RECON-Managed-Holdings · Architektur- und Scope-Bericht

Phase: Pre-Scope / Architecture-Thinking **Stand:** 2026-05-10 **Output-Status:** Architektur-Bericht; **kein** Code, **kein** File-Touch außerhalb dieses Doks **Bezug:** komplettiert die Q-T1..Q-T6 aus `recon_status_pin.md §3.3`; voraus zu RECON-2.4 / RECON-2.5 **Vorbild:** Pattern aus `mainnet_preflight_recon.md` + G10-Apply-State-Machine + RECON-2.3 PHP/Bot-Trennung

---

0 · Executive Summary

Der Operator möchte bestehende Spot-Holdings explizit an den Bot übergeben können — aber **NICHT** durch automatische Übernahme. Stattdessen läuft jeder Coin durch einen mehrstufigen Approval-Flow:

```
frozen → bot-analysiert → operator-bestätigt → managed_active
```

Bot ist Berater, Operator ist final authority. Mainnet bleibt blockiert bis RECON-2.5 Sign-Off.

Dieses Dokument liefert: - Erweiterte State-Machine (9 Zustände statt der ursprünglichen 4) - Race-Conditions + State-Drift-Vektoren + Bot-vs-Operator-Konflikte - Konflikt-Audit gegen DR-1..DR-13 - Datenmodell für `managed_state.json` + `risk_proposals/<id>.json` - 16 neue Audit-Events unter `managed.*`-Prefix - 15 offene Operator-Fragen (Q-MH-1..15) mit Default-Empfehlungen - 9-Phasen-Roadmap (RECON-MH-0..9) - 5 neue Stop-Regeln (SR-11..15)

Vor Start von Code-Phasen müssen entschieden sein: - Q-MH-14 (Worker-Daemon-Aktivierung als Vorbedingung) - DR-7-Konflikt-Lösung (wallet_signature bei policy-Wechsel) - T-SPLIT-3 t3-Allowlist-Regel für proposal_engine - CAP-1 investable_capital ↔ managed.max_allocation Interaktion

---

1 · Aktive Risiko- und Konflikt-Analyse

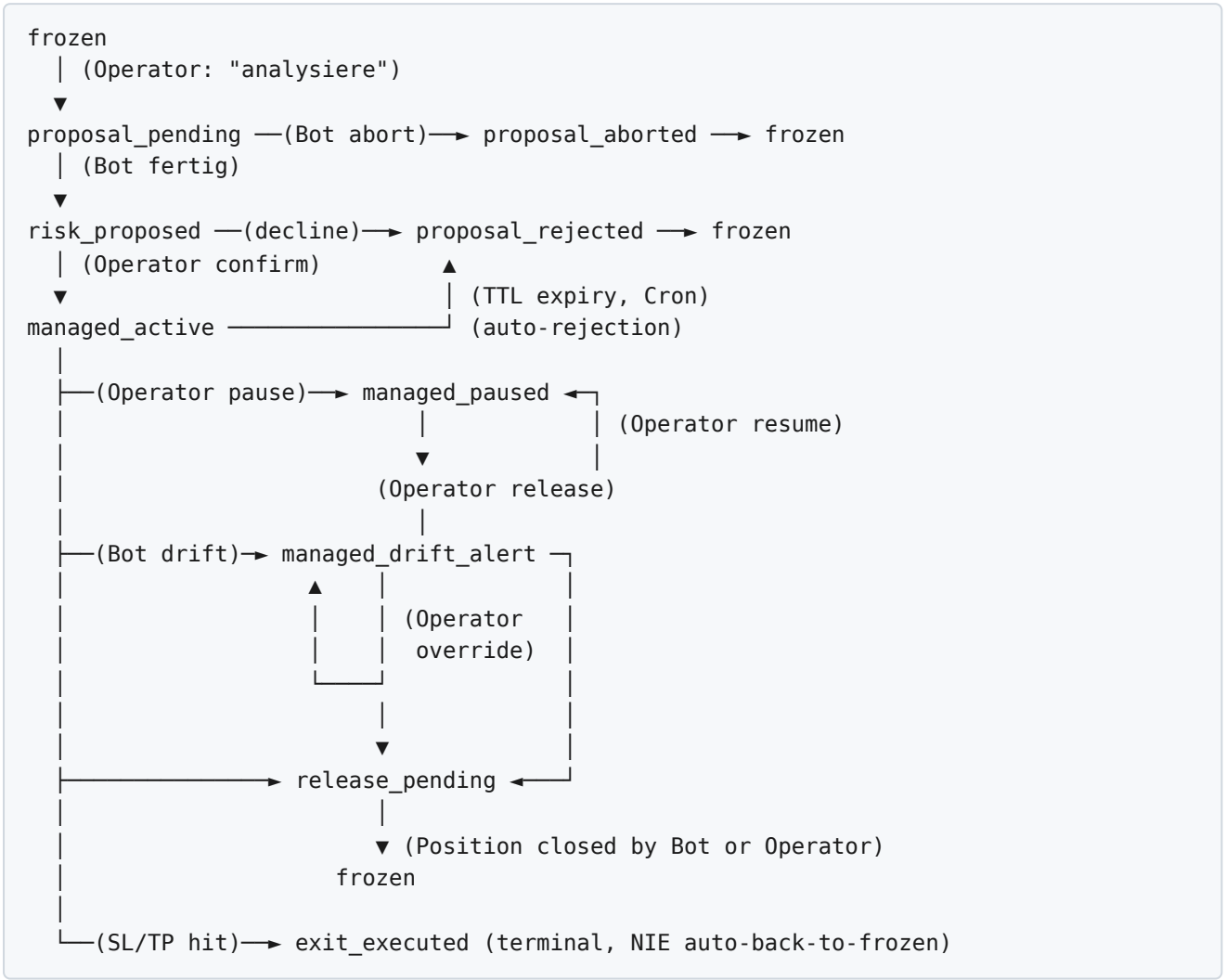
1.1 Erweiterung der State-Machine (vier zusätzliche Stationen)

Der ursprüngliche Vorschlag `frozen → risk_proposed → operator_confirmed → managed_active` ignoriert vier reale Lifecycle-Stationen:

1. `proposal_pending` — zwischen Operator-Trigger und Bot-Output. Markt-Daten + ATR + OB-Depth zu sammeln dauert Sekunden bis Minuten. Ohne diesen Zustand klickt der Operator zweimal und löst doppelte Proposals aus.
2. `managed_paused` — Operator möchte Bot temporär stoppen ohne Position zu schließen (z.B. vor Earnings, vor FOMC).

3. `managed_drift_alert` — Bot detected Anomalie (qty/price/regime), pausiert sich SELBST, wartet auf Operator. Bot darf NIEMALS autonom verkaufen.
4. `release_pending` — Operator hat „release" gewählt, aber offene Position muss erst geschlossen werden bevor `frozen` final terminal ist.

Vollständige State-Machine:



States im Überblick:

State	Beschreibung	Wer setzt?	Terminal?
`frozen`	Bot ignoriert, Default	Operator / System (TTL) / Default	Nein (Re-Entry möglich)
`proposal_pending`	Bot generiert Proposal	Operator triggert	Nein
`risk_proposed`	Proposal vorhanden, Operator-Action erwartet	Bot finalisiert	Nein
`proposal_aborted`	Bot konnte Proposal nicht erstellen	Bot (Balance/Markt down)	Ja → frozen
`proposal_rejected`	Operator declined oder TTL expired	Operator / System	Ja → frozen
`managed_active`	Bot trackt + tradet	Operator confirmed	Nein
`managed_paused`	Operator pausiert, kein neuer Trade	Operator	Nein
`managed_drift_alert`	Bot pausiert sich selbst wegen Drift	**Bot autonomously**	Nein
`release_pending`	Graceful exit angefordert	Operator	Nein
`exit_executed`	SL/TP hit	Bot (per Proposal-SL/TP)	Ja, Operator entscheidet ob neu propose oder bleibend frozen

Übergangs-Tabelle (Verantwortliche):

From → To	Actor
frozen → proposal_pending	Operator
proposal_pending → risk_proposed	Bot
proposal_pending → proposal_aborted	Bot
risk_proposed → managed_active	Operator
risk_proposed → proposal_rejected	Operator
risk_proposed → frozen (TTL)	System (Cron)
managed_active → managed_paused	Operator
managed_paused → managed_active	Operator
managed_paused → release_pending	Operator
managed_active → managed_drift_alert	**Bot** (autonomously)
managed_drift_alert → managed_active	Operator (override)
managed_drift_alert → release_pending	Operator
managed_active → exit_executed	Bot (SL/TP hit)
release_pending → frozen	System (after position closed)
exit_executed → frozen	**Operator** (manuelle Entscheidung; NIE automatisch — DR-11)
exit_executed → proposal_pending	Operator (re-engage)

Durable Rule: NUR Operator (oder System-Timeout-Job) darf zu `frozen` zurück. Der Bot kann NIEMALS `managed_*` → `frozen` autonom (DR-11).

1.2 Race Conditions

#	Race	Schadenpotential	Mitigation
R1	Operator approved `risk_proposed` zur gleichen Zeit wie Bot `drift_detected`	beide schreiben `managed_state.json[asset].state` → last-write-wins	CommandBus + DB-row-level lock auf `managed_assets.<asset>`
R2	Wallet-Balance ändert sich während Bot Proposal generiert (Operator macht Sell auf Binance)	Proposal basiert auf qty=5, real qty=0 → fehlerhafte synthetic_entry	Engine refetcht balance unmittelbar vor file-write; bei qty=0 → state `proposal_aborted_no_balance`
R3	Zwei Admins approven dasselbe Proposal	doppeltes Promote, doppelter audit	idempotency_key auf `proposal_id`; zweiter Command wird `precondition_rejected` mit `proposal_already_decided`
R4	Bot's update_prices Cycle SL-trigger same time as Operator clicks Pause	Pause-Command pending in commands-Tabelle, Bot-Cycle liest noch alten state → Sell läuft trotz Pause	**Worker-Daemon** statt manuelles `--once` (siehe Q-MH-14)
R5	Worker `--once` läuft NACH Operator-Pause aber VOR neuem Bot-Cycle → Pause-Command verarbeitet, Bot liest aktualisierten state — OK	kein Schaden	–
R6	TTL-Expiry-Job läuft zur gleichen Zeit wie Operator approved	proposal kann doppelt-decided werden	dieselbe row-level lock wie R1

Mitigation-Pattern: - **alle State-Mutationen via CommandBus** mit `FOR UPDATE SKIP LOCKED` (G6.5 Pattern) - **idempotency_key auf proposal_id** für approve/reject - **Bot's update_prices liest managed_state am Cycle-Anfang** — atomarer Snapshot pro Cycle - **R4 ist nicht vermeidbar OHNE Worker-Daemon** — manuell `--once` getriggerter Worker hat per Definition Latenz. Q-MH-14 entscheidet.

1.3 State-Drift-Vektoren

#	Drift	Detection	Reaktion
D1	`managed_state.qty` ≠ `state['positions'][SYM/USDT].qty` (Bot DCA hat qty erhöht)	per-cycle Reconciliation	bei diff > X% → managed_drift_alert
D2	`state['positions'][SYM/USDT].qty` > `fetch_balance()[SYM]` (Wallet-Sell extern)	per-cycle	managed_drift_alert + audit
D3	`risk_proposals/<id>.json` exists, aber `managed_state.pending_proposals` referenziert ihn nicht	scheduled-job	log warn, Datei nach `attic/` verschieben
D4	`baseline_holdings.json` policy=managed aber kein managed_state-Eintrag	bootstrap-Boot-Check	Bot startet nicht, sys.exit(1) — DR-Konflikt
D5	Asset im Wallet, aber NICHT in baseline + NICHT in managed_state — neuer Airdrop	bootstrap	fall under default_policy_for_unlisted (frozen)

Mitigation: Reconciliation-Job als per-cycle Hook in `main.py` ODER separate Cron. Schreibt Audit-Event `managed.drift_detected` mit `drift_kind ∈ {qty, price, regime, wallet_external}`.

1.4 Bot-vs-Operator-Konflikte

C1 — Operator override SL während Bot's Proposal SL aktiv: - managed_state speichert HISTORISCHEN synthetic_entry/SL/TP (Promote-Zeitpunkt-Werte) - state['positions'] speichert LIVE-Werte (mit Override) - Diese sollen **NICHT** im Sync sein. Operator-Override schreibt Audit `managed.operator_override` mit dem Diff. Tests pinnen das Pattern.

C2 — Bot will SELL (SL hit), Operator will Pause same Cycle: - Pause-Command schreibt synchron managed_state (Q-MH-14 Pattern) - Bot's update_prices Cycle-Anfang liest managed_state.state - Wenn state=managed_paused beim Read → kein Sell-Trigger - Race-Window: Worker-Latenz. Reduzierbar nur durch Daemon-Worker.

C3 — Operator setzt managed→frozen via Apply (Versehen): - DR-11 Enforcement: ApplyBaselineService MUSS prüfen, ob Asset currently managed_active ist - Wenn ja, baseline-Apply darf policy=frozen NUR setzen, wenn ZUSÄTZLICH ein release_managed_asset Command ausgeführt wurde - Audit-Event `managed.policy_change_blocked` wenn Versuch erkannt

---

1.5 Mainnet-spezifische Gefahren

#	Gefahr	Mitigation
M1	aggressive Variante auf Mainnet zu wide SL → Riesenverluste	aggressive Variante auf Mainnet **disabled** (Q-MH-13)
M2	wallet-signature mismatch nach promote (compute_account_hash sieht policy-Wechsel)	Algorithmus muss policy-Status ignorieren ODER managed-promote setzt baseline-policy auch im selben Command (transactional)
M3	Bot verkauft pre-existing Coin auf Mainnet, weil synthetic_entry zu eng am SL	Validator: synthetic_entry MUSS ≥ 5% Abstand zu SL haben (Mainnet-strenger als Testnet)
M4	Operator approved managed mit max_allocation > investable_capital (CAP-1-Konflikt)	Sizer kapppt bei investable_capital + audit `managed.allocation_capped`
M5	t3_copy_trading-Recommendation während t3 noch „planned/disabled"	proposal_engine validiert allowed_strategy_groups; Fall-back zu t1_core mit Warning

---

1.6 Brechende existierende Komponenten

Komponente	Was bricht	Anpassung
`baseline_bootstrap.py` auto_import	importiert ALLE policy=managed assets als synthetic position	nur `managed_state.state == "managed_active"` importieren
`scanner/universe.py` filter	filtert `base in baseline.frozen_assets()`	erweitern: `base in {frozen} OR base in managed_state.NOT(managed_active)`
`paper_trade.execute_buy` Guard	`_is_base_baseline_frozen`	umbenennen zu `_is_base_unmanaged` , prüft beide Dateien
`live_trade.execute_buy` Guard	dito	dito
`balance_provider` quote-subtract	subtract frozen quote	offene Frage: managed-quote auch subtract? Vermutlich NEIN weil Bot soll Quote benutzen
RECON-2.3 Filament Page	zeigt nur `baseline.*` Audit	Section „managed.*" Audit ergänzen
RECON-2.3c Bot-Worker	2 Handler	+7 Handler

1.7 Konflikte mit existierenden Durable Rules

DR	Konflikt?	Anpassung
DR-1 (nur via CommandBus)	✅ kompatibel — managed_state.json + risk_proposals/* erben Pattern	DR-1 textuell auf „alle bot-side-state-files" erweitern
DR-2 (managed nie default)	✅ unverändert	–
DR-3 (testnet only)	✅ alle 7 neuen Command-Types testnet only	–
DR-4 (Hard-Confirm dynamisch)	✅ approve_managed_proposal Confirm-String dynamisch	Q-MH-11 entscheiden
DR-5 (Backup-before-mutate)	✅ Writer-Pattern reused	–
DR-6 (Audit-Prefix strict)	✅ `managed.*` Prefix	–
DR-7 (account_hash excludes tradable_quote)	⚠ **KONFLIKT** wenn promote die baseline-Policy ändert (M2)	wallet_signature-Algorithmus überdenken
DR-8 (Watchdog clawbot)	✅ unverändert	–
DR-9 (Backup vor Live-Action)	✅ alle MH-Phasen folgen	–
DR-10 (Bot-PID Timeline)	✅ unverändert	–
DR-11 (managed niemals auto-frozen)	**NEU enforcen**	Worker-Validator: jede transition managed_* → frozen MUSS audit `actor=user_*` haben
DR-12 (Risk-Proposals versioniert)	**NEU enforcen**	proposal_engine Output validiert vor Schreib
DR-13 (managed.* Audit-Trail)	**NEU enforcen**	strict separation gepinnt durch source-grep test

---

1.8 Erweiterungsbedarf für G10 / T-SPLIT / CAP-1 / B-OUTAGE

G10: Apply-State-Machine ist Vorbild für Approval-Pattern, aber RECON-MH ist DB+JSON+CommandBus + asynchroner Bot-Step. Größerer Scope. Service-Klasse separat. Pattern wiederverwendet, Code nicht.

T-SPLIT: proposal_engine recommended `strategy_group` ∈ `{t1_core, t2_pump_dump}`. **t3_copy_trading hard-blocked** solange T-SPLIT-3 t3 als „planned/disabled" markiert. proposal_engine Allowlist-Validator nötig.

CAP-1: managed_state.max_allocation_usdt ist **Soft-Cap**. CAP-1.investable_capital ist **Hard-Cap**. Sizer kapppt mit Audit-Event wenn capped (`managed.allocation_capped`).

B-OUTAGE: CircuitBreaker `open` blockiert BUYs, aber NICHT Promotion. Promotion in offener-breaker-Phase ist erlaubt — managed_active heißt „Bot trackt", nicht „Bot kauft sofort".

---

2 · Offene Fragen an Operator (Q-MH-1..15)

#	Frage	Optionen	Default-Empfehlung
Q-MH-1	Proposal-TTL?	24h / 7d / configurable	**7d**
Q-MH-2	Wie viele Varianten generiert Bot?	1 / 2 / 3 (rec/cons/aggr)	**3** auf Testnet, **2** (rec+cons, kein aggr) auf Mainnet
Q-MH-3	Operator-Override-Tiefe?	minimal / medium / maximal	**medium** (SL/TP/max_alloc/trailing/dca, NICHT strategy_group)
Q-MH-4	release_pending Behavior?	graceful (warte SL/TP) / immediate (Bot off, position bleibt) / choice	**choice** im UI, default graceful
Q-MH-5	Multi-Asset-Bulk-Proposal?	ja / nein / Bulk-Markierung mit Einzel-Workflows	**C** (Bulk-Markierung, separate Workflows pro Asset)
Q-MH-6	Bot self-initiated Proposals?	nie / nur Alerts ohne Auto-Promote / Auto-Promote bei score>X	**B** (Bot darf vorschlagen, Operator-Confirm bleibt zwingend)
Q-MH-7	Drift-Schwellen?	qty / price / regime — welche und wieviel?	**qty>5% OR price out 2-ATR OR regime BEAR↔BULL**
Q-MH-8	Pause-Verhalten?	SL/TP active / SL/TP off / choice	**A** (SL/TP bleibt aktiv, kein neuer Trade)
Q-MH-9	Re-Engage nach exit_executed?	manual via frozen / auto-can-re-engage state	**A** (frozen, Operator muss neu propose'n)
Q-MH-10	synthetic_entry-Method?	current_price / cost_basis / VWAP / Operator-choice	**D** (Operator-choice, default current_price)
Q-MH-11	Hard-Confirm-String approve?	Asset / Asset:Variante / proposal_id-suffix	**B** `<asset>:<variant>` z.B. `SOL:recommended`
Q-MH-12	Cool-down nach reject?	nein / 24h / configurable	**24h** auf Testnet, **7d** auf Mainnet
Q-MH-13	Mainnet-Disable-Features?	aggressive variant / DCA in proposal / t2 als rec	**alle drei: disabled auf Mainnet**
Q-MH-14	Worker-Daemon-Aktivierung als Vorbedingung?	optional / required for RECON-MH	**required** — sonst R4-Race nicht akzeptabel

#	Frage	Optionen	Default-Empfehlung
Q-MH-15	managed_state Source of Truth?	JSON / DB / beide	**C** (JSON für Bot, DB als Cache + GUI-Read)

---

3 · Datenmodell-Entwurf

3.1 baseline_holdings.json (BESTEHEND, minimal-Erweiterung)

Default-Policy bleibt simpel. Lifecycle wandert in **separate** Datei `managed_state.json`. Vermeidet Vermischung von „Operator-Default-Konfiguration" mit „Live-Lifecycle-State".

Optional: `policy_lifecycle_managed: true` Flag pro Asset, signalisiert „diese Policy wird von managed_state.json überschrieben".

3.2 managed_state.json (NEU, Bot-Side)

Eigene Datei, eigener Writer/Reader, eigener CommandBus-Pfad.

```
{
  "_meta": {
    "schema_version": "recon-mh-1",
    "captured_at": "2026-05-10T...",
    "environment": "testnet"
  },
  "managed_assets": {
    "SOL": {
      "state": "managed_active",
      "current_proposal_id": "uuid-...",
      "synthetic_entry": 145.32,
      "synthetic_entry_method": "current_price",
      "stop_loss": 130.0,
      "take_profit": 165.0,
      "max_allocation_usdt": 500.0,
      "strategy_group": "t1_core",
      "promoted_at": "2026-05-10T...",
      "promoted_by_user_id": 42,
      "history": [
        {"ts": "...", "from": "frozen", "to": "proposal_pending", "actor": "user_42",
"event_type": "managed.asset_proposal_requested"},
        {"ts": "...", "from": "proposal_pending", "to": "risk_proposed", "actor":
"bot", "event_type": "managed.asset_proposal_generated"},
        {"ts": "...", "from": "risk_proposed", "to": "managed_active", "actor":
"user_42", "event_type": "managed.asset_promoted"}
      ]
    }
  },
  "pending_proposals": {
    "DOGE": {
      "proposal_id": "uuid-...",
      "state": "proposal_pending",
      "requested_at": "...",
      "requested_by_user_id": 42,
      "expires_at": "..."
    }
  }
}
```

Wichtig: managed_state speichert NICHT die Live-qty. qty kommt immer aus `state['positions']`. Reconciliation-Job vergleicht beides periodisch.

3.3 risk_proposals/.json (NEU, eine Datei pro Proposal)

Versioniert (DR-12), eine Datei pro Proposal damit Audit-Trail durable nachvollziehbar bleibt.

```
{
  "proposal_id": "uuid-...",
  "asset": "SOL",
  "proposal_version": 1,
  "risk_model_version": "phase1-cautious-v1",
  "generated_at": "2026-05-10T...",
  "generated_by": "bot_proposal_engine",
  "expires_at": "2026-05-17T...",

  "market_context": {
    "current_price": 145.32,
    "atr_14d": 8.20,
    "volatility_30d_pct": 12.4,
    "volume_24h_usdt": 1234567890,
    "spread_pct": 0.05,
    "liquidity_score": 0.85,
    "regime": "BEAR",
    "regime_confidence": 0.72,
    "support_level": 138.0,
    "resistance_level": 158.0
  },

  "strategy_recommendation": {
    "strategy_group": "t1_core",
    "rationale": "Liquide Mid-cap, Mean-Reversion-tauglich",
    "alternative_groups": ["t2_pump_dump"]
  },

  "risk_recommendation": {
    "synthetic_entry_method": "current_price",
    "synthetic_entry_value": 145.32,
    "stop_loss": 130.00,
    "stop_loss_pct": 10.5,
    "take_profit": 165.00,
    "take_profit_pct": 13.5,
    "trailing_stop_enabled": false,
    "rr_ratio": 1.29,
    "max_allocation_usdt": 500.0,
    "max_allocation_pct_of_portfolio": 5.0,
    "dca_recommended": false,
    "dca_settings": null
  },

  "context_warnings": {
    "is_blacklisted": false,
    "is_stablecoin_or_peg": false,
    "concentration_warning": false,
    "correlation_with_existing_positions": [
      {"symbol": "ETH/USDT", "rho": 0.83, "warning": false}
    ],
    "regime_compatibility": "ok",
    "circuit_breaker_state": "closed"
  },

  "confidence": {
```

```
    "overall_score": 0.65,
    "explanation": "Liquide, aber BEAR-Regime + niedrige RR — moderate Empfehlung"
  },

  "alternative_proposals": [
    {"variant": "conservative", "stop_loss": 120.0, "take_profit": 155.0,
"max_allocation_usdt": 250.0},
    {"variant": "aggressive",   "stop_loss": 138.0, "take_profit": 175.0,
"max_allocation_usdt": 750.0}
  ]
}
```

Versionierung: `risk_model_version` ist HARDCODED in der Engine. Jeder Bump = neuer String. Alte Proposals bleiben auswertbar weil Schema-Versioned.

3.4 GUI-DB Tabellen (NEU, Cache + GUI-Read)

managed_proposals: - `proposal_id` (uuid pk) - `asset` (varchar) - `state` (varchar enum) - `proposal_json` (json) — Inhalt der proposal-Datei für GUI-Read - `generated_at`, `expires_at`, `decided_at` - `requested_by_user_id`, `decided_by_user_id` - `decision` (approved/rejected/expired)

managed_assets_history: - `id`, `asset`, `state_from`, `state_to`, `actor` (user_id or 'bot'/'system') - `event_type`, `ts`, `metadata_json`

Source of truth: `managed_state.json` + `risk_proposals/*.json` (Bot-Side). DB ist Read-Cache. GUI liest aus DB für Performance, Bot schreibt in beide.

3.5 Audit-Events (16 neu)

```
managed.asset_proposal_requested      (Operator triggert Analyse)
managed.asset_proposal_generated      (Bot legt Proposal-Datei an)
managed.asset_proposal_aborted        (Bot kann nicht generieren — kein Balance / Markt
down / blacklisted)
managed.asset_proposal_expired        (TTL hit)
managed.asset_proposal_rejected       (Operator declined)
managed.asset_promoted                (Operator confirmed → managed_active)
managed.asset_paused                  (Operator pausiert)
managed.asset_resumed                 (Operator resumes)
managed.asset_release_requested       (Operator → release_pending)
managed.asset_released                (back to frozen, terminal)
managed.synthetic_entry_set           (Initial-Setting beim promote)
managed.synthetic_entry_adjusted      (Operator override)
managed.drift_detected                (Bot detects qty/price/regime drift)
managed.operator_override             (Operator setzt Wert manuell, ignoriert Bot-
Vorschlag)
managed.exit_executed                 (SL/TP hit)
managed.policy_change_blocked         (DR-11: Auto-Demote-Versuch wurde geblockt)
```

---

4 · Scope-Liste

4.1 Komponenten neu (Bot-Side, Python)

- `trading/managed_state_writer.py` — atomic apply/clear (analog BaselineHoldingsWriter)
- `trading/managed_state_reader.py` — pure SELECT
- `trading/proposal_engine.py` — generates risk_proposals
- `trading/proposal_writer.py` — atomic write `risk_proposals/<id>.json`
- `trading/proposal_reader.py` — read

4.2 Komponenten erweitert (Bot-Side)

- `trading/baseline_bootstrap.py` — auto_import liest managed_state.json statt baseline.json:policy=managed
- `trading/scanner/universe.py` — filter berücksichtigt managed_state
- `trading/execution/paper_trade.py` + `live_trade.py` — `_is_base_unmanaged` statt `_is_base_baseline_frozen`
- `trading/execution/balance_provider.py` — managed-asset qty subtract evaluieren (offen)
- `trading/main.py` — drift-detection per-cycle hook
- `trading/command_worker.py` — +7 Handler

4.3 Komponenten neu (PHP-Side)

- `app/Services/Managed/ProposalService.php` — request, approve, reject
- `app/Services/Managed/ManagedStateService.php` — pause/resume/release
- `app/Services/Managed/ProposalSchemaRegistry.php` — schema versioning
- `app/Models/ManagedProposal.php`, `app/Models/ManagedAssetHistory.php`
- `app/Filament/Resources/ManagedProposalResource.php` — Liste + Detail
- `app/Filament/Pages/ManagedHoldings.php` — Live-Lifecycle-Tabelle
- 7 neue Einträge in `CommandTypeRegistry`:
- `request_managed_proposal`
- `approve_managed_proposal`
- `reject_managed_proposal`
- `pause_managed_asset`
- `resume_managed_asset`
- `release_managed_asset`
- `flag_managed_drift` (bot-self-emitted via internal channel)

4.4 Neue Policies

- `app/Policies/ManagedProposalPolicy.php` — admin-only für approve/reject
- Operator-Role bekommt: view + request_proposal
- Admin-Role bekommt: alles

4.5 Neue Stop-Regeln (5)

- **SR-11** proposal_engine ohne `risk_model_version` → STOP, abort proposal (DR-12)
- **SR-12** Promote-Versuch ohne approve-Audit → STOP (DR-11 enforcement)

- **SR-13** managed_state.json ohne Backup-Path → STOP (DR-5)
- **SR-14** Bot recommended t3_copy_trading → STOP (T-SPLIT durable)
- **SR-15** drift_detection autonomous-sell-Versuch → STOP (Bot ist Berater bei Drift)

---

5 · Phasen-Roadmap (RECON-MH-0..9)

Phase	Inhalt	Bot-Restart	Mainnet	Reversibel
MH-0 *(jetzt)*	Architektur-Closure: dieses Doc + Q-MH Antworten	nein	nein	trivial
MH-1	PHP-Schema + CommandTypeRegistry-Erweiterung um 7 Types + ~25 Tests	nein	nein	git revert
MH-2	Bot-Side managed_state_reader + proposal_reader (dormant) + ~25 Tests	nein	nein	git revert
MH-3	proposal_engine.py + ~30 Tests (standalone, kein CommandBus)	nein	nein	git revert
MH-4	PHP ProposalService + ManagedStateService + ~30 Tests	nein	nein	git revert
MH-5	Filament UI ManagedHoldings Page + ProposalResource + ~25 Tests	nein	nein	git revert
MH-6	Bot-Worker 7 Handler + ~30 Tests	nein	nein	git revert
MH-7	Bot-Side Wiring (bootstrap/universe/execute_buy) + drift-hook + ~25 Tests	**JA** *(separater RESTART-Sub-Step)*	nein	docker cp + restart-revert
MH-8	Testnet-Drill: 1 frozen-only baseline + 1 managed-promote-flow E2E	indirekt	nein	clear command
MH-9	Worker-Daemon-Aktivierung + scheduled jobs + reconciliation + Mainnet-Sign-Off-Audit	**JA**	bleibt blockiert	docker compose down

Vorbedingung MH-9: Q-MH-14 muss „ja, Daemon required" entschieden sein.

Vor jeder Phase: Backup-Pflicht, User-Approve, sha256-Verify post-cp, Cycle-Verify post-restart.

---

6 · Test-Strategie

6.1 Bot-Tests (~145 total geschätzt)

- managed_state_writer apply/clear/no_effect/idempotency (15)
- proposal_engine schema-conform + versioning + edge cases (30)
- bootstrap_baseline reads managed_state correctly (10)

- universe + execute_buy guards respect managed_state (15)
- drift detection (15)
- DR-11 enforcement (Bot-Side actor must be user) (10)
- 7 worker handlers each (35)
- E2E drill scenarios (15)

6.2 PHP-Tests (~120 total geschätzt)

- ProposalService (25)
- ManagedStateService (20)
- Registry-Validators (20)
- Filament Page admin-only + audit-trail (15)
- approve idempotent + race-resolution (15)
- expire-proposals job (10)
- DR-11 enforcement (PHP-Side audit actor check) (10)
- E2E coverage (5)

6.3 Source-grep Boundary-Tests

- `managed.*` events nicht in `baseline.*` / `runtime_config.*` Code-Pfaden
- DR-13 strict separation gepinnt
- proposal_engine schreibt nicht managed_state direkt (nur via CommandBus)

---

7 · Empfehlungen + nächster Schritt

7.1 Reading-Order für Operator-Review

1. **§1.1 + §1.4 + §1.7** — State-Machine + Bot-vs-Operator + DR-Konflikte. Wenn DR-7 (wallet_signature-Konflikt M2) nicht gelöst ist, gibt's keinen sicheren Promote.
2. **§2 Q-MH-1..15** — entscheiden VOR Code-Phasen. Insbesondere Q-MH-14 (Worker-Daemon required?) und Q-MH-15 (managed_state in DB oder JSON) sind architektur-prägend.
3. **§3 Datenmodell** — JSON-Shape + DB-Tabellen. Wenn das so passt, kann MH-1..3 starten.
4. **§5 Roadmap** — sind 9 Sub-Phasen vertretbar? Oder zusammenfassen?

7.2 Nicht starten mit MH-1 bevor:

- Q-MH-14 entschieden (Worker-Daemon)
- DR-7-Konflikt gelöst (M2 — wallet_signature bei policy-Wechsel)
- T-SPLIT-3 t3-Allowlist-Regel für proposal_engine gepinnt
- CAP-1 investable_capital interaction-Pattern entschieden

7.3 Reihenfolge (durable nach Operator-Approve)

1. RECON-MH-0 (dieses Doc) → Q-MH-Antworten → DR-7-Lösung

2. RECON-2.3-DEPLOY (klein, schon im Pin priorisiert)
3. RECON-2.4 Frozen-only Drill (NUR frozen, kein managed-import)
4. RECON-MH-1..7 (PHP + Bot Code-Phasen)
5. RECON-MH-8 Testnet-Drill mit erstem managed-Promote
6. RECON-MH-9 Hardening + Worker-Daemon
7. RECON-2.5 Mainnet-Readiness Review

---

8 · Cross-Reference

Doc	Inhalt
`mainnet_preflight_recon.md` (Repo)	Originaler RECON-Phasen-Plan + Q-1..Q-12
`recon_status_pin.md` (Memory)	Konsolidierte RECON-Übersicht + DR-1..DR-13 + SR-1..SR-10
`recon_2_1_status.md` (Memory)	RECON-2.1 Schema-Scaffolding
`recon_2_2a_status.md` (Memory)	RECON-2.2a Bot-Awareness wiring
`recon_2_2b_status.md` (Memory)	RECON-2.2b Restart-Drill
`recon_2_3_status.md` (Memory)	RECON-2.3 PHP-Side
`recon_2_3c_status.md` (Memory)	RECON-2.3c Bot-Worker handlers
`g10_runtime_config_operator_runbook.md` (Repo)	G10 Apply/Clear Pattern als Template
`t_split_status.md` (Memory)	strategy_group durable rule, t3 disabled

---

9 · Boundaries gehalten

- **kein Code geschrieben** außerhalb dieses Doks
- **keine Migration**
- **kein Refactor**
- **kein Bot-Restart**
- **kein Worker-Start**
- **kein Mainnet**
- **kein Push**

Output dieses Doks: reine Architektur-Phase. Nächster Schritt = Operator-Review + Q-MH-Antworten.

— Ende RECON-Managed-Holdings Architektur-Bericht (Stand 2026-05-10) —