MH-4b-2 ProposalService::createApproveCommand — Plan-Review

Projekt: Steve-TradingBot · Phase: RECON-MH-4b-2 · Author: claude-opus-4-7[1m]
Generated: 2026-05-12 19:57 UTC · master HEAD: `3fd7846` (MH-4b-1 closed)
Status: **NO CODE** Plan-Review only — Operator-GO erforderlich vor MH-4b-2 Code-Phase

1 — Live-State Verification

Check	Soll	Ist	OK
master HEAD	`3fd7846`	`3fd7846 mh-4b-1: add proposal request command service`	✓
`git status`	clean	empty output	✓
Bot in-container PID	29984	`29984 python3 main.py --paper`	✓
Worker Host PID	338185	`python3 -m trading.command_worker` running	✓
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	✓

2 — Konsumierte Artefakte

Komponente	Status	Konsumiert in MH-4b-2
MH-4a: `managed_proposals` + `managed_assets_history` Tables + Enum + Models	`83dafca`	**READ-only** auf `managed_proposals` für State-Validation
MH-4b-1: `ProposalService::createRequestCommand`	`3fd7846`	Service-Skelett-Pattern + AuditMetadataScrubber-Helper bleiben
MH-1: `approve_managed_proposal` CommandType-Stub	`ea11637`	`$registry->get('approve_managed_proposal')` Existence-Check
MH-3a: Engine V1 + `CONFIDENCE_THRESHOLD=0.50`	`26c8ab3`	Service-Konstante `CONFIDENCE_THRESHOLD=0.50` (parity-mirror)
MH-3b: ProposalWriter immutable	`62a08f4`	Service nutzt `proposal_sha256` + `proposal_file_path` aus managed_proposals als Integrity-Snapshot im approve-Payload
MH-2: ProposalReader	`50cc5c2`	Worker-only — Service nicht angefasst
`ManagedProposalState::RiskProposed` (Enum)	`83dafca`	State-Validation nutzt das Enum
`ProposalService::createRejectCommand`	MH-4b-3	nicht in MH-4b-2
Worker-Handler	MH-6	nicht in MH-4b-2
Filament Wizard	MH-5	nicht in MH-4b-2

Pattern-Source: `gui/app/Services/Apply/Baseline/ApplyBaselineService::createApplyCommand` (Idempotency-Pre-Check, DB::transaction, writeAudit) + `gui/app/Services/Apply/Baseline/BaselineHoldingsAllowlist::buildHardConfirmString` (Hard-Confirm-Pattern `<count>:<sha8>`).

3 — MH-4b-2 Scope-Definition

Approve-Lebenszyklus-Stelle

```
Operator-Wizard (MH-5)
  ↓ submit final step
ProposalService::createApproveCommand()                ← MH-4b-2
  ↓
DB::transaction:
   - READ managed_proposals row (validation)
   - READ existing commands with idempotency-key (decide-pattern)
   - INSERT commands (status='pending', payload, key)
   - INSERT audit_events (managed.asset_promote_requested)
  ↓
==================== Service boundary ====================
  ↓
Worker --once / Daemon                                  ← MH-6
  ↓
_handle_approve_managed_proposal:
   - re-validate proposal_sha256 against on-disk file
   - Two-File-Atomic (managed_state.json + baseline_holdings.json)
   - UPDATE managed_proposals SET state='managed_active', decided_at=...
   - INSERT managed_assets_history
   - audit managed.asset_promoted
```

Welche States dürfen approved werden?

Erlaubt:

- `risk_proposed` — der einzige State, in dem `approve_managed_proposal` semantisch Sinn macht (Worker hat Engine + Writer ausgeführt, proposal_json + sha256 + file_path sind gesetzt)

Verboten (jeweils mit klarer Exception):

State	Rea
`frozen`	Asse nicht analy
`proposal_pending`	Worl Engi nicht lasse
`proposal_rejected`	bere abge
`proposal_aborted`	Engi abor (z.B. no_b
`managed_active` / `managed_paused` / `managed_drift_alert` / `managed_released` / `exit_executed`	bere prom past- Lebe

Plus die DB-Cache-Integrity-Checks:

Condition	Exception
`managed_proposals` row not found	`ProposalNotFoundException` (5. Exception)
`proposal_json IS NULL`	`ProposalNotCachedException`
`proposal_sha256 IS NULL`	`ProposalNotCachedException`
`proposal_file_path IS NULL`	`ProposalNotCachedException`
`decided_at IS NOT NULL`	`ProposalAlreadyDecidedException`

proposal_sha256 + proposal_file_path Behandlung

Service liest beide aus `managed_proposals` row und **echoed sie in den approve-Payload** als Integrity-Snapshot:

```
{
  "proposal_sha256":    "sha256:abc...",
  "proposal_file_path": "/home/node/.openclaw/.../risk_proposals/<id>.json"
}
```

Vorteile:

- Worker MH-6 kann beim Execute den on-disk file lesen + sha256 neu berechnen + gegen payload.proposal_sha256 vergleichen → fängt Tampering zwischen Engine-Run und Worker-Execute ab
- Audit-Trail enthält den exakten hash der approved Variante
- Hard-Confirm-Trail: was Operator approved hatte ist sha-hash-verifiziert nachvollziehbar

proposal_id Validierung

Strict-ASCII Regex `^[a-zA-Z0-9_-]{1,128}$` (1:1 mirror MH-3b `PROPOSAL_ID_REGEX`). Service-internes private const + Parity-Test (analog `ManagedProposalStateEnumTest`).

Hard-Confirm-Format

Operator-pinned in MH-4b Plan-Review:

- Format: `<asset>:<variant>`
- z.B. `"ETH:recommended"` oder `"BTC:conservative"`
- **case-sensitive** (HardConfirmMismatchException bei case-mismatch)
- Build-Helper: `ProposalService::buildHardConfirmString($asset, $variant)`
- Filament-Wizard zeigt erwarteten String und Operator muss exakt eintippen

Confidence-Override-Logik

```
score = managed_proposals.proposal_json.confidence.overall_score
threshold = 0.50  // CONFIDENCE_THRESHOLD, parity mit MH-3a engine

if score < threshold AND confidenceOverride !== true:
    throw ConfidenceOverrideRequiredException(score, threshold)

if score < threshold AND confidenceOverride === true:
    payload['confidence_override_used'] = true
    audit.metadata['confidence_override_used'] = true
    // weiter mit approve

if score >= threshold:
    payload['confidence_override_used'] = false
    // confidenceOverride wird ignoriert (kein Effekt)
```

Idempotency-Pattern (Shared-Key `mh:decide:<pid>`)

Szenario	Verhalten
1st approve, no existing command	INSERT new approve command + audit; return new
2nd approve (double-click), existing approve	return existing (idempotency hit, NO 2nd audit)
approve, existing reject command for same pid	throw `ProposalAlreadyDecidedException(detail='rejected')`
approve, existing approve in `succeeded` state	return existing (idempotent)

Wichtig: Service muss `existing->command_type` prüfen, NICHT nur Existence. Bei Mismatch → explicit exception (kein silent return des falschen Typs).

Was wird gelesen vs. geschrieben

READ (SELECT):

- `managed_proposals` row by proposal_id
- `commands` row by idempotency_key (existing-decide check)

WRITE (INSERT):

- `commands` (new approve command, status=pending)
- `audit_events` (managed.asset_promote_requested)

NEVER WRITE/UPDATE:

- `managed_proposals` (Worker MH-6)
- `managed_assets_history` (Worker MH-6)
- jede File (no FS write)

4 — Strict Boundary

Tabu	Pin
`managed_proposals` INSERT/UPDATE	Source-Grep: `ManagedProposal::cr` `->save()` / `->update()` verboten; `ManagedProposal::find($pid)` u `>where()` erlaubt
`managed_assets_history` Write	Source-Grep: `ManagedAssetHistory` verboten
`managed_state.json` / `baseline_holdings.json` / `risk_proposals/*.json` Write	File-IO-Call-Tokens (`file_put_conte` `fwrite(` , etc.) Pin
`ProposalWriter`-Aufruf	architektonisch unmöglich (PHP ↔ Pytho Source-Grep
`command_worker.py` Touch	git-diff nur `gui/` files
Bot-Code Touch	git-diff nur `gui/` files
Filament-UI	MH-5 — nur Service + Tests + Exceptio
Mainnet	Service-Konstante `APPROVE_ENVIRON` `= 'testnet'` hardcoded

5 — Sonder-Anforderungen geprüft

Frage	Antwort
Idempotency `mh:decide:<pid>` shared mit reject	Pattern. Service::createApproveCommand prüft Existence + Command-Type. Bei reject-mismatch: `ProposalAlreadyDecidedException`.
Existierender reject-command	→ `ProposalAlreadyDecidedException(detail='rejected')`
Existierender approve-command	→ return existing (idempotency hit; NO 2nd audit)
audit_event prefix `managed.*`	Service nutzt `event_type='managed.asset_promote_requested'`; Test: assertStartsWith('managed.', ...)
Transaction rollback	DB::transaction-Wrap; Test: simulate AuditEvent::create failure → command rollback verified
Hard-Confirm gegen falsche Proposal-ID	Operator gibt `<asset>:<variant>` ein, NICHT proposal_id. Service vergleicht gegen `managed_proposals.asset + ':' + $variant`. Falsche asset/variant-Kombi → HardConfirmMismatchException.
confidence.overall_score und override_required	siehe §3 Confidence-Override-Logik; 3 Pfade pinned
Mainnet/testnet enforcement	(a) Service hardcodes `environment='testnet'`; (b) CommandTypeRegistry `allowedEnvironments=['testnet']`; (c) Worker-Handler MH-6 prüft zusätzlich; (d) Engine MH-3a `MainnetBlockedError`; (e) Writer MH-3b `MainnetBlockedError`. → 5-Layer-Block intakt

6 — Empfohlene Lieferung

Service-API

```
ProposalService::createApproveCommand(
    string  $proposalId,         // ^[a-zA-Z0-9_-]{1,128}$
    int     $userId,             // > 0
    string  $variant,            // ∈ {recommended, conservative, aggressive}
    array   $overrides,          // optional, whitelist [stop_loss,
take_profit, max_allocation_usdt, trailing_stop_enabled]
    string  $hardConfirm,        // exact match "<asset>:<variant>"
    bool    $confidenceOverride = false,
): Command

ProposalService::buildHardConfirmString(string $asset, string $variant): string
    // returns "<asset>:<variant>"
```

Erlaubte Service-Methoden

- `createRequestCommand(...)` (MH-4b-1, existing)
- `createApproveCommand(...)` (MH-4b-2, **neu**)
- private `validateProposalId(string)`
- private `validateVariant(string)`
- private `validateOverrides(array)` (whitelist)
- private `validateHardConfirm(string $hc, string $asset, string $variant)`
- private `validateConfidence(array $proposalJson, bool $override): bool`
- private `loadProposal(string $pid): ManagedProposal` (returns row or throws)
- private `assertApprovable(ManagedProposal $p)` (state + decided_at + json/sha/path null-checks)
- private `assertIdempotencyAvailable(string $key)` (decide-pattern check)
- static `buildHardConfirmString(string $asset, string $variant): string`
- private `writeAudit(...)` (existing helper)

Verbotene Service-Methoden

- `Command::update()` — Worker setzt status; Service nur INSERT-pending
- `ManagedProposal::create()` / `->save()` / `->update()` — Worker MH-6 only
- `ManagedAssetHistory::*` — Worker MH-6 only
- File-IO (`file_get_contents(` , `file_put_contents(` , `fopen(` , `fwrite(` , `fputs(`)
- Subprocess (`exec(` , `shell_exec(` , `proc_open(` , `passthru(` , `system(` , `popen(`)
- `environment != 'testnet'` Akzeptanz

Exception-Liste

Operator hatte **4 Exceptions** spezifiziert. Empfehlung: **5 Exceptions** für maximale Caller-Klarheit:

#	Exception	When	Recovery
1	`ProposalNotFoundException`	row mit proposalId existiert nicht in managed_proposals	Operator-Action: zurück zum Wizard, neuer Request
2	`ProposalAlreadyDecidedException`	state ∉ risk_proposed OR decided_at IS NOT NULL OR existing reject command	Operator-Action: GUI zeigt aktuellen State; "already decided as X"
3	`ProposalNotCachedException`	state=proposal_pending OR proposal_json/ sha256/file_path IS NULL (Worker hat noch nicht gelaufen)	Operator-Action: warten + retry; alternativ Worker manuell `--once`
4	`HardConfirmMismatchException`	hardConfirm ≠ `<asset>:<variant>`	Operator-Action: re-type confirm-string
5	`ConfidenceOverrideRequiredException`	score < 0.50 und confidenceOverride=false	Operator-Action: confidenceOverride=true setzen ODER reject

Alternative (4 Exceptions, operator-original): ProposalNotFound wird in ProposalAlreadyDecidedException gefoldet mit `detail='not_found'`. → Operator entscheidet.

Payload-Schema (commands.payload_json)

```
{
  "environment":                "testnet",
  "proposal_id":                "<pid>",
  "asset":                      "<asset>",
  "variant":                    "recommended|conservative|aggressive",
  "overrides":                  { /* optional whitelisted keys */ },
  "hard_confirm":               "<asset>:<variant>",
  "confidence_override_used":   false,
  "confidence_score_at_decide": 0.65,
  "proposal_sha256":            "sha256:abc...",
  "proposal_file_path":         "/home/node/.../<pid>.json",
  "decided_by":                 <userId>,
  "expires_at":                 "<iso8601-z>"
}
```

`overrides` Whitelist:

- `stop_loss` (number)
- `take_profit` (number)
- `max_allocation_usdt` (number, positive)
- `trailing_stop_enabled` (bool)
- **NICHT erlaubt:** `strategy_group`, `variant`, `confidence_threshold`, andere keys → InvalidCommandPayloadException

7 — Test-Plan

Test-Klasse	Anzahl	Inhalt
ProposalServiceCreateApproveTest (happy path)	6	INSERT commands + audit · payload env=testnet · payload alle required keys · audit prefix managed.* · hardConfirm exact-match · confidence high-score
ProposalServiceCreateApproveStateTests	8	reject when state=frozen · proposal_pending · proposal_rejected · proposal_aborted · managed_active · managed_paused · managed_drift_alert · managed_released · exit_executed
ProposalServiceCreateApproveNotFoundTest	2	proposal_id not in DB → ProposalNotFoundException · empty managed_proposals table
ProposalServiceCreateApproveCacheTests	4	proposal_json IS NULL · proposal_sha256 IS NULL · proposal_file_path IS NULL · all 3 NULL
ProposalServiceCreateApproveDecidedTests	2	decided_at IS NOT NULL · existing reject command (shared-key)
ProposalServiceCreateApproveHardConfirmTests	6	exact match · asset mismatch · variant mismatch · case mismatch (`eth:recommended`) · empty string · invalid format (no colon)
ProposalServiceCreateApproveConfidenceTests	5	high-score happy (override ignored) · low-score-no-override → exception · low-score-with-override → happy + audit-flag · score=0.50 (boundary) · missing confidence.overall_score → exception
ProposalServiceCreateApproveVariantTests	4	recommended · conservative · aggressive · invalid variant string
ProposalServiceCreateApproveOverridesTests	7	empty overrides · only stop_loss · only take_profit · all 4 whitelist keys · reject strategy_group · reject variant · reject custom key
ProposalServiceCreateApproveIdempotencyTests	4	approve-twice returns existing (no 2nd audit) · approve after reject → ProposalAlreadyDecidedException · idempotency-key format pinned · DB-Unique-Constraint
ProposalServiceCreateApproveAtomicityTest	2	failed audit insert rolls back command · no orphan
ProposalIdRegexParityTest	2	PHP regex matches MH-3b char-for-char · accept/reject cases parity
ConfidenceThresholdParityTest	1	Service::CONFIDENCE_THRESHOLD === 0.50 (mirror MH-3a engine)

BoundaryTests	3	no managed_proposals UPDATE · no managed_assets_history INSERT · no file-IO/ subprocess tokens
HardConfirmBuilderTest	3	builds `<asset>:<variant>` · case preserved · usable in Filament
Total	**~59 Tests**	(über Roadmap-Schätzung ~30; viele State-Combos)

8 — Stop-Regeln MH-4b-2

ID	Stop wenn…
MH-4b-2-SR-1	Service UPDATE auf `managed_proposals` (auch nicht `Model::update()`)
MH-4b-2-SR-2	Service INSERT in `managed_assets_history`
MH-4b-2-SR-3	Service file IO oder subprocess
MH-4b-2-SR-4	Service akzeptiert `environment != 'testnet'`
MH-4b-2-SR-5	Service ruft Worker oder spawnt Subprocess
MH-4b-2-SR-6	Service-Methode NICHT in `DB::transaction`
MH-4b-2-SR-7	`proposal_id` Regex weicht von MH-3b ab
MH-4b-2-SR-8	`hardConfirm` Validierung ist case-insensitive
MH-4b-2-SR-9	`CONFIDENCE_THRESHOLD` weicht von MH-3a 0.50 ab (Parity-Test bricht)
MH-4b-2-SR-10	`confidence_override_used` Flag fehlt in audit-metadata bei override
MH-4b-2-SR-11	Audit-Event-Prefix ist nicht `managed.*`
MH-4b-2-SR-12	`overrides` Whitelist akzeptiert `strategy_group` / `variant` / `confidence_threshold`
MH-4b-2-SR-13	Shared-Key reject-Command wird stumm zurückgegeben statt ProposalAlreadyDecidedException
MH-4b-2-SR-14	`proposal_sha256` / `proposal_file_path` fehlen im commands.payload (Integrity-Trail)

9 — Backup / Migration / Restart-Bedarf

Aktion	Pflicht?	Begründung
pg_dump GUI-DB	**NEIN**	keine Migration; reine Service-Code-Phase
`live_portfolio.json` snapshot	**NEIN**	kein State-Touch
`state/` snapshot	**NEIN**	kein State-Touch
Memory-Sicherung	**NEIN**	Closure-Pin reicht
DB Migration	**NEIN**	MH-4a Schema bleibt unverändert
Bot-Restart	**NEIN**	kein Bot-Code-Touch
Worker-Restart	**NEIN**	command_worker.py unverändert
docker cp	**NEIN**	PHP via Laravel Live-Volume mount
GUI Cache-Clear	**NEIN (per Operator-MH-4b-1-Korrektur)**	keine Config/Routes/Views-Änderung

→ **MH-4b-2 = Reine Code+Test+Commit Phase**.

10 — Risiken-Übersicht

#	Risiko	Severity	Mitigation
R1	Service UPDATE auf managed_proposals versehentlich	LOW	Source-Grep `ManagedProposal::create|update|save|delete` Test
R2	proposal_id Regex Drift PHP↔MH-3b	LOW	Parity-Test gegen hardcoded MH-3b-mirror
R3	CONFIDENCE_THRESHOLD Drift PHP↔MH-3a	LOW	Parity-Test prüft `=== 0.50`
R4	Hard-Confirm-Bypass via case-mismatch	LOW	case-sensitive `===` + 6 Test-Cases
R5	Confidence-Score-Read aus stale DB-Cache (Worker hat ausgelaufenen file aber DB nicht aktualisiert)	MEDIUM	Service liest aus `managed_proposals.proposal_json` (DB), nicht von Disk. Worker verifies sha256 zur Execute-Zeit. Akzeptable Read-Cache-Staleness im Service-Scope.
R6	Approve-Reject-Race ohne Type-Check	MEDIUM	Idempotency-Pre-Check prüft `existing->command_type` und throws `ProposalAlreadyDecidedException` bei mismatch
R7	confidence_override silently true ohne `score < threshold`	LOW	Audit-metadata `confidence_override_used` ist nur `true` wenn beides zutrifft. Test pinned.
R8	overrides whitelist drift (neue keys ohne G-DR-3 Q-MH-3 Review)	LOW	Whitelist-Konstante + Test pinnt exakt die 4 erlaubten keys
R9	proposal_sha256/file_path stale	LOW (MH-6 prüft)	Service echoed das DB-Cache-Value; Worker-Handler MH-6 verifies sha256 gegen on-disk Datei zur Execute-Zeit (defense-in-depth)
R10	DB::transaction-Rollback bei AuditEvent failure	LOW	Pattern bewährt aus ApplyBaselineService + MH-4b-1; Test simuliert
R11	proposal_json deserialization JSON-Cast-Drift	LOW	Eloquent `$casts = ['proposal_json' => 'array']` (MH-4a). Test pinned.
R12	`confidence.overall_score` missing in proposal_json (Worker-Bug)	LOW	Service validiert path-existence; throws `InvalidCommandPayloadException` (subtype)

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

Subschnitt-Optionen

Variante	Inhalt	LOC	Tests	Risk
MH-4b-2-monolithic	createApproveCommand + 5 Exceptions + 15 Test-Klassen + buildHardConfirmString	~1900	~59	LOW-MEDIUM
MH-4b-2-a	NUR Skelett: createApproveCommand mit happy-path + 1 Exception + 1 Test-Klasse	~400	~12	LOW
MH-4b-2-b (folgt)	State/Cache/Decided Exceptions (3 Exceptions) + 3 Test-Klassen	~500	~16	LOW
MH-4b-2-c (folgt)	Hard-Confirm + Confidence + Overrides + Idempotency + Atomicity + Boundary + Parity	~1000	~31	MEDIUM

Empfehlung: MH-4b-2-monolithic

Begründung dieses Mal MONOLITHISCH (anders als MH-4b-1):

1. **Approve ist atomar in der Logik** — alle 5 Exceptions + Hard-Confirm + Confidence + Overrides bauen auf demselben Eingangs-Validation-Stack auf. Sub-Cuts würden den Validations-Stack unvollständig deployen → erhöhtes Bug-Risiko bei intermediate-state.
2. **Pattern bereits etabliert** — `ApplyBaselineService::createApplyCommand` ist die Vorlage; alle Sub-Mechaniken (DB::transaction, audit, idempotency) sind bewährt. Kein Pattern-Discovery in MH-4b-2.
3. **Test-Vollständigkeit** — Approve hat den größten Validation-Stack im gesamten MH-Service-Layer. Sub-Cuts würden Coverage-Lücken hinterlassen (z.B. Hard-Confirm würde in `b` fehlen wenn `a` deployed ist).
4. **R5 / R6 / R10 nur durch volles Set abdeckbar** — Race-Schutz, Stale-Cache und Atomicity sind kein add-on, sondern integral.
5. **Rollback-Cost minimal** — alles in einem Commit `git revert`-bar; DB hat 0 Rows im managed_proposals → 0 Side-Effect-Risiko.

Falls Operator dennoch Sub-Cut bevorzugt → **MH-4b-2-a** (Skelett + happy-path + ProposalNotFoundException) als Erstcommit ist die sicherste Variante (etablisiert das `loadProposal`-Helper-Pattern).

Pre-Implementation-Q an Operator

1. **MH-4b-2-monolithic oder MH-4b-2-a (Skelett)?** ← **Empfehlung monolithisch**
2. **5 oder 4 Exceptions?** ProposalNotFound als 5. eigene Exception, oder in ProposalAlreadyDecidedException gefoldet? ← **Empfehlung 5 separate Exceptions** (Caller-Klarheit)
3. **Hard-Confirm exakt `<asset>:<variant>`** oder `<asset>:<variant>:<sha8>`? ← **Empfehlung `<asset>:<variant>`** (Operator-Original; sha256 ist im Payload separat)

4. **Overrides-Whitelist**: 4 keys (stop_loss / take_profit / max_allocation_usdt / trailing_stop_enabled) — alle akzeptieren? ← **Empfehlung JA** (Q-MH-3 Operator-Override-Scope)
5. **CONFIDENCE_THRESHOLD = 0.50** als Service-Konstante mit Parity-Test gegen MH-3a — bestätigen? ← **Empfehlung JA** (Defense-in-Depth durch Parity-Pin)
6. **Audit-event_type**: `managed.asset_promote_requested` (Service-Layer-Event distinct von Worker-Success-Event `managed.asset_promoted`)? ← **Empfehlung JA**

Scope-Größe + Risiko-Bewertung MH-4b-2-monolithic

Aspekt	Bewertung
LOC	~700 Service + ~150 Exceptions + ~1100 Tests = ~1900
Tests	~59
Komplexität	Pattern-Mirror ApplyBaselineService; größter Validation-Stack im Service-Layer
Blast-Radius	NULL — INSERT-only auf `commands` + `audit_events`
Roll-Back-Cost	minimal: `git revert` + DB hat 0 neue Rows
Dependencies-Klärung	Q-MH-3 Override-Whitelist, CONFIDENCE_THRESHOLD Pin
Backup-Pflicht	NEIN
Migration-Pflicht	NEIN
Restart-Pflicht	NEIN
Mainnet-Touch	NEIN
docker cp	NEIN

STOP vor Implementierung. Erwarte Operator-GO mit Subschnitt-Wahl (monolithisch empfohlen) + Q1-Q6-Approval.