

PLAN_SNAPSHOT_EMIT_COMPLETENESS

Status: planned (backlog, awaits operator GO) **Priority:** P2 **Erstellt:** 2026-05-17 **Quelle:** GUI-VIEW-POSITION-POLISH-1 Befund — Detail-Page-Felder leer **Roadmap-ID:** SNAPSHOT-EMIT-COMPLETENESS **Voraussetzung:** T1-QUALITY-SCORE-SHADOW-1 (C1) durch, 24-48h Beobachtung gesammelt.

1. Motivation

Auf `/admin/positions/{id}` (View Position) sind diverse Identifier- / Lifecycle-Felder leer, obwohl die GUI-Resource sie als `TextEntry` rendert:

```
strategy_id      : -
profile_id       : -
decision_id      : -
opened_at        : -
duration_seconds : -
risk_reward      : -
metadata_json    : (leer)
```

Root cause: der per-scan-cycle Aufruf von `emit_position_snapshot()` in `trading/main.py:448` überträgt nur die volatilen Felder (`current_price`, `unrealized_pnl`, `position_value`, ...). Die statischen Identifier-Felder, die der initial-BUY-Emit hat, gehen auf dem per-scan-tick verloren.

Operator sieht im Adminpanel deshalb eine ausgedünnte View.

GUI hat in GUI-VIEW-POSITION-POLISH-1 bereits die Anzeige-Seite aufgeräumt (Sections, Placeholder `—`, Decision-Log-Link). Die Felder bleiben aber leer bis der Bot sie auch beim per-scan-emit mitschickt.

2. Scope (was geändert wird)

Eine einzelne Code-Stelle: `trading/main.py:448` Aufruf von `emit_position_snapshot()` in der Loop über `live_trader.state['positions']`.

Erweitern um folgende Felder aus dem `_pos`-Dict:

Snapshot-Feld	Quelle im <code>_pos</code>	Notiz
<code>decision_id</code>	<code>_pos.get('decision_id')</code>	DATA-LINK-1 hat das Feld bereits in <code>_pos</code>
<code>opened_at</code>	<code>_pos.get('entry_time')</code>	ISO-String, vom Bot beim Buy gesetzt
<code>strategy_id</code>	<code>_pos.get('strategy_id')</code>	wird bereits übergeben, oft <code>None</code> für scanner-buy
<code>profile_id</code>	<code>_pos.get('profile_id')</code>	derzeit <code>None</code> — bleibt <code>None</code> bis <code>ConfigProfile-Wiring</code>
<code>risk_reward</code>	<code>_pos.get('risk_reward')</code> oder berechnet aus <code>(tp-entry)/(entry-sl)</code>	wenn <code>None</code> : nicht setzen
<code>metadata</code>	dict mit Auszug aus <code>_pos</code> : <code>quality_score</code> , <code>_sl_kind</code> , <code>_tp_kind</code> , <code>_last_dca_entry_price</code> , <code>_labell_started</code> , <code>_binance_pair_check</code>	nur Schlüssel die existieren

`duration_seconds` wird **nicht** vom emit gesetzt — es ist semantisch ein closed-trade-Feld. Open-Positions bleiben dort leer. View-Page zeigt „—“.

3. Boundaries

- **Bot-Touch** ja (`main.py:448`). Container-Recreate notwendig.
- 0x Strategieparameter-Tuning
- 0x DB-Migration (alle benutzten Spalten existieren bereits)
- 0x DB-Mass-Mutation (historische Rows bleiben sparse — frische Snapshots ab Cutover füllen sich)
- 0x Mainnet
- 0x Worker-Recreate
- 0x CommandBus-Version-Bump
- 0x Push ohne separates GO

4. Cutover-Plan (SOT-1d)

1. Pre-cutover snapshot (HEAD, container PID, env flags).
2. Watchdog freeze `CUTOVER_FREEZE_SNAPSHOT_EMIT_COMPLETENESS`.
3. `docker compose build clawbot`.
4. Container-Test im neuen Image: `python3 -m unittest tests.test_snapshot_emit_completeness`
5. `docker compose up -d --force-recreate --no-deps clawbot`.
6. 3-Way MD5 Repo == Image == Container für `main.py`.
7. Bot spawn + healthcheck + 0 Tracebacks.
8. Live-Verify nach erstem Scan-Cycle: `sql SELECT decision_id, strategy_id, opened_at, metadata_json FROM position_snapshots WHERE id IN (SELECT DISTINCT ON (position_id) id FROM position_snapshots WHERE status='open' AND created_at > NOW() - INTERVAL '5 minutes' ORDER BY position_id, created_at DESC);`
Erwartet: für mind. eine neue Position sind die Felder gesetzt (alte Positionen bleiben sparse, weil die initial-emit-Daten nicht im `_pos`-Dict gespeichert wurden — die `_pos`-State-Datei muss abwarten bis bot beim nächsten BUY frische Felder produziert).
9. Watchdog re-enable.
10. Roadmap-Update `SNAPSHOT-EMIT-COMPLETENESS` → done.

5. Tests (Plan)

`trading/tests/test_snapshot_emit_completeness.py` (neu):

- `emit_position_snapshot` Mock prüft dass `decision_id / opened_at / strategy_id` aus `_pos` durchgereicht werden.
- `metadata`-Block enthält nur die Schlüssel die in `_pos` existieren (kein None-Padding).
- `risk_reward` Fallback-Berechnung wenn `_pos.get('risk_reward')` None und SL/TP/Entry vorhanden: `(tp - entry) / (entry - sl)`.
- Backwards-compatible: Position-Dicts ohne Identifier-Felder lassen die Snapshot-Felder None — kein Crash.

6. State-Migration für bestehende Positionen

Die 6 aktuell offenen Positionen wurden vor diesem Plan geöffnet. Ihr in-memory `_pos`-Dict hat keine `decision_id / opened_at`-Felder. Nach Cutover bleiben deren Snapshots weiterhin sparse. Erst ein neuer BUY produziert ein vollständig ausgefülltes `_pos`.

Optional follow-up (nicht in dieser Phase):

- `_pos`-Backfill aus DB: beim Bot-Boot pro offener Position einmalig `decision_id / opened_at` aus `trade_logs / decision_logs` ziehen und ins `_pos` schreiben. Eigener Plan, falls Operator das will.

7. Erwartete Lieferung

- `trading/main.py` — Diff < 30 Zeilen.
- Tests grün (100/100 nach Cutover + Phase-D + neue Tests).
- Closure-Form analog Phase D.

Commit-Vorschlag: `snapshot-emit-completeness: thread decision_id / opened_at / metadata into per-scan snapshot`

8. STOP

Kein Code vor Operator- `G0 SNAPSHOT-EMIT-COMPLETENESS`.

Bis dahin: * Detail-Page zeigt — Placeholder für leere Felder (per GUI-VIEW-POSITION-POLISH-1). * Bestehende Filter / Searches funktionieren unverändert. * C1-Monitoring (Quality-Shadow) läuft weiter und sammelt Daten.