]{1,128}$
    int    $userId,        // > 0
    string $reason,        // non-empty, trim, max 256 chars
    string $hardConfirm,   // exact "<asset>:reject:<sha8>"
): Command

ProposalService::buildRejectHardConfirmString(
    string $asset,
    string $proposalSha256,
): string  // returns "<asset>:reject:<sha8>"
```

4 — Konfliktcheck Sonder-Anforderungen

Anforderung	MH-4b-3-Touch	Verdict
Idempotency `mh:decide:<pid>` shared	Pattern symmetrisch zu approve	**konform**
Existing reject → return existing	kein 2nd audit	**konform**
Existing approve → ProposalAlreadyDecidedException	detail='already_approved'	**konform**
Other type → defensive STOP	symmetrisch zu approve	**konform**
audit_event prefix `managed.*`	`managed.asset_reject_requested`	**konform**
Transaction rollback	DB::transaction-Pattern wiederverwendet	**konform**
Hard-Confirm `<asset>:reject:<sha8>`	neuer Helper	**konform**
Mainnet/testnet enforcement	environment='testnet' hardcoded	**konform**

5 — Exception-Wiederverwendung

Exception	Reuse aus MH-4b-2?	Neue Variante?
`ProposalNotFoundException`	**Reuse**	nein
`ProposalAlreadyDecidedException`	**Reuse**	nein
`ProposalNotCachedException`	**Reuse**	nein
`HardConfirmMismatchException`	**Reuse**	nein
`ConfidenceOverrideRequiredException`	**nicht relevant für reject**	nein
`InvalidCommandPayloadException` (existing)	**Reuse für reason-Validierung**	nein

Ergebnis: 0 neue Exception-Klassen. Alle bestehenden 5 (4 davon relevant) sind reusable.

6 — Service-Implementation (Skizze)

Validation-Stack (in Reihenfolge)

```
1. validateProposalId(...)      → InvalidCommandPayloadException
2. validateUserId(...)          → InvalidCommandPayloadException
3. validateReason(...)          → InvalidCommandPayloadException  (NEU: trim +
length cap)
4. loadProposal(...)            → ProposalNotFoundException
5. assertRejectable(...)        → ProposalAlreadyDecidedException OR
ProposalNotCachedException
   (== assertApprovable Logik – gleiche state-allowlist + cache-fields)
6. expectedHardConfirm =
   buildRejectHardConfirmString($proposal->asset, $proposal->proposal_sha256)
7. hardConfirm exact match      → HardConfirmMismatchException
8. commandRegistry->get('reject_managed_proposal')
9. DB::transaction:
   a. Idempotency-Pre-Check + command_type discrimination
      → ProposalAlreadyDecidedException OR return existing
   b. Command::create(...)
   c. writeAudit(...) – event_type='managed.asset_reject_requested'
```

Hauptunterschied zu approve: keine Confidence-Validation, keine variant-Validierung, keine overrides-Validierung. Stattdessen reason-Validierung.

`assertRejectable` vs `assertApprovable`

Empfehlung: Refactor `assertApprovable` → `assertDecidable(ManagedProposal $p, string $cmdToken)` mit `$cmdToken` als Error-Message-Prefix. createApproveCommand und createRejectCommand rufen beide auf mit ihrem jeweiligen Token.

7 — Boundary-Contract

Tabu	Pin
`managed_proposals` INSERT/UPDATE	bereits MH-4b-1/2-pinned via Source-Grep
`managed_assets_history` Write	bereits pinned
File-IO	bereits pinned
Subprocess	bereits pinned
`ProposalWriter` / `ProposalReader` (Python)	architektonisch unmöglich
`command_worker.py` Touch	git-diff
Bot-Code Touch	git-diff
Filament-UI	MH-5
Mainnet	`environment='testnet'` hardcoded

8 — Test-Plan

Test-Klasse	Anzahl	Inhalt
ProposalServiceCreateRejectTest (happy)	5	INSERT commands + audit · payload env=testnet · alle required keys · audit prefix managed.asset_reject_requested · hardConfirm exact match
ProposalServiceCreateRejectStateTests	9	reject when state=frozen / proposal_pending (→ NotCached) / proposal_rejected / proposal_aborted / managed_active / managed_paused / managed_drift_alert / managed_released / exit_executed
ProposalServiceCreateRejectNotFoundTest	2	proposal_id not in DB · empty managed_proposals table
ProposalServiceCreateRejectCacheTests	4	proposal_json/sha256/file_path NULL (alle 3 + kombiniert)
ProposalServiceCreateRejectDecidedTest	1	decided_at IS NOT NULL → ProposalAlreadyDecidedException
ProposalServiceCreateRejectHardConfirmTests	6	exact match · asset mismatch · sha8 mismatch · case mismatch · empty string · missing 'reject' segment · missing sha8 segment
ProposalServiceCreateRejectReasonTests	6	non-empty required · whitespace-only rejected · max 256 chars · exactly 256 chars accepted · 257 chars rejected · free-form chars/unicode accepted
ProposalServiceCreateRejectIdempotencyTests	4	reject-twice returns existing (no 2nd audit) · reject after existing approve → ProposalAlreadyDecidedException · idempotency-key format pinned · other-command-type conflict
ProposalServiceCreateRejectAtomicityTest	1	failed audit insert rolls back command
RejectHardConfirmBuilderTest	3	builds `<asset>:reject:<sha8>` · case preserved · works with raw hex
ProposalServiceCreateRejectBoundaryTests	3	no managed_proposals UPDATE during reject · no managed_assets_history INSERT · source-grep symmetric to approve
ProposalServiceCreateRejectUserIdTests	2	zero rejected · negative rejected
Total	**~46 Tests**	(kleiner als approve weil keine variant/ overrides/confidence)

9 — Stop-Regeln MH-4b-3

ID	Stop wenn…
MH-4b-3-SR-1	Service UPDATE auf `managed_proposals`
MH-4b-3-SR-2	Service INSERT in `managed_assets_history`
MH-4b-3-SR-3	Service file IO oder subprocess
MH-4b-3-SR-4	Service akzeptiert `environment != 'testnet'`
MH-4b-3-SR-5	Service ruft Worker / Engine / Writer / Reader
MH-4b-3-SR-6	Service-Methode NICHT in `DB::transaction`
MH-4b-3-SR-7	reason NULL / empty / whitespace-only akzeptiert
MH-4b-3-SR-8	reason > 256 chars akzeptiert
MH-4b-3-SR-9	Hard-Confirm-Validierung ist case-insensitive
MH-4b-3-SR-10	Audit-Event-Prefix ist nicht `managed.*`
MH-4b-3-SR-11	Shared-Key approve-Command wird stumm zurückgegeben statt ProposalAlreadyDecidedException
MH-4b-3-SR-12	Reject akzeptiert variant / overrides / confidence_override Parameter (nicht zur API gehören)

10 — Backup / Migration / Restart-Bedarf

Aktion	Pflicht?
pg_dump GUI-DB	**NEIN — keine Migration**
State-Files snapshot	**NEIN**
Memory-Sicherung	**NEIN**
DB Migration	**NEIN**
Bot-Restart	**NEIN**
Worker-Restart	**NEIN**
docker cp	**NEIN**
GUI Cache-Clear	**NEIN (per Operator-Korrektur seit MH-4b-1)**

→ **MH-4b-3 = Reine Code+Test+Commit Phase.**

11 — Risiken-Übersicht

#	Risiko	Severity	Mitigation
R1	reject-without-Worker-cache desired aber blockiert	LOW	Cache-Pflicht ist symmetric mit approve; Operator-Workflow: wait+retry oder Worker `--once` manuell
R2	Hard-Confirm-Format-Verwechslung mit approve	LOW	Wizard MH-5 baut den expected-string explizit; Operator copy-pastet aus UI
R3	reason free-form erlaubt XSS-Vektoren in späterer Filament-Anzeige	LOW	Filament 3 escapes by default; AuditMetadataScrubber filtered sensitive tokens
R4	reason zu lang verschwendet DB-Storage	LOW	256 chars cap; pragmatischer Wert
R5	Reuse of `assertApprovable` ist semantisch reject-fremd	LOW	Refactor zu `assertDecidable($p, $cmdToken)` löst
R6	approve-after-reject Race im Wizard	MEDIUM	Idempotency-Pre-Check mit command_type-Discrimination; Test pinnt
R7	DB::transaction rollback bei audit failure	LOW	Pattern bewährt
R8	reason mit Newlines / Unicode bricht Audit-Metadata-Scrubber	LOW	AuditMetadataScrubber existing; Reuse + Test
R9	Reject-Hard-Confirm-Builder-Drift mit approve-Builder	LOW	beide getrennte static methods; Parity-Test pinnt buildRejectHardConfirmString

12 — Kleinste sichere Code-Phase + GO/NO-GO

Subschnitt-Optionen

Variante	Inhalt	LOC	Tests	Risk
MH-4b-3-monolithic ← empfohlen	createRejectCommand + buildRejectHardConfirmString + assertDecidable-Refactor + Tests	~600	~46	LOW
MH-4b-3-a	NUR createRejectCommand happy-path + 1 Test-Klasse	~200	~10	LOW
MH-4b-3-b (folgt)	State + Cache + Decided + Hard-Confirm Tests	~250	~22	LOW
MH-4b-3-c (folgt)	Reason + Idempotency + Atomicity + Boundary	~150	~14	LOW

Empfehlung: MH-4b-3-monolithic

Begründung:

1. **Massive Code-Wiederverwendung** aus MH-4b-2 — kein neuer Pattern, kein neuer Discovery-Aufwand
2. **0 neue Exception-Klassen** — reduziert Surface gegenüber MH-4b-2

3. **Symmetrie zu approve** — Test-Skelett ist mirror; reject ohne approve würde Asymmetrie hinterlassen
4. **assertDecidable Refactor** ist klein + risk-frei + reuse-fördernd
5. **Validation-Stack ist kürzer als approve** (kein variant/overrides/confidence) → kleinere Komplexität
6. **Rollback-Cost minimal** — `git revert` + 0 DB-Rows

Pre-Implementation-Q an Operator

1. **MH-4b-3-monolithic** oder Sub-Cut? ← **Empfehlung monolithisch**
2. **Hard-Confirm-Format** `<asset>:reject:<sha8>` bestätigen? ← **Empfehlung JA** (symmetrisch zu approve)
3. **Reason max-Length 256 chars** ok? ← **Empfehlung 256** (pragmatisch)
4. **Payload-Variante A (minimal)** oder **B (symmetric mit confidence_score + file_path)?** ← **Empfehlung B** (Forensik-Symmetrie)
5. **assertApprovable** umbenennen zu **assertDecidable** mit `$cmdToken`-Parameter? ← **Empfehlung JA** (saubere Reuse, kein Code-Duplikat)
6. **Reuse aller 4 relevanten Exceptions** ohne neue? ← **Empfehlung JA bestätigen** (0 neue Exception-Klassen in MH-4b-3)
7. **audit_event_type =** `managed.asset_reject_requested`? ← **Empfehlung JA**

Scope-Größe + Risiko-Bewertung MH-4b-3-monolithic

Aspekt	Bewertung
LOC	~600 (Service-Erweiterung + ~500 Tests)
Tests	~46
Komplexität	LOW — massive Reuse aus MH-4b-2
Blast-Radius	NULL — INSERT-only auf `commands` + `audit_events`
Roll-Back-Cost	minimal: `git revert` + 0 neue DB-Rows
Dependencies-Klärung	Q-2/Q-3/Q-4/Q-5/Q-7 Operator-Approval
Backup-Pflicht	NEIN
Migration-Pflicht	NEIN
Restart-Pflicht	NEIN
Mainnet-Touch	NEIN
docker cp	NEIN
Worker-PID-Wechsel-Auswirkung	KEINE (Service-Code-only)

STOP vor Implementierung. Erwarte Operator-GO mit Subschnitt-Wahl + Q1-Q7-Approval, insbesondere Hard-Confirm-Format-Pin + Payload-Variante A/B + assertDecidable-Refactor.