MH-4b ProposalService — Plan-Review

Projekt: Steve-TradingBot · Phase: RECON-MH-4b · Author: claude-opus-4-7[1m]
Generated: 2026-05-12 19:14 UTC · master HEAD: `83dafca` (MH-4a closed)
Status: **NO CODE** Plan-Review only — Operator-GO erforderlich vor MH-4b Code-Phase

1 — Live-State Verification

Check	Soll	Ist	OK
master HEAD	`83dafca`	`83dafca mh-4a: add managed_proposals + managed_assets_history tables`	✓
`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	✓
Worker-Daemon	alive	`clawbot-worker Up 18h` (Healthcheck-Drift pre-existing)	⚠
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	✓
`managed_proposals` table	live, 0 rows	**0 rows** ✓	✓
`managed_assets_history` table	live, 0 rows	**0 rows** ✓	✓
Tracebacks last 200 lines	0	0 matches	✓

2 — Konsumierte Artefakte (MH-1 → MH-4a)

Komponente	Status	Verfügbar für MH-4b
MH-1: 8 `managed.*` CommandTypes + Validators in CommandTypeRegistry	`ea11637`	`register_managed_proposal` / `approve_managed_proposal` / `reject_managed_proposal` validators ready — Service nutzt `$registry->get('...')`
MH-2: `ManagedStateReader` + `ProposalReader` dormant	`50cc5c2`	Worker-only — Service touched NICHT
MH-3a: `RiskProposalEngine` dry-run	`26c8ab3`	Worker-only — Service touched NICHT
MH-3b: `ProposalWriter` immutable	`62a08f4`	Worker-only — Service touched NICHT
MH-4a: `managed_proposals` + `managed_assets_history` + `ManagedProposalState` Enum + Models	`83dafca`	**direkt konsumiert** — Service READ-only auf `managed_proposals` für approve/reject Validation
`ManagedStateService` (pause/resume/release)	MH-4c	nicht in MH-4b Scope
Worker-Handler `_handle_*`	MH-6	nicht in MH-4b
Filament Wizard	MH-5	nicht in MH-4b

Pattern-Source: `gui/app/Services/Apply/Baseline/ApplyBaselineService.php` (RECON-2.3, 600 LOC). Übernehmen:

- `DB::transaction` Atomicity
- Idempotency-Pre-Check via `Command::where('idempotency_key', $key)->first()`
- `$this->commandRegistry->get($type)` für Payload-Validator-Aufruf
- `writeAudit()` private Helper mit AuditMetadataScrubber
- Hard-restrict `environment = 'testnet'` in Payload-Core
- Pre-Audit-Row bei Validation-Failure (`baseline.apply.precondition_rejected` Pattern)

3 — MH-4b Scope-Rekonstruktion

Service-Verantwortlichkeiten

`ProposalService` ist **command-builder + DB-cache-validator only**. Es:

- INSERTs in `commands` + `audit_events`
- READs `managed_proposals` für approve/reject Validation
- NIEMALS UPDATE / INSERT auf `managed_proposals` oder `managed_assets_history`
- NIEMALS Datei-Touch (keine risk_proposals, managed_state, baseline)
- NIEMALS Engine/Writer-Call
- NIEMALS Worker-Spawn

Pipeline (MH-4b ↔ MH-6 Trennung)

```
Operator-Klick (MH-5 Wizard, future)
   ↓
ProposalService::createXxxCommand()                ← MH-4b
   ↓
DB::transaction:
   - INSERT commands (status='pending', idempotency_key='mh:...')
   - INSERT audit_events (managed.asset_*_requested)
   - return Command instance
   ↓
==================== Service boundary ====================
   ↓
Worker --once / Daemon                               ← MH-6
   ↓
_handle_request_managed_proposal:
   - call Engine.generate() → ProposalResult
   - call Writer.apply()    → ApplyResult
   - INSERT managed_proposals (proposal_id, state='risk_proposed',
       proposal_json=..., proposal_file_path=..., proposal_sha256=...,
       proposal_cached_at=now(), generated_at=..., expires_at=...)
   - INSERT managed_assets_history
   - emit audit_event managed.asset_proposal_generated
```

3 Service-Methoden im Detail

a) `createRequestCommand(asset, userId, intent='operator_request')`

- **Input:** `asset` (Symbol, e.g. "ETH"), `userId` (operator), `intent` (string-tag)
- **Validation:**
  - `asset` matches `/^[A-Z0-9]{1,32}$/`
  - `userId` is positive int
  - `intent` is non-empty string ≤ 64 chars
- **Idempotency:** `mh:propose:{asset}:{YmdHis}` (per-second distinct)
- **DB-Writes:**
  - INSERT `commands` (command_type=`request_managed_proposal`, status=pending, payload, idempotency_key, requested_by=userId, timeout_at)
  - INSERT `audit_events` (event_type=`managed.asset_proposal_requested`, summary, user_id, metadata)
- **NIEMALS:** INSERT `managed_proposals` (kein proposal_id bekannt vor Engine-Run). DB-Cache-Row wird in MH-6 Worker-Handler erzeugt.

b) `createApproveCommand(proposalId, userId, variant, overrides=[], hardConfirm, confidenceOverride=false)`

- **Input:** `proposalId` (string, MH-3b regex), `userId`, `variant` (recommended|conservative|aggressive), `overrides` (SL/TP override-array), `hardConfirm` (string `<asset>:<variant>`), `confidenceOverride` (bool)
- **Validation:**
  - `proposalId` matches `/^[a-zA-Z0-9_-]{1,128}$/` (MH-3b strict ASCII)

  - `variant` ∈ {recommended, conservative, aggressive}
  - **READ `managed_proposals`:** row exists, `state === 'risk_proposed'`, `decided_at IS NULL`, `proposal_json IS NOT NULL` (Worker hat schon gelaufen)
  - `hardConfirm === "{$proposal->asset}:{$variant}"` (case-sensitive exact match) → sonst `HardConfirmMismatchException`
  - Read `proposal_json.confidence.overall_score`; wenn `< 0.50` und `confidenceOverride !== true` → `ConfidenceOverrideRequiredException`
  - `overrides` (optional): nur Keys `stop_loss`, `take_profit`, `max_allocation_usdt`, `trailing_stop_enabled` erlaubt (G-DR-3 Q-MH-3 — keine strategy_group / variant override)
- **Idempotency:** `mh:decide:{proposalId}` (shared mit reject — Race-Schutz)
- **DB-Writes:**
  - INSERT `commands` (command_type=`approve_managed_proposal`, status=pending, payload, idempotency_key)
  - INSERT `audit_events` (event_type=`managed.asset_promote_requested`)
- **NIEMALS:** UPDATE managed_proposals (Worker macht das in MH-6 nach Promote-Handler)

c) `createRejectCommand(proposalId, userId, reason)`

- **Input:** `proposalId`, `userId`, `reason` (non-empty string ≤ 256 chars)
- **Validation:**
  - `proposalId` matches MH-3b regex
  - READ `managed_proposals`: row exists, `state === 'risk_proposed'`, `decided_at IS NULL`
  - `reason` non-empty
- **Idempotency:** `mh:decide:{proposalId}` (shared mit approve)
- **DB-Writes:**
  - INSERT `commands` (command_type=`reject_managed_proposal`)
  - INSERT `audit_events` (event_type=`managed.asset_reject_requested`)

Operator-Frage-Klärungen

Frage	Antwort
Wie wird `proposal_json` aus Datei in DB-Cache gespiegelt?	**NICHT in MH-4b.** Worker (MH-6) liest on-disk `risk_proposals/<id>.json` und UPDATE managed_proposals SET proposal_json=... NACH Engine.generate() läuft. MH-4b Service touched proposal_json niemals (auch nicht READ — außer für approve/reject validation, dann reads from DB-cache nicht Datei).
Wie werden `proposal_file_path` / `proposal_sha256` / `proposal_cached_at` gesetzt?	**NICHT in MH-4b.** Worker (MH-6) füllt diese Spalten nach `ProposalWriter.apply()` aus `ApplyResult.target_path` / `.sha256` / `.written_at`.
Wie wird proposal_id Regex validiert?	Service hat private static helper `validateProposalId(string $pid)` mit regex `/^[a-zA-Z0-9_-]{1,128}$/` (1:1 mirror MH-3b `PROPOSAL_ID_REGEX`). Aufgerufen in `createApproveCommand` und `createRejectCommand`. **Nicht** in `createRequestCommand` (proposal_id existiert dort noch nicht).
Welche State-Transitions sind erlaubt?	Service erzeugt nur **commands**, keine State-Transitions. Worker (MH-6) führt Transitions aus mit `ManagedStateGuard`. Service prüft READ-only: Approve/Reject erfordern `state==='risk_proposed'` (Caller-Side Validation — Worker macht zusätzlich Guard-Check).
Wie werden approve/reject vorbereitet ohne Worker auszuführen?	Service inserts `commands.status='pending'`. Worker-Daemon pollt `commands` und führt Handler aus, wenn Worker `--once` läuft oder Daemon pollt. Service kehrt nach DB::transaction zurück — Operator sieht `Command` instance + idempotency_key + UUID.
Wo endet MH-4b?	Service writes commands+audit_events only. Service liest managed_proposals READ-only. Service ruft NIEMALS Worker.
Wo beginnt MH-4c?	`ManagedStateService` (pause/resume/release). Selbe Pattern, andere CommandTypes.
Wo beginnt MH-5?	Filament-Page + Wizard + Filament-Actions die ProposalService-Methoden aufrufen.
Wo beginnt MH-6?	Worker-Handler `_handle_request_managed_proposal` etc. in `command_worker.py`. Liest Engine, ruft Writer, UPDATEs managed_proposals, INSERTs managed_assets_history.

4 — Konfliktcheck Sonder-Anforderungen

Anforderung	MH-4b-Touch	Verdict
JSON Bot-SoT bleibt führend	Service touched **keine JSON-Datei**. Liest nur DB-Cache (`managed_proposals` row).	**Hybrid C eingehalten**
DB bleibt nur GUI-Cache	Service liest `managed_proposals` für Validation. Writes ausschließlich `commands` + `audit_events`.	**konform**
Proposal-Dateien bleiben immutable	Service touched NICHT `risk_proposals/<id>.json`. ProposalWriter wird NICHT aus PHP aufgerufen.	**MH-3b Immutability intact**
Kein ProposalWriter-Aufruf aus PHP	Service-Code importiert NICHT ProposalWriter. AST-pin: kein `use App\Services\.*Writer|Engine`. (PHP hat das nicht — Bot ist Python.)	**Architekturell unmöglich + Test-Pin**
Kein managed_state.json write	Service touched keine state files.	**konform**
Kein baseline_holdings.json write	Service touched keine state files.	**konform**
Kein command_worker Touch	`command_worker.py` bleibt unverändert.	**konform**
Kein Bot-Wiring	main.py / live_trade.py / paper_trade.py / risk_manager.py bleiben unverändert.	**konform**
Kein Filament Wizard	MH-5. Service ist Builder, nicht UI.	**konform**
Kein Mainnet	Service hard-restricts `environment='testnet'` in Payload-Core. CommandTypeRegistry-Validator prüft das zusätzlich.	**Defense-in-Depth**

5 — Identifizierte Risiken

#	Risiko	Severity	Mitigation
R1	**Approve auf Proposal mit `proposal_json IS NULL`** — Worker hat Initial-INSERT noch nicht durchgeführt	MEDIUM	Service::createApproveCommand prüft `proposal_json IS NOT NULL` vor INSERT. Throws `ProposalNotCachedException` sonst. Test simuliert die Race.
R2	**proposal_id-Regex-Drift PHP↔Bot** — MH-3b Bot-Regex und PHP-Service-Regex könnten divergieren	LOW	Private const `PROPOSAL_ID_REGEX = '/^[a-zA-Z0-9_-]{1,128}$/'` im Service + Test-Pin (`test_proposal_id_regex_parity_with_bot`) der gegen hardcoded mirror der MH-3b regex prüft.
R3	**Confidence-Override-Bypass** — Operator setzt confidenceOverride=true ohne reale Notwendigkeit	LOW	Service prüft `score < 0.50` UND benötigt explicit `confidenceOverride=true` . Audit-Metadata kennzeichnet `confidence_override_used=true` für Audit-Trail. Test: 3 Pfade (high-score happy, low-score-no-override exception, low-score-with-override happy + audit-flag).
R4	**Hard-Confirm-Bypass** — User klickt approve mit falschem Confirm-String	LOW	Service exact-match-validates `hardConfirm === "{$asset}:{$variant}"` (case-sensitive). `HardConfirmMismatchException` . Test: 5 Pfade (mismatch case, mismatch asset, mismatch variant, lowercase mismatch, happy).
R5	**Approve+Reject Race (gleiche proposal_id)**	LOW	Shared idempotency-key `mh:decide:<pid>` . DB-Unique-Constraint auf `commands.idempotency_key` (existing). Erste schreibt → ok; zweite findet existing → returnt existing Command (kein 2nd INSERT). Test: simuliert Race.
R6	**Service insertet managed_proposals row prematurely** (proposal_id noch nicht bekannt im createRequestCommand)	MEDIUM	**Architektur-Entscheidung MH-4b:** Service INSERTs `managed_proposals` **NICHT**. Worker (MH-6) macht das nach Engine-Run wenn proposal_id existiert. Hierdurch entfällt die "tentative proposal_id"-Problematik. Plan-Review-Doc pinnt das.
R7	**Worker hat Initial-INSERT noch nicht — GUI zeigt nichts**	LOW	UX-Issue, kein Service-Bug. Filament (MH-5) zeigt "Proposal angefragt, Worker pending" status basierend auf `commands.status='pending'` ohne `managed_proposals` row. Erst nach Worker-Run erscheint Row.
R8	**AuditMetadataScrubber-Bypass**	LOW	Service nutzt existing scrubber, Pattern aus ApplyBaselineService. Test pinnt.
R9	**DB::transaction-Rollback bei Audit-Insert-Failure**	LOW	DB::transaction wirft automatic rollback bei Exception. Test: simuliert AuditEvent::create() failure → kein Command row übrig.
R10	**Idempotency-Key-Collision verschiedener Operators**	LOW	createRequestCommand idempotency `mh:propose:{asset}:{YmdHis}` ist Operator-agnostic. Wenn zwei Operators denselben Asset in derselben Sekunde

			anfragen → second returns first's command (audit-trail erhalten). Akzeptable Semantik.

6 — Test-Plan

Test-Klasse	Anzahl	Inhalt
ProposalServiceCreateRequestTest	7	happy path (commands + audit inserted) · asset regex validation · userId positive int · intent length cap · idempotency same-second collision · CommandTypeRegistry validator integration · environment=testnet hardcoded
ProposalServiceCreateApproveTest	10	happy path · proposalId regex · proposal-not-found · state-not-risk_proposed · already-decided · proposal_json-not-cached · hardConfirm mismatch (5 cases) · confidence-override-required · confidence-override-accepted (audit flag) · overrides whitelist (rejects strategy_group)
ProposalServiceCreateRejectTest	6	happy path · proposalId regex · proposal-not-found · state-not-risk_proposed · already-decided · reason non-empty
IdempotencyTests	4	request: same-second double-click returns existing · approve+reject share key · reject after approve returns approve's command · DB-Unique-Constraint blocks 2nd INSERT
MainnetGuardTest	3	Service payload always has `environment='testnet'` · Service rejects payload with environment='mainnet' explicit · CommandTypeRegistry validator blocks mainnet
AuditPrefixTest	3	All audit_events have `managed.` prefix · no baseline.* / runtime_config.* leakage · scrubber called for metadata
DbTransactionAtomicityTest	3	failed audit insert rolls back command · failed command insert rolls back nothing (atomicity preserved) · no orphan rows
ProposalIdRegexParityTest	2	PHP regex matches MH-3b Bot regex char-for-char · accept/reject cases match Bot tests
CustomExceptionTests	4	ProposalAlreadyDecidedException · HardConfirmMismatchException · ConfidenceOverrideRequiredException · ProposalNotCachedException (new for R1)
Total	**~42 Tests**	(etwas über Roadmap-Schätzung ~30)

7 — Betroffene Dateien (Final-Inventory)

Neue Files

Datei	Status	Kategorie
`gui/app/Services/Managed/ProposalService.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/app/Exceptions/ProposalNotCachedException.php`	**NEU**	Custom Exception (R1)
`gui/tests/Feature/ProposalServiceCreateRequestTest.php`	**NEU**	Test
`gui/tests/Feature/ProposalServiceCreateApproveTest.php`	**NEU**	Test
`gui/tests/Feature/ProposalServiceCreateRejectTest.php`	**NEU**	Test
`gui/tests/Feature/ProposalServiceIdempotencyTest.php`	**NEU**	Test
`gui/tests/Feature/ProposalServiceMainnetGuardTest.php`	**NEU**	Test
`gui/tests/Feature/ProposalServiceAuditPrefixTest.php`	**NEU**	Test
`gui/tests/Feature/ProposalServiceAtomicityTest.php`	**NEU**	Test
`gui/tests/Feature/ProposalIdRegexParityTest.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` (wiederverwendet)
- `gui/app/Models/Command.php` / `gui/app/Models/AuditEvent.php` / `gui/app/Models/ManagedProposal.php` / `gui/app/Models/ManagedAssetHistory.php`
- `gui/app/Enums/ManagedProposalState.php`
- `gui/database/migrations/*` (keine neue Migration)
- **0 Bot-Side Touches** (trading/*.py)
- **0 docker cp**, **0 Restart**, **0 Migration**

Erwartete LOC: ~450 Service + ~120 Exceptions (5 × ~25) + ~900 Tests = **~1500 LOC total**.

8 — Erlaubte vs. Verbotene Service-Methoden

ERLAUBT

- `createRequestCommand(string $asset, int $userId, ?string $intent, ?string $capturedVia): Command`
- `createApproveCommand(string $proposalId, int $userId, string $variant, array $overrides, string $hardConfirm, bool $confidenceOverride): Command`
- `createRejectCommand(string $proposalId, int $userId, string $reason): Command`
- private static helpers für Validierung (validateProposalId, validateVariant, validateHardConfirm, validateAsset)
- private `writeAudit()` helper (Pattern aus ApplyBaselineService)
- private `buildXxxIdempotencyKey()` helpers
- READ-only `ManagedProposal::find($proposalId)` für approve/reject Validation

VERBOTEN

- `UPDATE managed_proposals` (Worker MH-6 only)
- `INSERT managed_proposals` (Worker MH-6 only — siehe R6)
- `INSERT managed_assets_history` (Worker MH-6 only)
- `file_get_contents()` / `file_put_contents()` auf risk_proposals/managed_state/baseline (kein FS-Touch)
- `shell_exec` / `exec` (kein Process-Spawn)
- `Command::update()` (Worker setzt status; Service nur INSERT-pending)
- direkter SQL bypass of Eloquent (alles via Models + DB::transaction)
- `environment != 'testnet'` Payload-Akzeptanz

9 — Stop-Regeln MH-4b

ID	Stop wenn…
MH-4b-SR-1	Service schreibt nach `risk_proposals/*.json` / `managed_state.json` / `baseline_holdings.json`
MH-4b-SR-2	Service INSERT auf `managed_proposals` oder `managed_assets_history`
MH-4b-SR-3	Service UPDATE auf `managed_proposals`
MH-4b-SR-4	Service akzeptiert `environment != 'testnet'`
MH-4b-SR-5	Service ruft `Command::update()` oder triggert Worker (`shell_exec` , `process` etc.)
MH-4b-SR-6	Service-Methode NICHT in `DB::transaction`
MH-4b-SR-7	`proposal_id` Regex weicht von MH-3b ab (Parity-Test bricht)
MH-4b-SR-8	`hardConfirm` Validierung ist case-insensitive
MH-4b-SR-9	`confidenceOverride=true` ohne audit-metadata-flag
MH-4b-SR-10	Audit-Event-Prefix ist nicht `managed.*`
MH-4b-SR-11	`proposal_json` IS NULL Pfad in createApproveCommand fehlt
MH-4b-SR-12	Service ruft `Engine.generate()` / `Writer.apply()` (Python-side — architektonisch unmöglich, aber Test-Pin)

10 — 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
`.env` snapshot	**NEIN**	keine Mutation
`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-Container Cache-Clear	**JA (sicherheitshalber)**	`php artisan config:clear && php artisan route:clear && php artisan view:clear` nach Service-Files-Add

→ **MH-4b = Reine Code+Test+Commit Phase**. Bot komplett unbehelligt.

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

Subschnitt-Optionen

Variante	Inhalt	LOC	Tests	Risk
MH-4b-monolithic	ProposalService (3 Methoden) + 5 Exceptions + 9 Test-Klassen	~1500	~42	LOW-MEDIUM
MH-4b-1 ← empfohlen	NUR `createRequestCommand` + 0 Exceptions + 4 Test-Klassen	~250	~15	LOW
MH-4b-2 (folgt)	`createApproveCommand` + 3 Exceptions + 3 Test-Klassen	~600	~17	LOW-MEDIUM (HardConfirm + Confidence + R1-Race)
MH-4b-3 (folgt)	`createRejectCommand` + 1 Exception + 2 Test-Klassen	~250	~8	LOW
MH-4b-4 (folgt)	Cross-cutting tests (atomicity / parity / audit-prefix)	~150	~9	LOW

Empfehlung: MH-4b-1 — createRequestCommand zuerst

Begründung:

1. **Maximaler Scope-Lock:** Eine Methode + 0 Exceptions = klar auditierbar

2. **Einfachster Pfad:** Keine `managed_proposals` READ (das kommt erst bei approve/reject)
3. **Pattern-Klarheit:** Service-Skelett wird in MH-4b-1 etabliert; MH-4b-2/3 folgen demselben Schema
4. **Test-First:** Idempotency + Mainnet-Guard + Audit-Prefix + Atomicity-Tests bauen direkt am Anfang
5. **Rollback-Cost minimal:** `git revert <commit>` + die `managed_proposals` Migration bleibt intakt (MH-4a unverändert)

Pre-Implementation-Q an Operator

1. **MH-4b-1 (NUR createRequestCommand) oder MH-4b-monolithic?** ← **Empfehlung MH-4b-1** (max scope-lock + Pattern-Etablierung)
2. **R6 architektonisch bestätigen:** Service INSERTs **NICHT** in `managed_proposals` (Worker MH-6 macht das). ← **Empfehlung JA bestätigen** — vermeidet "tentative proposal_id" Problematik
3. **`captured_via`-Parameter** in createRequestCommand: pflicht oder optional? Roadmap-Pattern ist optional. ← **Empfehlung optional** (Default `'gui_operator_request'`)
4. **`intent`-Parameter**: free-form string oder enum? ← **Empfehlung free-form mit Length-Cap 64 chars** (Operator soll kurz beschreiben können)
5. **Service-Namespace:** `App\Services\Managed\ProposalService` oder `App\Services\ProposalService` (flach)? Existing baselineservice ist `App\Services\Apply\Baseline\`. ← **Empfehlung `App\Services\Managed\`** für Konsistenz mit Apply/Baseline-Nesting

Scope-Größe + Risiko-Bewertung MH-4b-1

Aspekt	Bewertung
LOC	~250 (Service-Skelett + 1 Methode) + ~250 (Tests)
Tests	~15
Komplexität	Pattern-Mirror ApplyBaselineService::createApplyCommand
Blast-Radius	NULL — INSERT-only auf `commands` + `audit_events` (existing tables)
Roll-Back-Cost	minimal: `git revert` + DB hat 0 neue Rows (alles ist pending bis Worker läuft)
Dependencies-Klärung	R6 (managed_proposals INSERT-Strategie) — Operator approval needed
Backup-Pflicht	NEIN
Migration-Pflicht	NEIN
Restart-Pflicht	NEIN
Mainnet-Touch	NEIN
docker cp	NEIN

STOP vor Implementierung. Erwarte Operator-GO mit Subschnitt-Wahl (MH-4b-1 empfohlen) + Q1-Q5-Approval, insbesondere **R6 architektonisches Verdict** (Service INSERTs NICHT in managed_proposals).