LABEL-1 Plan-Review — Exit Reason Subtypes

Datum: 2026-05-14 21:35 UTC | Repo: `Steve-TradingBot` | master HEAD: `431ada5` (EXEC-MODE-LABEL-1) | Bot-Image: `f6f11397f791` | Modus: **EXCHANGE_TESTNET** | Scope: **READ-ONLY** | Implementation: **GO (mit D1-D3)**

Executive Summary

Root-Cause der "stop_loss"-Sammellabel: alle SL-Trigger fließen in `paper_trade.py:update_prices` durch denselben Pfad `if price <= pos['stop_loss']: execute_sell(reason='stop_loss')`. Der `pos['stop_loss']`-Wert selbst kann jedoch durch 4 verschiedene Mechaniken vorher angehoben/verändert worden sein (Trailing, News-Tightening, Break-Even, 96h-Hard-Exit) — diese Information geht beim SL-Trigger verloren.

Lösung ohne DB-Migration: `position_manager.py` taggt die letzte Änderung an `pos['stop_loss']` mit `pos['_sl_kind']`. `paper_trade.py:update_prices` liest dieses Tag beim Trigger und mappt auf einen präzisen Subtyp. **Feld `trade_logs.exit_reason` ist bereits `varchar(255)`** — keine Migration nötig.

Strukturell wichtig: `LiveTrader(PaperTrader)` erbt von PaperTrader, `live_trader.update_prices()` ist tatsächlich `PaperTrader.update_prices()`. Nur `execute_sell` ist überschrieben (echte Binance-Orders). → Label-Logik landet in **paper_trade.py + position_manager.py**, gilt automatisch für Live-Trader.

GO/NO-GO: **GO** für Implementation (klein, ohne Migration, ohne Strategie-Touch).

1. Exit-Pfade (gefunden)

#	Datei:Zeile	Pfad	aktueller `reason`	Sub-Mechanik
1	`paper_trade.py:911`	`update_prices` SL-Branch	`'stop_loss'`	jeder SL-Trigger, egal ob original oder modifiziert
2	`paper_trade.py:918`	`update_prices` TP-Branch	`'take_profit'`	echter TP-Hit
3	`paper_trade.py:457`	`execute_sell(reason='manual')`	`'manual'` (default)	direkter externer Aufruf
4	`paper_trade.py:795`	T3 `check_tier3_exits` Hard-Stop	`'hard_stop_loss'`	T3 Hard-Stop −10%
5	`paper_trade.py:810`	T3 `check_tier3_exits` Time-Stop	`'time_stop_4h'`	T3 Time-Stop 4h
6	`paper_trade.py:824`	T3 Partial +50%	`'partial_profit_50pct'`	Einsatz raus
7	`paper_trade.py:832`	T3 Partial +100%	`'partial_profit_100pct'`	Half rest raus
8	`position_manager.py:82`	`apply_trailing`	(kein Sell — Mod)	hebt `pos['stop_loss']`
9	`position_manager.py:142`	`apply_news_adjustment`	(kein Sell — Mod)	BEARISH SL enger / BULLISH TP weiter
10	`position_manager.py:200`	`apply_breakeven` (60h)	(kein Sell — Mod)	SL auf Entry+0.8%
11	`position_manager.py:255`	`apply_breakeven` Hard-Exit (96h)	(kein Sell — Mod)	`pos['stop_loss'] = current * 1.10` → forciert nächsten SL-Trigger
12	`live_trade.py:660`	`LiveTrader.execute_sell`	passthrough	überschreibt PaperTrader-Methode mit echten Orders, `reason`-Parameter durchgereicht

Wichtige Erkenntnis: Die Pfade 8–11 ändern nur `pos['stop_loss']`, nicht den Exit. Wenn später `update_prices` den modifizierten SL-Trigger sieht, weiß es **nicht**, wie er entstanden ist.

2. Logische Exit-Typen im Code

Typ	existiert?	aktueller Label-Schlüssel	Notiz
Hard Stop-Loss (initial)	✓	`stop_loss` (uneindeutig)	`pos['stop_loss']` unverändert seit Entry
Trailing-Stop in Profit	✓	`stop_loss` (uneindeutig)	nach `apply_trailing`, pnl > 0
Trailing-Stop in Verlust	seltener Fall	`stop_loss` (uneindeutig)	nach Trailing-Anhebung + Reversal unter Trail-Level
Break-Even-Stop	✓	`stop_loss` (uneindeutig)	nach `apply_breakeven` (60h), SL = Entry + 0.8%
News-Tightened Stop	✓	`stop_loss` (uneindeutig)	nach BEARISH `apply_news_adjustment`
Hard-Time-Exit (96h)	✓	`stop_loss` (uneindeutig)	`apply_breakeven` setzt SL = current * 1.10
Take-Profit	✓	`take_profit` (korrekt)	echte TP-Trigger
News-Tightened TP-Hit	unscharf	`take_profit`	TP enger gezogen → TP-Hit
Manual Close	✓	`manual`	externer Aufruf
T3 Hard-Stop	✓	`hard_stop_loss`	T3-spezifisch, präzise
T3 Time-Stop 4h	✓	`time_stop_4h`	T3-spezifisch, präzise
T3 Partial-Profit	✓	`partial_profit_50pct` / `_100pct`	präzise
Circuit-Breaker-Exit	nein (CB blockt nur BUY)	n/a	kein Exit-Pfad
DCA-Rescue (kein Exit)	✓	n/a	Rescue-Pfad, kein Close

3. Felder im aktuellen Schema

Feld	Typ	Status	Bemerkung
`trade_logs.exit_reason`	`varchar(255)`	**vorhanden**	Strings reichen für alle Subtypes → **keine Migration**
`position_snapshots.metadata_json`	`json`	vorhanden	enthält bereits `exit_reason`-Key bei closed snapshots (main.py:335)
`trade_logs.mode` / `environment`	`varchar(255)`	vorhanden	gefüllt mit `paper` / `testnet` (mode-Wert noch Legacy — EXEC-MODE-LABEL-2 backlog)
`trade_logs.decision_id`	`varchar(255)`	53/53 NULL	DATA-LINK-1 (out-of-scope LABEL-1)
`trade_logs.opened_at`	timestamptz	53/53 NULL	DATA-LINK-1
`trade_logs.duration_seconds`	bigint	NULL	DATA-LINK-1

4. DB-Migration nötig?

Nein. Argumente:

- `exit_reason` ist `varchar(255)` → alle neuen Subtypes (`fixed_stop_loss`, `trailing_stop_profit`, ...) passen mit max. ~25 Zeichen
- `position_snapshots.metadata_json` ist freies JSON
- Filament-GUI rendert `exit_reason` als String, kein ENUM-Schema in DB
- Historische rows (2× `paper_trader.update_prices.close`, 1× `live_trader.update_prices.close`, alle mit `exit_reason='stop_loss'` / `'take_profit'`) bleiben unverändert → kein UPDATE

Wenn später ein Filament-`CHECK CONSTRAINT` o.ä. gewünscht ist → separater Plan-Review.

5. Subtypen-Vorschlag (final)

Neuer Subtype	Bedingung beim SL-/TP-Trigger	geschätzte Häufigkeit (53-Sample)
`fixed_stop_loss`	SL-Trigger, `pos['_sl_kind']` nicht gesetzt, pnl < 0	~23 (entspricht aktueller Verlust-Trades)
`trailing_stop_profit`	SL-Trigger, `pos['_sl_kind']=='trailing'`, pnl > 0	~26 (entspricht den 28 profitablen stop_loss-Trades)
`trailing_stop_loss`	SL-Trigger, `pos['_sl_kind']=='trailing'`, pnl < 0	seltener Fall — Trail angehoben, dann Reversal
`break_even_stop`	SL-Trigger, `pos['_sl_kind']=='breakeven'`	abhängig vom 60h-Alter
`news_tightened_stop_loss`	SL-Trigger, `pos['_sl_kind']=='news_bearish'`, pnl < 0	sentiment-getriggert
`time_stop_hard_exit`	SL-Trigger, `pos['_sl_kind']=='hard_exit_96h'`	Hard-Exit 96h
`take_profit`	TP-Trigger, `pos['_tp_kind']` nicht gesetzt	unchanged
`news_tightened_take_profit`	TP-Trigger, `pos['_tp_kind']=='news_bearish'`	sentiment-getriggert
`manual_exit`	`execute_sell(reason='manual')`	unchanged (rename `manual` → `manual_exit` für Konsistenz)
T3-Subtypes	bleiben: `hard_stop_loss`, `time_stop_4h`, `partial_profit_50pct`, `partial_profit_100pct`	präzise, kein Refactor nötig
`legacy_stop_loss`	Position vor LABEL-1 geöffnet, `_sl_kind` fehlt → Fallback	open positions zum Cutover-Zeitpunkt

Operator-Decisions D1–D3

- **D1**: Fallback für pre-LABEL-1 open positions → `legacy_stop_loss` ODER `fixed_stop_loss` (verwischt Trailing-Daten)?
- **D2**: T3-Subtypes vereinheitlichen mit Hauptbot-Naming (`t3_hard_stop_loss`)? Oder beibehalten?
- **D3**: `news_tightened_*` als eigene Subtypes ODER als Tag in `metadata_json` ?

6. Betroffene Dateien (Implementation-Phase)

Datei	Änderung	Est. LoC
`trading/execution/position_manager.py`	nach jedem Adjustment `pos['_sl_kind']` / `pos['_tp_kind']` setzen (`apply_trailing` → `'trailing'` , `apply_breakeven` → `'breakeven'` , Hard-Exit → `'hard_exit_96h'` , `apply_news_adjustment` → `'news_bearish'` / `'news_bullish'`)	~12
`trading/execution/paper_trade.py`	`update_prices` SL/TP-Trigger: Subtyp aus `pos['_sl_kind']` + `pnl_sign` ermitteln, `execute_sell(reason=subtype)` ; `execute_sell` selber bleibt unverändert	~15
`trading/main.py`	Optional: `metadata['sl_kind']` in `emit_position_snapshot` closed-Block übernehmen (Forensik)	~3
`trading/execution/live_trade.py`	**kein Touch** — erbt `update_prices` , hat eigenes `execute_sell` mit `reason` -Passthrough	0
`trading/db_emitter.py`	**kein Touch** — Feld bleibt String	0
Filament GUI `TradeLogResource`	Optional: Display-Mapping mit Icons pro Subtype — **out-of-scope LABEL-1** (Backlog **LABEL-1-GUI**)	0

Gesamt-LoC: ~30 Zeilen produktiv, ~50 Zeilen mit Tests.

7. Testplan

#	Typ	Inhalt
T1	Unit	Nach `apply_trailing` → `pos['_sl_kind'] == 'trailing'`
T2	Unit	Nach `apply_breakeven` 60h → `pos['_sl_kind'] == 'breakeven'`
T3	Unit	Nach `apply_breakeven` 96h Hard-Exit → `pos['_sl_kind'] == 'hard_exit_96h'`
T4	Unit	Nach `apply_news_adjustment` BEARISH → `pos['_sl_kind'] == 'news_bearish'`
T5	Unit	`update_prices` SL-Trigger ohne `_sl_kind` + pnl<0 → `reason='fixed_stop_loss'`
T6	Unit	`update_prices` SL-Trigger mit `_sl_kind='trailing'` + pnl>0 → `reason='trailing_stop_profit'`
T7	Unit	`update_prices` SL-Trigger mit `_sl_kind='trailing'` + pnl<0 → `reason='trailing_stop_loss'`
T8	Unit	`update_prices` SL-Trigger mit `_sl_kind='breakeven'` → `reason='break_even_stop'`
T9	Unit	TP-Trigger → `reason='take_profit'` (unchanged)
T10	Unit	T3 `execute_partial_sell` mit `reason='hard_stop_loss'` → bleibt
T11	Integration	DCA-Rescue löscht `_sl_kind` (sonst persistiert falsche Information nach Rescue)
T12	Integration	Bot-Restart mit open positions ohne `_sl_kind` in `live_portfolio.json` → graceful Fallback
T13	Live-Verify	Nächster natürlicher Close beim Bot → `trade_logs.exit_reason ∈ {subtypes}`
T14	grep	Keine Stelle setzt `reason='stop_loss'` mehr aus `update_prices`

Test-Ausführung: `python3 -m pytest trading/tests/test_label_1.py -v` (neue Test-Datei). KEIN `php artisan test` direkt — Safe Runner bei GUI-Touch (entfällt, GUI nicht touched). Live-Verify: Operator-Beobachtung 24–48h für Subtype-Verteilung.

8. Risikoanalyse

Risiko	Schwere	Mitigation
`pos['_sl_kind']` wird nicht beim DCA-Rescue zurückgesetzt → falsche Subtype-Zuordnung	mittel	`execute_buy(allow_add=True)` nach DCA muss `_sl_kind` löschen
`live_portfolio.json` hat keine `_sl_kind` für pre-LABEL-1 open positions	niedrig	Fallback `fixed_stop_loss` ODER `legacy_stop_loss` (D1)
Trailing-Logic zieht SL hoch, aber `_sl_kind` wird beim TP-Hit ignoriert	niedrig	TP-Hit nutzt eigenes `_tp_kind`; Default `take_profit` bleibt
T3-Subtypes weichen ab vom Hauptbot-Naming	kosmetisch	Operator-Decision D2
Filament-GUI zeigt neue Subtypes als unformatierte Strings	niedrig	LABEL-1-GUI als separate Phase
GUI-Filter / `audit_events` erwarten alten String `stop_loss`	mittel	grep im Filament-Code vor Implementation prüfen
Bot-Crash durch unerwartete `pos['_sl_kind']` in `_save_state`	niedrig	JSON-serializable strings, kein Risiko
Strategie-Verhalten ändert sich versehentlich	**STOP-Risiko**	Code nur Subtype-Label setzen, keine SL/TP-Wert-Änderung; alle reason-Tests müssen pass

9. Boundaries-Check (Plan-Phase)

Boundary	Status
Keine Strategieänderung	✓ (nur Labels, kein Param/Threshold-Touch)
Keine SL/TP/Trailing-Parameter-Änderung	✓
Keine DB-Migration	✓ (Feld passt)
Keine historischen DB-Mutations	✓
Keine künstlichen Trades	✓
Kein Mainnet	✓
Kein Push ohne separates GO	✓ master `431ada5` unverändert
Keine Secrets	✓
Kein env dump	✓
Kein compose config mit Secrets	✓
Kein Code in Plan-Phase	✓ (read-only)

10. GO / NO-GO für LABEL-1 Implementierung

GO mit folgenden Bedingungen:

1. Operator-Decisions D1 (`legacy_stop_loss` vs. `fixed_stop_loss` Fallback)
2. Operator-Decision D2 (T3-Subtype-Naming)
3. Operator-Decision D3 (news-tightened als Subtype vs. Tag)
4. Implementation-Phase Pre-Check: grep im Filament `gui/`-Code, ob `exit_reason='stop_loss'` irgendwo hardcoded gefiltert wird — falls ja: Operator-Pin für GUI-Compat-Anpassung im selben Commit
5. Tests T1–T14 müssen alle pass vor Image-Rebuild
6. Live-Verify mit Watchdog-Freeze während Bot-Recreate
7. Backup `live_portfolio.json` vor erstem Recreate
8. SoT-1d-Pfad: `git commit` → `docker compose build clawbot && up -d --no-deps clawbot` (kein Volume-Tanz)

Aufwand-Schätzung:

- Code: ~30 LoC produktiv + ~50 LoC Tests
- Implementation+Tests: ~1.5–2h
- Live-Verify-Window: 24–48h für statistische Subtype-Verteilung

11. Sequenz nach LABEL-1

1. **LABEL-1** (jetzt im Plan)
2. **DATA-LINK-1**: `decision_id` + `opened_at` befüllen → Regime-/Strategy-Attribution
3. ~3–5 Tage Daten sammeln
4. **TRAIL-1** Diagnose mit echten Subtype-Daten (zu enges Trailing? Median-Win nur $6)
5. **OHLCV-Backtest** mit korrekter Attribution
6. Optional: **LABEL-1-GUI** (Filament Display-Mapping mit Icons pro Subtype)
7. Operator-Decision Parameter-Tuning

STOP — Plan-Review fertig. Operator-GO für D1–D3 + Implementation erforderlich.