DATA-LINK-1 Plan-Review — trade_logs ↔ decision_logs Wiring

Datum: 2026-05-15 09:42 UTC | Repo: `Steve-TradingBot` | master HEAD: `f4075f8` (GUI-DESIGN-1a-FU1) | Stack: Python 3 Bot + PostgreSQL gui-db | Modus: **EXCHANGE_TESTNET** | Scope: READ-ONLY | Impl: **NO-GO (D1–D4 nötig)**

Executive Summary

Root-Cause (ein Satz): `_decision_log_id` wird in `main.py` korrekt erzeugt und in `decision_logs` geschrieben, aber **nicht an `execute_buy()` weitergereicht** — daher fehlt das Feld auf der Position, im trade_record und schließlich in `trade_logs.decision_id`. Identisch fehlen `opened_at` und `strategy_id`.

Lösung: **Fix-Pattern existiert bereits im Repo.** Die T-SPLIT-2-Phase hat `strategy_group` 1:1 entlang derselben Kette durchgepropagiert (`main.py` → `execute_buy()` → `_new_pos` → `execute_sell()` → `trade_record` → `emit_trade()`). **Wir wenden dasselbe Muster für `decision_id` + `opened_at` + `strategy_id` an.**

Keine DB-Migration nötig: `trade_logs.decision_id`, `opened_at`, `strategy_id` Spalten existieren bereits (alle `varchar(255)` bzw. `timestamptz` / `bigint`). Historische Daten werden **nicht** rückwirkend aktualisiert (Operator-Pin).

GO/NO-GO: GO für Implementation. Kleiner Refactor, ca. 30 LoC produktiv + 30 LoC Tests, geschätzt ~1,5–2 h. Risiko niedrig: pures Metadata-Forwarding, keine Trading-Logik berührt.

Warum jetzt? — Motivation

> *Aus dem Strategy Interim Report (PF 0.548, Loss-Asymmetrie 2.38x): Regime-Attribution und Trail-Diagnose sind **nicht belastbar**, solange `trade_logs.decision_id` zu 100 % NULL ist. Die BEAR-Regime-Auswertung ("BEAR ist Hauptverlustquelle"-Verdacht) lässt sich erst nach DATA-LINK-1 sauber prüfen. **DATA-LINK-1 ist die letzte harte Voraussetzung für TRAIL-1 + OHLCV-Backtest.***

1. Live-State (Preflight)

Item	Wert	Bewertung
master HEAD	`f4075f8` (GUI-DESIGN-1a-FU1)	—
`decision_logs` total / mit `decision_id`	**168.107 / 1.726**	1 %
`decision_logs` action `buy` / davon mit ID	**1.020 / 1.020**	✓ **BUY-Pfad funktioniert**
`decision_logs` `evaluating` / `reject` / davon mit ID	7.064 / 6.044 / **0**	nicht-actionable, by-design ohne ID
`trade_logs` total closed	**56**	—
`trade_logs` mit `decision_id`	**0**	✗ **100 % NULL**
`trade_logs` mit `opened_at`	**0**	✗
`trade_logs` mit `strategy_id`	**0**	✗
`trade_logs` mit `position_id`	**56**	✓ **T-SPLIT-2 + N8.2 funktionieren**

2. Wo bricht die Kette? — Pfad-Diagramm

decision_id-Pfad

```
generate_decision_id()
   ↓ (main.py:647)
_decision_log_id    [Python local variable]
   ├── emit_decision(action='buy', decision_id=_decision_log_id, …)
   │      ↓ (main.py:649)
   │   decision_logs.decision_id    ✓ in DB
   │
   ├── signal = {'decision_log_id': _decision_log_id, …}
   │      ↓ (main.py:672)
   │   signal-Dict    ✓ in memory
   │
   └── live_trader.execute_buy(symbol, cp, qty, sl, tp, allow_add=…)
          ↓ (main.py:714 / 726 / 750 / 862 / 971 / 1157 — 6 Call-Sites)
       ✗ kein `decision_id`-Parameter durchgereicht
          ↓
       execute_buy(…) → _new_pos {position_id, symbol, qty, entry_price, …}
       ✗ kein `_new_pos['decision_id']` Slot
          ↓
       …  (Position lebt, Trailing/Breakeven/News-Adjustments laufen)
          ↓
       execute_sell(...) → trade_record {position_id, pnl, exit_reason, …}
       ✗ kein `trade_record['decision_id']` Feld
          ↓
       emit_trade(trade_id=…, position_id=_pid_t, …)
       ✗ kein `decision_id=` Kwarg (main.py:286)
          ↓
```

```
trade_logs.decision_id = NULL     ✖ KETTE GEBROCHEN
```

opened_at + strategy_id

- `opened_at` : `pos['entry_time']` existiert als ISO-String, wird aber nicht via `execute_sell→trade_record→emit_trade` durchgereicht.
- `strategy_id` : vom signal/candidate stammend (`signal['strategy_id']` oder `candidate['strategy']`), wird in der gesamten Kette nicht durchgereicht. **NICHT zu verwechseln** mit `strategy_group` (T-SPLIT-2: t1_core/t2_pump_dump/t3_copy_trading/legacy_unknown) — `strategy_id` ist die spezifische Strategie-Bezeichnung (z. B. `rsi_breakout` , `pattern_recognition`).

3. Felder im Schema — keine Migration nötig

Feld	Typ	Status
`trade_logs.decision_id`	`varchar(255)`	**vorhanden** ✓
`trade_logs.opened_at`	`timestamp(0) with time zone`	**vorhanden** ✓
`trade_logs.strategy_id`	`varchar(255)`	**vorhanden** ✓
`trade_logs.duration_seconds`	`bigint`	**vorhanden** ✓ (kann derived sein)
`trade_logs.position_id`	`varchar(255)`	vorhanden ✓, befüllt
`decision_logs.decision_id`	`varchar(255)`	vorhanden ✓, 1.726 Werte

→ **0 DB-Migration**. Keine ALTER TABLE.

4. Fix-Patch-Pattern (Vorbild T-SPLIT-2)

Beispiel des bestehenden T-SPLIT-2-Patterns für `strategy_group`

```
// live_trade.py:493
def execute_buy(self, symbol, price, quantity,
                stop_loss, take_profit,
                allow_add=False,
                strategy_group: Optional[str] = None):  # T-SPLIT-2
    ...
    _new_pos = {'position_id': generate_position_id(symbol), ...}
    if strategy_group is not None:                      # T-SPLIT-2
        _new_pos['strategy_group'] = strategy_group      # T-SPLIT-2

// live_trade.py:execute_sell
trade_record = {'position_id': _pid, ...}
if 'strategy_group' in pos:                             # T-SPLIT-2
    trade_record['strategy_group'] = pos['strategy_group']  # T-SPLIT-2

// main.py:286 emit_trade
emit_trade(
    ...
    strategy_group=t.get('strategy_group'),             # T-SPLIT-2
)
```

Was wir 1:1 reproduzieren für decision_id / opened_at / strategy_id

Datei	Änderung	Est. LoC
`execution/paper_trade.py:316`	Signatur `execute_buy(…, strategy_group=None, decision_id=None, strategy_id=None)`	+2
`execution/paper_trade.py:458–475` `_new_pos`	`if decision_id is not None: _new_pos['decision_id'] = decision_id` · analog `strategy_id`	+4
`execution/paper_trade.py:execute_sell` trade_record	`trade_record['decision_id'] = pos.get('decision_id')` · `['strategy_id']` · `['opened_at'] = pos.get('entry_time')`	+6
`execution/live_trade.py:493–496`	analog Signatur-Erweiterung	+2
`execution/live_trade.py:618–637` `_new_pos`	analog	+4
`execution/live_trade.py:execute_sell` trade_record	analog	+6
`main.py:286` `emit_trade(…)` Aufruf	`decision_id=t.get('decision_id'), opened_at=t.get('opened_at'), strategy_id=t.get('strategy_id')`	+3
`main.py:714/726/750/862/971/1157` 6× `execute_buy(…)`	+ `decision_id=signal.get('decision_log_id'), strategy_id=signal.get('strategy_id')`	+6
`db_emitter.py:emit_trade`	Parameter bereits akzeptiert; ggf. ISO-String → datetime-Cast prüfen	0

Gesamt: ~30 LoC produktiv + ~30 LoC Tests, kein Schema-Change.

5. Bedenken / Sonderfälle

DCA-Rescue + Pyramid (`allow_add=True`)

`main.py:714/726` rufen `execute_buy(..., allow_add=True)` für **DCA-Rescue + Pyramid-Add** auf. Hier wird zur **bestehenden** Position addiert, nicht neue erzeugt. → Frage **D1**: Soll die `decision_id` der **ursprünglichen** BUY-Decision behalten werden ODER die der DCA-Decision genommen?

Empfehlung: ursprüngliche behalten (T-SPLIT-2 macht das analog für strategy_group: nur bei neuem `_new_pos` wird gesetzt; bei DCA bleibt der existierende Wert).

```
if symbol in self.state['positions'] and allow_add:
    # DCA/Pyramid: decision_id/strategy_id der ursprünglichen Position lassen
    pass
else:
    # Neue Position: decision_id/strategy_id setzen
    if decision_id is not None:
        _new_pos['decision_id'] = decision_id
```

strategy_id-Quelle

Manche Pfade haben nur `strategy_group` (T-SPLIT-2: `t1_core` etc.) — die spezifische `strategy_id` ("rsi_breakout", "pattern_recognition") wird in der Signal-Pipeline **nicht überall** propagiert. → Frage **D3**: Fallback wenn Quelle fehlt?

Empfehlung: strategy_id wenn vorhanden, sonst NULL (kein Mock-Wert).

opened_at vs. closed_at vs. duration_seconds

`pos['entry_time']` ist ISO-String (`datetime.now(timezone.utc).isoformat()`). `db_emitter.emit_trade` akzeptiert `opened_at: Optional[datetime] = None` — Cast ISO-String → datetime ist eine Frage des Wrapper-Codes (vermutlich bereits in db_emitter, sonst in main.py vor dem Aufruf).

`duration_seconds` kann direkt im Bot (`closed_at - opened_at`) gesetzt oder in einer downstream-Aggregation berechnet werden. Empfehlung: **im Bot berechnen** und mitgeben, vermeidet downstream-Joins.

Historische Daten

Frage	Antwort
Alte 56 trade_logs rückwirkend updaten?	**Nein** (Operator-Pin "keine historischen DB-Mutations")
Alte position_snapshots-rows?	unverändert
Alte decision_logs (1.020 BUY mit ID, 0 actionable Links zu trades)?	unverändert
Funktionaler Verify nach Fix	Beim **nächsten** echten BUY → Close-Event: neuer `trade_logs.decision_id` wird befüllt

6. Testplan

#	Typ	Inhalt
T1	Unit	`execute_buy(decision_id='abc', strategy_id='rsi')` → position-dict hat `decision_id='abc'` + `strategy_id='rsi'`
T2	Unit	DCA-Rescue (`allow_add=True`) lässt bestehende `decision_id` unverändert
T3	Unit	`execute_sell` → trade_record hat `decision_id` , `opened_at` , `strategy_id` aus pos kopiert
T4	Unit	`emit_trade` -Aufruf mit den 3 Feldern wird korrekt aufgerufen (Mock-Spy auf db_emitter)
T5	Integration	Bot-Spawn → Scan → BUY → Close → SQL-Check `trade_logs.decision_id IS NOT NULL`
T6	Live-Verify	JOIN-Query `trade_logs t JOIN decision_logs d ON d.decision_id = t.decision_id WHERE t.closed_at > <cutover_ts>` liefert Rows

7. Implementation-Phasen

Phase	Inhalt	Dauer
DATA-LINK-1-impl	Code-Patch (~30 LoC + Tests), Image-Rebuild, Container-Recreate via SOT-1d-Pfad	~2 h
DATA-LINK-1-MONITOR	~24–72 h Daten sammeln — erste new trade_logs mit `decision_id` beobachten	passive Wartezeit
DATA-LINK-1-VERIFY	JOIN-Test gegen `decision_logs` , BEAR-Regime-Attribution-Test	nach Monitor

Danach: **TRAIL-1** + **OHLCV-Backtest** mit endlich vollständiger Attribution.

8. Operator-Decisions D1–D4

ID	Frage	Empfehlung
D1	DCA-Rescue: `decision_id` der ursprünglichen BUY behalten ODER der DCA-Decision setzen?	**Ursprüngliche behalten** (analog T-SPLIT-2 strategy_group-Pattern)
D2	`opened_at` zusätzlich befüllen (für duration_seconds) ODER nur decision_id?	**Beides** (kostet 0 Extra-Code, gibt Hold-Time-Profil)
D3	`strategy_id`-Quelle: aus signal/candidate kommen — Fallback wenn fehlt?	strategy_id wenn vorhanden, sonst NULL (kein Mock-Wert)
D4	Filament Visual-Mapping: TradeLogResource zeigt `decision_id` als Klick-Link zu DecisionLogResource?	**Backlog GUI-DESIGN-1c** (post-DATA-LINK-1-MONITOR)

9. Risiko-Matrix

Risiko	Schwere	Mitigation
Trading-Logik versehentlich geändert	**HOCH**	Patch ist Pure Metadata-Forwarding; T-SPLIT-2-Pattern als Vorbild bewährt; Tests T1–T4 prüfen Position-State + trade_record genau
DCA-Rescue verliert ursprüngliche `decision_id`	mittel	D1 — Pattern explizit "behalten bei allow_add=True"
`strategy_id` Quelle unklar in einigen Pfaden	niedrig	D3 — NULL-Fallback OK, kein Mock
ISO-String → datetime-Cast in `emit_trade` fehlt	niedrig	Pre-Impl-grep im `db_emitter.py`; falls fehlt: Cast in main.py vor Aufruf
Historische DB-Werte aus Versehen mit `UPDATE` berührt	niedrig	0 SQL-Statements im Patch; nur Code-Pfad-Erweiterung
Bot crash durch unerwartete `pos['decision_id']`-Persistenz	niedrig	Bot-Restart liest live_portfolio.json → `.get()` mit None-Fallback (analog strategy_group)
Filament GUI bricht durch neue Felder	niedrig	GUI-Felder bereits im Schema; TradeLogResource zeigt sie schon, nur als leere Zellen

10. Boundaries (Plan-Phase eingehalten)

Boundary	Status
Kein Code	✓
Keine DB-Migration	✓ (Schema passt)
Keine historischen DB-Mutations	✓
Keine Trading-Logik-Änderung	✓ (Patch ist Pure Metadata-Forwarding)
Keine künstlichen Trades	✓
Kein Mainnet	✓ BINANCE_TESTNET=true
Kein Push ohne separates GO	✓ master `f4075f8` unverändert
Keine Secrets	✓

11. GO / NO-GO

Plan-Review: DONE

Implementation DATA-LINK-1-impl: NO-GO sofort — Operator-Decisions D1–D4 nötig.

Nach D1–D4 wäre Impl niedriges Risiko (~30 LoC, kein Schema-Change, T-SPLIT-2-Pattern bewährt). SOT-1d-Pfad greift: `git commit` → `docker compose build clawbot && up -d --no-deps clawbot` ohne Volume-Tanz.

12. Erklärungen — was ist DATA-LINK-1 eigentlich?

Begriffe

Begriff	Bedeutung
`decision_id`	UUID-artige Kennung pro BUY-/SELL-Decision. Wird im Bot bei jedem actionable Trade-Signal generiert (`generate_decision_id()`) und in `decision_logs` geschrieben.
`position_id`	UUID-artige Kennung pro echte Open-Position. Wird in `execute_buy` erzeugt und wandert mit der Position bis zum Close.
`strategy_group`	Coarse-Bucket der Strategie: `t1_core` , `t2_pump_dump` , `t3_copy_trading` , `legacy_unknown` (T-SPLIT-2).
`strategy_id`	Specific-Strategy-Name innerhalb des Buckets: z. B. `rsi_breakout` , `pattern_recognition` .
`opened_at`	Timestamp wann die Position eröffnet wurde (ISO-String `entry_time` im Bot-State).
DATA-LINK-1	Die fehlende Verkettung zwischen einem konkreten BUY-Entry (`decision_logs.decision_id`) und dem daraus resultierenden Close-Trade (`trade_logs.decision_id`).

Was geht nach DATA-LINK-1 endlich? (heutige Blocker)

1. **Regime-Attribution** pro Trade: "Welcher Anteil meines PnL kam aus BEAR-Regime-Trades?" — bisher unmöglich, weil `trade_logs` keinen Join auf `decision_logs.regime` erlaubt.
2. **Strategy-Attribution**: "Welche Strategie macht den größten Loss-Beitrag?" — heute nur über grobe `strategy_group` möglich.
3. **Hold-Time-Profile**: "Wie lange leben Wins vs. Losses im Schnitt?" — heute keine `opened_at` in trade_logs.
4. **Decision-Quality-Audit**: "Wie viele 'buy'-Decisions enden tatsächlich profitabel? Welche regime-/score-Kombination ist die beste?"
5. **OHLCV-Backtest mit Attribution**: Strategy-Backtest gegen historische Daten mit korrekter Regime/Strategy-Zuordnung.

Warum DATA-LINK-1 erst jetzt? — Historische Lage

Das Schema-Feld `trade_logs.decision_id` existiert seit dem ursprünglichen Schema-Entwurf, wurde aber nie befüllt. T-SPLIT-2 (2026-05-09) hat das identische Pattern für `strategy_group` bereits implementiert — DATA-LINK-1 erweitert dieses Pattern um 3 weitere Felder. Die Reihenfolge ist sinnvoll: erst **POS-CAP-1** (False-Alarm geklärt), dann **TELE-1b-B** (psycopg2/db_emitter geheilt), dann **DATA-CLEANUP-1** (closed snapshots), dann **LABEL-1** (exit_reason Subtypes) — jetzt **DATA-LINK-1** ist die letzte Wiring-Lücke vor TRAIL-1.

13. Sequenz

Phase	Voraussetzung	Output
DATA-LINK-1-impl	D1–D4 Operator-GO	~30 LoC, Image-Rebuild, Recreate
DATA-LINK-1-MONITOR	~24–72 h passive	Neue trade_logs mit `decision_id`
DATA-LINK-1-VERIFY	JOIN-Query liefert Rows	Regime-Attribution belastbar
TRAIL-1 / OHLCV-Backtest	nach VERIFY + LABEL-1-Monitor ~3–5 d	Strategy-Diagnose, Parameter-Tuning
Operator-Decision Parameter-Tuning	nach TRAIL-1	Profit-Factor-Verbesserung

STOP — Operator-GO für D1–D4 + Impl-Start erforderlich. Background-Agent (9 Widgets) läuft parallel.