

PLAN_ENTRY_PRICE_SEMANTIC

Status: planned (backlog, awaits operator GO) **Priority:** P1 **Erstellt:** 2026-05-18 00:00 UTC (Bewertungs-Followup) **Quelle:** Entwickler-Audit EVALUATION_MONITOR_REPORT_20260517T221634Z.pdf **Roadmap-ID:** ENTRY-PRICE-SEMANTIC-CLARIFY
Voraussetzung: Monitoring-Fenster abgeschlossen. **Beziehung:** läuft parallel zu / vor DCA-STATE-RECONCILE — semantische Klärung muss vor State-Forensik stehen.

1. Befund (Code-verifiziert)

`pos['entry_price']` wird beim DCA mit dem **gewichteten Durchschnitt** überschrieben:

```
# trading/execution/paper_trade.py:507
pos['entry_price'] = round_price(new_cost / new_qty, price) # Neuer Durchschnittspreis

# trading/execution/live_trade.py:617
pos['entry_price'] = round_price(new_cost / new_qty, avg_price)
```

Es existiert **keine** separate Speicherung des **Initial-Entry**. Damit gilt drei verschiedene Semantiken je nach Lebenszyklus:

Position-Zustand	Bedeutung von <code>entry_price</code>
ohne DCA	Initial-Entry (Buy-Preis)
mit korrekt persistiertem DCA	aktueller Average nach allen Tranchen
mit DCA aber verlorenem State (XLM-Fall heute)	wieder Initial-Entry, weil State-Reload das Override verworfen hat

Operator und GUI können nicht unterscheiden, welche der drei Bedeutungen vorliegt.

2. Auswirkung

2.1 Operator-Display

Anzeige `Entry price 0.10767` für ENA bedeutet **post-2-DCA-Average**, sieht aus wie Initial-Entry. Operator-Entscheidung über SL/TP, Manual-Close, Risk-Verständnis ist auf falscher semantischer Basis.

2.2 PnL-Berechnung

`unrealized_pnl = (current_price - entry_price) * quantity` ist **konsistent**, weil `quantity` ebenfalls die kumulierte Tranchen-Quantity ist. Mathematisch stimmt der USDT-Wert. Aber der **Prozentwert** ist relativ zum Average, nicht zum Initial-Entry — also nicht „wie viel Prozent Verlust seit erstem Kauf“ sondern „wie viel Prozent Verlust auf den effektiven Average“.

Bei XLM (State-Verlust) ist der angezeigte `unrealized_pnl_pct` **wirklich falsch**, weil der Average post-DCA real ein anderer ist als der aktuell angezeigte Initial-Entry.

2.3 Risk/Reward in StrategyStatsBuilder

```
// gui/app/Services/StrategyStatsBuilder.php:153
(take_profit - entry_price) / (entry_price - stop_loss)
```

Wenn `entry_price` = post-DCA-Avg, ist das R/R relativ zum Average. Bei State-Verlust ist es relativ zum Initial-Entry. **Inkonsistent.**

2.4 `trade_logs.entry_price`

Beim Close wird `pos['entry_price']` (im finalen Zustand) in `trade_logs.entry_price` geschrieben. Für ENA wäre das post-2-DCA-Avg = 0.1077; für XLM wegen State-Verlust = 0.1516 (Initial). Trade-Historie ist damit **inkonsistent typisiert**.

2.5 Chart-Marker (TradeChartOHLCVProvider)

Render: „entry @ 0.10767“ als Linie auf dem Chart. Tatsächlicher Entry-Tick war bei 0.1080+ (Initial-Buy). Marker liegt falsch.

3. Zwei Fix-Optionen

Option A — `entry_price` = Initial-Entry behalten + neues `avg_price` -Feld

```
# new schema
pos['entry_price']      # immer Initial-Entry (write-once)
pos['avg_price']        # = entry_price wenn ohne DCA; = post-DCA-Avg wenn mit DCA
pos['initial_quantity'] # = quantity beim First-Buy
```

Pro: - semantisch sauber: entry_price = „erster Trade-Eintrag“ wie in jeder Trading-Software - Operator-intuitiv

Contra: - Breaking-Change in DCA-Trigger-Logik (dca_manager.py:144,260) — die berechnen DCA-Trigger relativ zum entry_price → müssen auf avg_price umgeschwenkt werden - Migration: existierende live_portfolio.json hat keinen Initial-Entry mehr (durch Overwrite verloren); für offene Positionen mit DCA wäre Initial-Entry irrekuperabel ohne Log-Forensik - 17+ Code-Stellen in trading/ und gui/ müssen entscheiden welches Feld sie wollen

Option B — entry_price = Avg behalten + neues initial_entry_price -Feld

```
# new schema
pos['initial_entry_price'] # write-once beim First-Buy, NIE überschrieben
pos['entry_price']        # bleibt post-DCA-Avg-Semantik (kein Code-Pfad ändert sich)
```

Pro: - Kein Breaking-Change in bestehender Logik (Trigger, PnL, SL-Berechnung) - Code-Pfad-Diff klein: 2 Writer (execute_buy first-time) + 1 GUI-Anzeige - Migration: für offene Positionen ohne DCA = entry_price; für offene Positionen mit DCA = aus bot_stdout.log rekonstruierbar (Forensik DCA-STATE-RECONCILE liefert das)

Contra: - Semantik „entry_price“ bleibt unintuitiv für externe Leser - Doppelte Begriffe in der DB (entry_price + initial_entry_price)

Option C — Rename-Migration

```
pos['avg_price']      # = umbenannt aus aktuellem entry_price
pos['initial_entry_price'] # neu
```

Pro: semantisch optimal. Contra: vollständiger Rename, alle 25+ Code-Stellen + DB-Spalten + GUI + Tests. Hoher Aufwand.

Empfohlene Option

Option B. Kleinster Diff, kein Breaking-Change, ermöglicht GUI-Klarheit ohne Trading-Logik-Risiko.

4. Scope Option B

4.1 Bot-Code (4 Stellen)

Datei	Change
trading/execution/paper_trade.py execute_buy first-time-buy-path	pos['initial_entry_price'] = price setzen wenn not allow_add (first buy)
trading/execution/live_trade.py execute_buy first-time-buy-path	analog
trading/execution/paper_trade.py:507 DCA-overwrite	unverändert (entry_price = avg, korrekt)
trading/execution/live_trade.py:617 DCA-overwrite	unverändert

4.2 Telemetrie (Snapshot/Trade-Emit)

Datei	Change
trading/main.py:448 per-scan emit_position_snapshot	initial_entry_price=pos.get('initial_entry_price') durchgeben
trading/db_emitter.py emit_position_snapshot	optionalen kwarg ergänzen + ins payload
Postgres position_snapshots.initial_entry_price numeric(18,8) NULL	neue Spalte (Migration)
analog trade_logs.initial_entry_price	

→ DB-Migration erforderlich (nicht-trivial), kollidiert mit Plan v3.4 boundary "0x DB-Migration während Monitoring". Erst nach Monitor.

4.3 GUI (PHP)

Datei	Change
gui/app/Filament/Resources/PositionSnapshotResource.php infolist	neue Section „Entry History“ mit <code>initial_entry_price + entry_price</code> (renamed Label „Average Entry“) + <code>dca_count</code>
gui/app/Filament/Widgets/FreshOpenPositionsWidget.php blade	optional Spalte „Initial / Avg“
gui/app/Services/Ohlcv/TradeChartOHLCVProvider.php	<code>chart-marker entry</code> aus <code>initial_entry_price</code> ableiten wenn vorhanden, sonst <code>entry_price</code>
gui/app/Services/StrategyStatsBuilder.php R/R	dokumentieren ob R/R relativ zu Initial oder Avg sein soll (Operator-Decision)
gui/app/Filament/Resources/TradeLogResource.php	analog Position-Detail

4.4 Migration für offene Positionen

Vor Cutover existieren 5 offene Positionen, davon (laut DCA-STATE-RECONCILE) 3 mit DCA-Historie (ENA, XLM, SHIB).

Backfill-Strategie: - ENA: `initial_entry_price = 0.1080` (aus `bot_stdout.log` 2026-05-16 05:50 entry) - XLM: `initial_entry_price = 0.1516` (== aktueller `entry_price`, weil State-Verlust ohnehin gleich) - SHIB: `initial_entry_price = 0.00000592` (aus log) - TRUMP / XAUT / (anderen ohne DCA): `initial_entry_price = entry_price`

Backfill via `live_portfolio.json` einmaliger Edit, dann Bot-Restart. Read-only forensisch geprüft VOR dem Edit (DCA-STATE-RECONCILE liefert die Wahrheit).

5. Boundaries

- **Bot-Touch ja** (Cutover SOT-1d)
- **DB-Migration ja** (neue Spalten `initial_entry_price` in 2 Tabellen)
- 0x Strategieparameter-Tuning
- 0x Mainnet
- 0x CommandBus-v6
- 0x automatischer Backfill — manuelle Forensik pro Position
- 0x DCA-Trigger-Logik (`entry_price` bleibt avg, `dca_manager` unverändert)

6. Reihenfolge

1. **DCA-STATE-RECONCILE P1.5** zuerst — liefert die belastbaren Initial-Entry-Werte für die Backfill-Migration
2. **ENTRY-PRICE-SEMANTIC-CLARIFY P1** danach — Code + Migration + GUI
3. **DCA-STATE-SAVE-RACE-FIX P1** danach — atomares Speichern stellt sicher dass künftiger State nicht mehr divergiert

7. Tests

- `test_initial_entry_price_persists_through_dca` — first-buy setzt initial, DCA überschreibt es **nicht**
- `test_initial_entry_eq_entry_when_no_dca` — Positionen ohne DCA haben `initial == entry`
- AST-Guard: keine andere Stelle im Code überschreibt `initial_entry_price`
- PHPUnit `test_position_resource_shows_initial_entry`
- Migration-Test: existing rows mit `initial_entry_price IS NULL` werden GUI-side mit Fallback auf `entry_price` gerendert (kein Crash)

8. Acceptance Criteria

- Für jede neue Position ist Initial-Entry persistiert und unveränderlich nach DCA
- GUI zeigt klar getrennt „Initial Entry“ und „Average Entry“
- bestehende Positionen ohne `initial_entry_price` → Fallback auf `entry_price` (sauberes Display ohne semantisches Drama)
- `trade_logs.initial_entry_price` gefüllt für alle Trades **nach Cutover**
- Historische `trade_logs` bleiben unverändert (kein DB-Mass-Mutation)
- DCA-Trigger-Logik unverändert

9. Risk

Risiko	Severity	Mitigation
Migration bricht historische Rows	mittel	NULL-erlaubende neue Spalten, kein Drop
GUI rendert NULL für Initial-Entry-Spalte bei alten Rows	klein	Blade-Fallback auf entry_price
Operator versteht Average vs Initial nicht	mittel	Tooltips + Helper-Text in Infolist
DCA-Trigger ausversehen auf Initial-Entry umgeschrieben	hoch	AST-Test: dca_manager.py liest nur entry_price (= avg), nicht initial
Backfill für offene Positionen fehlerhaft	mittel	manuelle Forensik pro Position aus log; kein Auto-Backfill

10. STOP

Kein Code vor: - Monitoring-Fenster abgeschlossen - DCA-STATE-RECONCILE forensisch durch - Operator- G0 ENTRY-PRICE- SEMANTIC-CLARIFY

Bis dahin: - Operator-Display zeigt weiter post-DCA-Avg als „Entry price“ mit dem aktuellen Risiko der Semantik-Verwirrung - GUI-VIEW-POSITION-POLISH-1 hat zumindest Sections + Back-Button, aber kein Initial-vs-Avg-Hinweis - Workaround bis Fix: Operator weiß dass entry_price = post-DCA-Avg ist (jetzt dokumentiert in Evaluation §4.1)