MH-4c ManagedStateService — Plan-Review

Projekt: Steve-TradingBot · Phase: RECON-MH-4c · Author: claude-opus-4-7[1m]
Generated: 2026-05-12 21:17 UTC · master HEAD: `93f4026` (MH-4b-3 closed)
Status: **NO CODE** Plan-Review only — Operator-GO erforderlich vor MH-4c Code-Phase

1 — Live-State Verification

Check	Soll	Ist	OK
master HEAD	`93f4026`	`93f4026 mh-4b-3: add proposal reject command service`	✓
`git status`	clean	empty output	✓
Bot in-container PID	29984	unverändert	✓
Worker Host PID	(egal)	**2486135** (pre-existing Watchdog-Respawn, nicht MH-4c-blocking)	⚠
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	✓
`state/risk_proposals/`	absent	not present	✓
Tracebacks last 200 lines	0	0 matches	✓

2 — Konsumierte Artefakte

Komponente	Status
MH-1: `pause_managed_asset` / `resume_managed_asset` / `release_managed_asset` CommandTypes + Validators	`ea11637`
MH-4a: `managed_assets_history` Table + Models + Enum	`83dafca`
MH-4b-1/2/3: ProposalService-Pattern	`3fd7846` / `be9cab7` /
`ManagedProposalState` Enum	MH-4a
Worker-Handler `_handle_pause/_handle_resume/_handle_release`	MH-6
Filament Wizard mit Pause/Resume/Release-UI	MH-5

Wichtig: ProposalService bleibt unverändert. MH-4c liefert eine **eigene neue Klasse** `ManagedStateService` (separate File).

3 — MH-4c Scope-Rekonstruktion

Roadmap-Definition (canonical aus `05_commandbus_worker §8`)

```
class ManagedStateService
{
    public function createPauseCommand(string $asset, int $userId): Command
{ ... }
    public function createResumeCommand(string $asset, int $userId): Command
{ ... }
    public function createReleaseCommand(string $asset, int $userId, string
$strategy): Command { ... }
}
```

3 Service-Methoden im Detail

a) `createPauseCommand(string $asset, int $userId)`

- **Semantik:** Operator pausiert managed-asset → Worker setzt später state `managed_active` → `managed_paused`
- **Validation:** asset regex, userId positiv
- **Idempotency-Key:** `mh:pause:{asset}:{YmdHis}` (per-click distinct, NICHT shared)
- **Hard-Confirm:** ❌ **NEIN** (low-risk, reversibel via resume)
- **CommandType:** `pause_managed_asset` (Registry timeout 120s)
- **Audit:** `managed.asset_pause_requested`

b) `createResumeCommand(string $asset, int $userId)`

- **Semantik:** Operator resumed → Worker setzt later state `managed_paused` → `managed_active`
- **Validation:** asset regex, userId positiv
- **Idempotency-Key:** `mh:resume:{asset}:{YmdHis}`
- **Hard-Confirm:** ❌ **NEIN**
- **CommandType:** `resume_managed_asset`
- **Audit:** `managed.asset_resume_requested`

c) `createReleaseCommand(string $asset, int $userId, string $strategy, string $hardConfirm)`

- **Semantik:** Operator released → Asset verlässt managed-Lebenszyklus. **High-Risk, semantisch irreversibel**
- **Validation:** asset regex, userId positiv, strategy whitelisted, hardConfirm exact-match
- **Idempotency-Key:** `mh:release:{asset}:{YmdHis}`
- **Hard-Confirm:** ✅ **JA** — `<asset>:release:<strategy>` (case-sensitive)
- **CommandType:** `release_managed_asset`
- **Audit:** `managed.asset_release_requested`

Strategy-Whitelist für Release

Vorschlag — 2 Werte:

- **`to_frozen`**: Asset returns to baseline `frozen` policy. Worker setzt state `managed_*` → `managed_released` → später `frozen` (Two-File-Atomic, MH-6 Worker-Job).
- **`mark_released_only`**: State-flip auf `managed_released` ohne Policy-Wechsel. Asset bleibt in `managed_released` für Audit-Trail. Operator entscheidet später separat über frozen-Revert.

Welche DB-Reads/Writes sind erlaubt?

READ (Phase 1, empfohlen):

- ❌ NICHT `managed_proposals` (wrong domain — proposals vs. managed-assets)
- ❌ NICHT `managed_assets_history` (Service schreibt da nichts; muss da auch nichts lesen in Phase 1)

- ❌ NICHT `managed_state.json` (Hybrid C — GUI liest keine Bot-Files)

→ **MH-4c-Phase-1 macht KEINE State-Cache-Lookup**. Service ist pure command-builder. Worker MH-6 ist final state-validator.

WRITE:

- ✅ INSERT `commands` (status=pending)
- ✅ INSERT `audit_events` (managed.asset_*_requested)
- ❌ NIE `managed_proposals`
- ❌ NIE `managed_assets_history` (Worker MH-6)
- ❌ NIE State-Files

State-Transitions: nur vorbereitet, nicht ausgeführt

Command	Vorbereitet von MH-4c	Ausgeführt von MH-6
pause	INSERT pending	managed_active → managed_paused
resume	INSERT pending	managed_paused → managed_active
release(to_frozen)	INSERT pending	managed_* → managed_released → frozen (Two-File-Atomic)
release(mark_released_only)	INSERT pending	managed_* → managed_released (single file)

4 — Konfliktcheck Sonder-Anforderungen

Anforderung	MH-4c-Touch	Verdict
Kein `managed_state.json` write	Service touched keine state files	**konform**
Kein `baseline_holdings.json` write	Service touched keine state files	**konform**
Kein ProposalWriter-Aufruf	architekturell unmöglich + Source-Grep Pin	**konform**
Kein Worker-Handler	command_worker.py bleibt unverändert	**konform**
Kein Bot-Wiring	main.py / paper_trade.py / live_trade.py / risk_manager.py bleiben unverändert	**konform**
Kein Filament Wizard	UI ist MH-5	**konform**
Kein Mainnet	hardcoded `environment='testnet'`	**konform**
Kein `runtime_config`-touch	nicht im Service-Scope	**konform**

5 — Empfohlene Lieferung

Service-API

```
namespace App\Services\Managed;

class ManagedStateService
{
    public function __construct(private readonly CommandTypeRegistry
$commandRegistry) {}

    public function createPauseCommand(string $asset, int $userId): Command;
    public function createResumeCommand(string $asset, int $userId): Command;
    public function createReleaseCommand(
        string $asset, int $userId, string $strategy, string $hardConfirm
    ): Command;

    // Public static helper (reusable in MH-5 Wizard)
    public static function buildReleaseHardConfirmString(
        string $asset, string $strategy
    ): string;  // returns "<asset>:release:<strategy>"

    // Constants
    public const REQUEST_ENVIRONMENT     = 'testnet';
    public const AUDIT_SUBJECT_TYPE      = 'ManagedAsset';
    public const PAUSE_TIMEOUT_SECONDS   = 120;
    public const RESUME_TIMEOUT_SECONDS  = 120;
    public const RELEASE_TIMEOUT_SECONDS = 120;
    public const ALLOWED_RELEASE_STRATEGIES = ['to_frozen',
'mark_released_only'];
}
```

Erlaubte Methoden

- 3 public command-builder
- 1 public static helper (Hard-Confirm-Builder für release)
- private `validateAsset` (mirror aus ProposalService)
- private `validateUserId` (mirror)
- private `validateStrategy` (NEU — whitelist-check)
- private `validateHardConfirm` (NEU für release)
- private `writeAudit` (mirror)
- private `buildXxxIdempotencyKey` (3 builders)

Verbotene Methoden

- `Command::update()`
- `ManagedProposal::*` (different domain)
- `ManagedAssetHistory::*` (Worker MH-6)
- File-IO / Subprocess
- `environment != 'testnet'`

Exceptions

0 neue Exception-Klassen. Reuse aus MH-4b:

- `InvalidCommandPayloadException` für asset/userId/strategy/hardConfirm-Validierung
- Keine state-spezifischen Exceptions in MH-4c-Phase-1 (kein state-cache-lookup)

Payload-Schemas

pause / resume:

```
{
  "environment":  "testnet",
  "asset":        "<asset>",
  "requested_by": <userId>,
  "expires_at":   "<iso8601-z>"
}
```

release:

```
{
  "environment":  "testnet",
  "asset":        "<asset>",
  "strategy":     "to_frozen|mark_released_only",
  "hard_confirm": "<asset>:release:<strategy>",
  "requested_by": <userId>,
  "expires_at":   "<iso8601-z>"
}
```

Idempotency-Keys

Command	Key-Pattern	Verhalten
pause	`mh:pause:{asset}:{YmdHis}`	per-click distinct — same-second double-click returns existing
resume	`mh:resume:{asset}:{YmdHis}`	per-click distinct
release	`mh:release:{asset}:{YmdHis}`	per-click distinct

NICHT shared wie decide. Operator kann legitim mehrfach pause/resume/release klicken — jeder Click = neuer Command + neuer Audit-Trail.

6 — Test-Plan

Test-Klasse	Anzahl	Inhalt
ManagedStateServiceCreatePauseTest	8	happy path · payload env=testnet · payload alle keys · audit prefix managed.* · audit event_type=managed.asset_pause_requested · asset regex · userId positive · same-second idempotency
ManagedStateServiceCreateResumeTest	6	happy path · payload · audit · asset regex · userId · same-second idempotency
ManagedStateServiceCreateReleaseTest	12	happy path · payload · audit event_type=managed.asset_release_requested · strategy whitelist (to_frozen / mark_released_only) · strategy reject unknown · strategy reject empty · hardConfirm exact match · hardConfirm asset mismatch · hardConfirm strategy mismatch · hardConfirm case mismatch · hardConfirm empty · same-second idempotency
ManagedStateServiceBoundaryTests	4	no managed_proposals write · no managed_assets_history write · no file/subprocess tokens · no ProposalWriter/Engine refs
ManagedStateServiceIdempotencyTests	4	pause and resume have distinct keys (not shared) · multiple pause within same second returns existing · different-second different commands · per-asset isolation
ManagedStateServiceMainnetGuardTest	3	pause payload environment=testnet hardcoded · resume same · release same
ManagedStateServiceAtomicityTest	1	failed audit insert rolls back command (test against release)
BuildReleaseHardConfirmStringTest	3	builds `<asset>:release:<strategy>` · case preserved · 2 strategies
AuditPrefixTest	3	every audit prefix `managed.*` · per-type event_type pinned
ManagedStateServiceConstantsPinned	2	ALLOWED_RELEASE_STRATEGIES pinned · timeout-Konstanten pinned
Total	**~46 Tests**	(kleiner als approve weil keine confidence/state-cache lookup)

7 — Betroffene Dateien

Neue Files

Datei	Status
`gui/app/Services/Managed/ManagedStateService.php`	**NEU**
`gui/tests/Feature/ManagedStateServiceCreatePauseTest.php`	**NEU**
`gui/tests/Feature/ManagedStateServiceCreateResumeTest.php`	**NEU**
`gui/tests/Feature/ManagedStateServiceCreateReleaseTest.php`	**NEU**
`gui/tests/Feature/ManagedStateServiceBoundaryTest.php`	**NEU**

Unverändert

- `gui/app/Services/Managed/ProposalService.php` (MH-4b komplett intact)
- `gui/app/Services/CommandTypeRegistry.php` (3 Validators existieren ab MH-1)
- alle MH-4a/4b Modelle + Migrations
- Bot-Side trading/*.py — **0 Touches**

Erwartete LOC: ~450 Service + ~750 Tests = ~1200 LOC

8 — Stop-Regeln MH-4c

ID	Stop wenn…
MH-4c-SR-1	Service UPDATE auf `managed_proposals`
MH-4c-SR-2	Service INSERT in `managed_assets_history`
MH-4c-SR-3	Service file IO / subprocess
MH-4c-SR-4	Service akzeptiert `environment != 'testnet'`
MH-4c-SR-5	Service ruft Worker / Engine / Writer / Reader
MH-4c-SR-6	Service-Methode NICHT in `DB::transaction`
MH-4c-SR-7	release strategy ist case-insensitive whitelist-check
MH-4c-SR-8	release Hard-Confirm ist case-insensitive
MH-4c-SR-9	Audit-Event-Prefix ist nicht `managed.*`
MH-4c-SR-10	pause/resume idempotency key shared mit anderen (nur per-click distinct erlaubt)
MH-4c-SR-11	release whitelist akzeptiert unbekannte strategy
MH-4c-SR-12	pause/resume haben Hard-Confirm (architectural decision: nur release)

9 — Backup / Migration / Restart-Bedarf

Aktion	Pflicht?
pg_dump GUI-DB	**NEIN**
State-Files snapshot	**NEIN**
DB Migration	**NEIN**
Bot-Restart	**NEIN**
Worker-Restart	**NEIN**
docker cp	**NEIN**

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

10 — Risiken-Übersicht

#	Risiko	Severity	Mitigation
R1	Service-Methode ohne state-validation lässt Operator pause auf frozen-asset → Worker MH-6 wird ablehnen	LOW	UX-Issue, kein Service-Bug. Filament-Wizard (MH-5) zeigt nur erlaubte Aktionen pro state. Worker wirft `InvalidStateTransitionException` zum Audit. Symmetric mit `createRequestCommand` Pattern.
R2	Release Strategy-Drift Service↔Worker	LOW	Whitelist als Service-Konstante + future Worker-side mirror. Parity-Test in MH-6.
R3	Operator klickt pause+resume+pause schnell hintereinander	LOW	Per-click distinct Idempotency-Keys mean alle 3 Commands persisted. Audit-Trail vollständig. Worker verarbeitet sequenziell.
R4	Release Hard-Confirm-Format-Verwechslung mit approve `<asset>:<variant>:<sha8>`	LOW	Distinct Literal `release` in mitte verhindert; sha8 not used (asset hat kein proposal_sha256-Kontext).
R5	strategy=Unknown silent-akzeptiert	LOW	Whitelist-check explicit; Test pinned.
R6	DB::transaction-Rollback bei AuditEvent failure	LOW	Pattern bewährt aus MH-4b-1/2/3
R7	pause/resume ohne Hard-Confirm → Operator-Fehler kann mehrere assets versehentlich pause-storm	LOW	Filament-UI bietet Confirm-Dialog ohne Hard-Confirm-String; reversibel via resume
R8	release ohne strategy-Pflicht durchgewunken	LOW	Service erfordert strategy als 3. Param; PHP-type-system erzwingt non-null
R9	Strategy `mark_released_only` lässt Asset in undefiniertem state hängen	MEDIUM	UX-Doc-Issue für MH-5 Wizard; Service trägt strategy nur weiter; Worker MH-6 ist authoritative
R10	ProposalService::buildHardConfirmString und ManagedStateService::buildReleaseHardConfirmString drift	LOW	Beide separate static methods; Test pinnt jeweils Format

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

Subschnitt-Optionen

Variante	Inhalt	LOC	Tests	Risk
MH-4c-monolithic ← empfohlen	ManagedStateService mit allen 3 Methoden + buildReleaseHardConfirmString + Tests	~1200	~46	LOW
MH-4c-a	NUR createPauseCommand + happy path Tests	~350	~10	LOW
MH-4c-b (folgt)	createResumeCommand + Tests	~250	~10	LOW
MH-4c-c (folgt)	createReleaseCommand + buildReleaseHardConfirmString + Tests	~500	~20	LOW

Empfehlung: MH-4c-monolithic

Begründung:

1. **Pattern voll etabliert** durch MH-4b — keine Discovery
2. **3 Methoden gehören semantisch zusammen** als asset-lifecycle commands; sub-cuts würden Wizard-Build (MH-5) unvollständig deployen
3. **Service hat NICHTS Komplexes** — pure command-builder ohne state-cache lookup
4. **Release ist die einzige nicht-triviale Methode** (Hard-Confirm + Strategy); pause/resume sind trivial
5. **Rollback-Cost minimal:** `git revert` + 0 DB-Rows

Pre-Implementation-Q an Operator

1. **MH-4c-monolithic** oder Sub-Cut? ← **Empfehlung monolithisch**
2. `ManagedStateService` als separate Klasse ODER Methoden in `ProposalService` einbauen? ← **Empfehlung separate Klasse** (saubere domain-separation)
3. **Release Strategy-Whitelist:** `['to_frozen', 'mark_released_only']` — beide? andere Werte? ← **Empfehlung beide**, Operator kann später erweitern
4. **Release Hard-Confirm Format:** `<asset>:release:<strategy>` (3-Teil mit strategy-binding) ODER `<asset>:release` (2-Teil simpel)? ← **Empfehlung 3-Teil** (bindet an strategy-Wahl)
5. **pause/resume Hard-Confirm:** NEIN (low-risk, reversibel)? ← **Empfehlung JA, NEIN bestätigen**
6. **State-Cache-Lookup-Strategie:** Phase 1 NO lookup (Worker = authoritative) ← **Empfehlung Phase 1 NO lookup**
7. **Code-Reuse aus ProposalService:** Duplizieren ODER Trait extrahieren? ← **Empfehlung Duplizieren** (kein Premature-Abstraction)
8. **0 neue Exception-Klassen** bestätigen? ← **Empfehlung JA** (InvalidCommandPayloadException reicht)
9. **audit_event_types:** `managed.asset_pause_requested` / `managed.asset_resume_requested` / `managed.asset_release_requested` bestätigen?

Scope-Größe + Risiko-Bewertung MH-4c-monolithic

Aspekt	Bewertung
LOC	~450 Service + ~750 Tests = ~1200
Tests	~46
Komplexität	LOW (massive Reuse aus MH-4b-Pattern)
Blast-Radius	NULL — INSERT-only auf `commands` + `audit_events`
Roll-Back-Cost	minimal: `git revert` + 0 neue DB-Rows
Dependencies-Klärung	Q-3 (Strategy-Whitelist) + Q-4 (Hard-Confirm-Format)
Backup-Pflicht	NEIN
Migration-Pflicht	NEIN
Restart-Pflicht	NEIN
Mainnet-Touch	NEIN
docker cp	NEIN

STOP vor Implementierung. Erwarte Operator-GO mit Subschnitt-Wahl (monolithic empfohlen) + Q1-Q9-Approval, insbesondere Strategy-Whitelist + Hard-Confirm-Format + State-Cache-Lookup-Strategie.