DATA-CLEANUP-1 Plan-Review

2026-05-14 UTC · master `7f65097` · **Plan-Phase** **5 Sub-Phasen identifiziert** **Code-Recon vollständig**

> **Zweck**: Datenverträge zwischen Bot, DB und (zukünftiger) GUI reparieren, BEVOR GUI-DESIGN-1b/1c live geht. Der Bot läuft jetzt nach TELE-1b-B wieder, aber: position_snapshots haben kein Close-Lifecycle, trade_logs.decision_id ist immer NULL, exit_reason ist Sammellabel. GUI würde diese Defekte sichtbar machen.

1. Boundary-Snapshot

Item	Wert
master HEAD	`7f65097`
git status	clean
Bot main.py PID (in-container)	199
Bot Container Host-PID	3082815, healthy, restartCount=0
Worker Host-PID	2658058 (unverändert)
GUI Host-PID	2656633 (unverändert)
BINANCE_TESTNET	true
cmd 13 / mp / history	cancelled / 0 / 0
regime_latest	BEAR
Worker heartbeat age	1 s
0 Tracebacks seit 13:31	bestätigt
decision_logs latest	14:06:49 (age 69s)
position_snapshots latest	14:06:33 (age 85s)
regime_logs latest	14:06:29 (age 89s)
trade_logs latest	2026-05-14 05:07:57 (kein neuer Trade — Bot generiert aktuell 0 BUY-Signale)
5 fresh Positionen post-13:31	$2,188 nominal / -$31 upnl
runtime_config / managed_state / baseline JSON	nicht im erwarteten Pfad (Reader/Writer-Module vorhanden, aber keine aktiven JSON-Files persistiert)

2. Code-Recon-Highlights

2.1 emit_position_snapshot

Befund	Detail
Signature	`db_emitter.py:290` — `status: str # REQUIRED — open|closing|closed|error`
Aufrufer im Bot-Code	nur **1 Stelle**: `trading/main.py:318` (Snapshot beim Scan-Loop, immer mit `status='open'`)
Close-Aufrufer	**0** — kein `emit_position_snapshot(status='closed', ...)` existiert
Folge	position_snapshots akkumuliert auf 15173 rows, alle `status='open'`

2.2 Close-Pfade (paper_trade.py + live_trade.py)

```
paper_trade.py:515-540  → trade_record = {'reason': reason, 'entry_time': ..., 'exit_time': ...,
                            'strategy_group': pos['strategy_group']}
                        → state['closed_trades'].append(trade_record)
                        → del state['positions'][symbol]
                        → SL/TP cooldown setzen

live_trade.py:759-961  → analoge Struktur

KEIN emit_position_snapshot(status='closed', position_id=pid, ...) wird hier aufgerufen.
```

2.3 emit_trade-Call

```
trading/main.py:281-300:
    emit_trade(
        trade_id=...,
        position_id=ensure_position_id(t, sym_t),
        symbol=...,
        status='closed',
        entry_price=..., exit_price=..., quantity=...,
        realized_pnl=..., realized_pnl_pct=...,
        exit_reason=t.get('reason'),       ← reason direkt aus close
        strategy_group=t.get('strategy_group'),
    )
    # KEIN decision_id=... !
```

→ Bestätigt: `trade_logs.decision_id` ist 100% NULL weil emit_trade-Aufruf den Parameter nicht übergibt.

2.4 decision_log_id Datenpfad

```
main.py:607  _decision_log_id = generate_decision_id()
main.py:611  emit_decision(decision_id=_decision_log_id, ...)      ← decision_logs.decision_id korrekt
main.py:633  signal = {..., 'decision_log_id': _decision_log_id}   ← Signal-dict hat es
paper_trade.py: ???                                                ← geht's ins pos-dict?
paper_trade.py:520 trade_record = {... reason ... entry_time ...}  ← decision_log_id FEHLT hier
main.py:281 emit_trade(...)                                        ← liest trade_record, daher kein decision_id
```

3. Scope-Pin gemäß Operator-Plan (A-E)

ID	Beschreibung	Status nach Recon
A — SNAPSHOT-LIFECYCLE-1	Beim Positions-Close NEUEN Snapshot mit `status='closed'` emiten. Historie nicht mutieren. position_snapshots bleiben rein historische Events. Aktuelle offene Positionen NIE aus DISTINCT-ON-open Query ableiten.	Fix-Pfad klar: in paper_trade.py + live_trade.py beim Close-Append einen `emit_position_snapshot(status='closed', position_id=...)` -Call ergänzen. Pro File ~5 Zeilen.
B — OPEN-POSITIONS-DATA-CONTRACT-1	Verlässliche Methode für „aktuelle offene Positionen". Optionen 1-3 (FRESH-Filter / paper-state-SoT / read-only View).	**Operator-Decision nötig**. Empfehlung: Option 1 (FRESH-Filter) als Kurz-Lösung + Option 3 (SQL-View) als Mittel-Frist + Option 2 (paper-state) für GUI-SoT nach Code-Baking (IMAGE-2).
C — DATA-LINK-1	trade_logs.decision_id muss gesetzt werden. Korrelation Entscheidung→Trade.	Fix-Pfad: signal['decision_log_id'] muss bis ins trade_record propagieren. Anpassungen in paper_trade.py (Position-Open speichert decision_log_id in pos-dict; Close-Path nimmt es in trade_record auf) + main.py:281 emit_trade(decision_id=t.get('decision_log_id')).
D — LABEL-1	exit_reason differenzieren: hard_stop_loss, trailing_stop, break_even_stop, time_stop, take_profit, manual_exit, circuit_breaker_exit, unknown. Alte Daten als legacy_stop_loss markieren.	Fix-Pfad: an SL-Trigger-Stellen in paper_trade/live_trade die jeweilige reason-Variante setzen. Aktueller Code nutzt nur `reason in ('stop_loss','sl','take_profit','tp')` — Differenzierung muss inline an Trigger-Stellen rein. Historische Daten: einmaliges UPDATE auf `'stop_loss_legacy'`.
E — STRATEGY-INTERIM-REPORT-CORRECTION	Strategie-Audit mit korrigierter Datenbasis neu ausgeben.	Pure Read-only Re-Run nach D-Migration. Telemetry-Lücke 07:20–13:31 UTC explizit markieren. 5 echte offene Positionen statt 42. Profit Factor / Loss-Asymmetrie aus trade_logs bleibt valide.

4. Empfohlene Sub-Phasen-Reihenfolge

Reihenfolge	Phase	Typ	Aufwand
1	DATA-CLEANUP-1 Plan-Review (dieses Doc)	read-only	fertig
2	**SNAPSHOT-LIFECYCLE-1** Implementation (A)	Code-Edit + Tests	~3-4h
3	OPEN-POSITIONS-DATA-CONTRACT-1 Plan-Review (B)	Operator-Decision + read-only	~1-2h
4	OPEN-POSITIONS-DATA-CONTRACT-1 Implementation	SQL-View + Doc	~2-3h
5	**DATA-LINK-1** Implementation (C)	Code-Edit	~2-3h
6	**LABEL-1** Implementation (D)	Code-Edit + Migration	~3-4h
7	**STRATEGY-INTERIM-REPORT-CORRECTION** (E)	read-only Re-Audit + PDF	~1-2h
8	GUI-DESIGN-1b/1c starten (Live-Daten erlaubt nach 5+7)	—	—

Total: ~12-18h verteilt über mehrere Tage. Phasen können teilweise parallel (z.B. B-Plan-Review während A-Implementation).

5. Constraint-Matrix (Operator-Plan)

Constraint	Sub-Phase A	B	C	D	E
Bestehende Daten nicht blind mutieren	JA — nur neue Close-Snapshots	JA — read-only View	JA — nur neue Trades bekommen decision_id	partial — historische als legacy markieren ist OK	JA — read-only
Strategy-Logik unangetastet	JA	JA	JA	JA — nur Labeling	JA
BINANCE_TESTNET=true	JA	JA	JA	JA	JA
Keine DB-Migration	JA — Tabelle existiert, nur INSERT	JA — View ist DDL aber non-mutating	JA — nur INSERT-Argument	NEIN — UPDATE für legacy-Markierung nötig (P2)	JA
Kein Bot-Restart	NEIN — Code-Change → Recreate	JA	NEIN — Recreate	NEIN — Recreate	JA

6. GUI-Block-Compliance

> **GUI-DESIGN-1b/1c BLOCKIERT** bis Sub-Phasen A+C+E abgeschlossen. Begründung:
> - GUI würde `position_snapshots` direkt anzeigen → ohne A würden 38 stale Symbols als „offen" erscheinen
> - GUI „Trade-Detail"-View braucht decision_id-Link für Score/Source/Regime-Anzeige → ohne C nur teilweise nutzbar
> - GUI „Verlustanalyse" braucht differenzierte exit_reasons → ohne D nur als „stop_loss"-Klumpen

GUI-DESIGN-1a (Theme/Styling only, keine Live-Daten-Widgets) **darf parallel laufen**.

7. Operator-Decisions (vor Implementierung)

1. **D1 — OPEN-POSITIONS-DATA-CONTRACT-1 Variante**:
   - (a) FRESH-Filter (created_at > letzter Bot-Restart-Marker) — schnell, aber sensitiv für Restarts
   - (b) paper/wallet state als SoT — saubest, aber abhängig von IMAGE-2 Code-Baking
   - (c) SQL-View `v_current_open_positions` — sauber, performant, einfache GUI-Konsumption
   - Empfehlung: **(c) + (a) als fallback während Phase 2-4**, später (b) wenn Code gebaked ist
2. **D2 — LABEL-1 Migration-Strategy**:
   - (a) historische 47 stop_loss-Rows in trade_logs als `stop_loss_legacy` markieren
   - (b) historische Daten unverändert lassen, nur neue Trades differenziert labeln
   - Empfehlung: (a) — Operator-Audit-Hygiene
3. **D3 — Reihenfolge A oder C zuerst?**
   - A liefert größeren GUI-Benefit (Position-Anzeige korrekt) sofort
   - C ist Voraussetzung für E (Strategy-Audit-Korrektur)
   - Empfehlung: **A vor C** — größtes Risiko-Bild zuerst

8. GO/NO-GO

GO Plan-Sequenz mit Operator-Decisions D1-D3. Erstes Implementation-GO empfohlen für **SNAPSHOT-LIFECYCLE-1** (A) als P1-Sofort, weil:

- kleinster Diff (~5 Zeilen × 2 Files)
- direkter GUI-Block-Lift
- kein DB-Schema-Change (Tabelle existiert)
- kein Strategie-Code-Touch

9. Boundaries — eingehalten

0× Code-Touch · 0× DB-Write · 0× Restart · 0× docker cp · 0× Mainnet · 0× Orders · 0× Positionsänderung · 0× runtime_config/managed_state/baseline-Mutation · 0× Secret-Output · 0× env-Dumps · 0× `compose config` · 0× `/proc` bulk.