

PLAN_PHANTOM_SELL_LOOP_FIX

Status: planned (P0) **Priority:** P0 — kritisch vor Mainnet, dringend für Bot-Stabilität **Erstellt:** 2026-05-18 (Nightly-Audit-Finding) **Quelle:** reports/nightly/2026-05-18/ALERTS.md P0-A **Roadmap-ID:** PHANTOM-SELL-LOOP-FIX **Bezug:** Symptom durch SHIB-STATE-DRIFT-RECONCILE-1 (commit faa86b0) manuell beseitigt; Code-Pfad bleibt broken.

1. Befund (verifiziert)

Am 2026-05-17 zeigte der Bot für SHIB/USDT :

Counter	Wert
SL-Trigger-Logs	29
Fehlgeschlagene TESTNET SELLS (insufficient_funds permanent)	29
Position geschlossen -Logs	0
closed_trade-Einträge in DB	0
Loop-Dauer	22 h

Vergleichswerte alle anderen Symbole am selben Tag: 1:1:1 sauber (1 trigger → 1 sell → 1 close).

2. Root cause

```
# trading/execution/live_trade.py:714-721 (Zitat aus nightly forensik)
order_out = self._call_write(self._exchange.create_market_sell_order, ...)
if not order_out.ok:
    logger.error(f"{self._label} SELL {symbol} fehlgeschlagen: ...")
    self._track_sell_failure(symbol)
    return None
# Pfad endet hier – Position bleibt in self.state['positions']
```

Konsequenz: 1. Bot loggt Fehler. 2. `_track_sell_failure` inkrementiert internen Zähler. 3. `return None` ohne State-Cleanup. 4. Position bleibt in `live_portfolio.json`. 5. Nächster Scan-Cycle ruft `update_prices` → SL-Trigger erneut → SELL erneut → fail erneut. 6. **Endlos-Loop.**

3. Auswirkung

Layer	Effekt
Bot-Runtime	API-Spam, jede 2-3min ein Fehlerlog
Operator-Dashboard	„Open Position“ SHIB sichtbar obwohl exchange-side nicht mehr existent
Mainnet-Risk	Rate-Limit-Hit + Telegram-Spam + falsche PnL-Berechnung
State-Files	<code>live_portfolio.json</code> verschmutzt mit Phantom
Risk-Guard	nicht zuständig (Phase D = DCA-Rescue, nicht SELL-Fail)

4. Fix-Optionen

Option A — State-Cleanup bei permanent-fail (empfohlen)

In `live_trade.py:714-721` :

```
if not order_out.ok:
    logger.error(...)
    self._track_sell_failure(symbol)
    # NEW: für permanent-category fails den State-Cleanup triggern
    if order_out.category == 'permanent' and 'insufficient_funds' in (order_out.reason or ''):
        self._handle_phantom_position(symbol, reason=order_out.reason)
    return None

def _handle_phantom_position(self, symbol, *, reason):
    """Position ist exchange-side nicht mehr existent. State cleanup + emit closed_trade."""
    pos = self.state['positions'].get(symbol)
    if not pos: return
```

```
# Emit synthetic close with reason='phantom_position_cleanup'
self._emit_phantom_close(symbol, pos, reason)
# Remove from state
del self.state['positions'][symbol]
self._save_state()
# Operator-Alert
self._notify_operator_phantom(symbol, pos, reason)
```

Pro: - Selbstheilend - saubere `trade_logs` -Row (exit_reason= `phantom_position_cleanup`) - Operator wird benachrichtigt (Telegram) - State und Exchange werden konsistent

Contra: - Echter `insufficient_funds` -Befund auf einer realen Position (vorher hängender Auftrag, Reserve-Lock) würde fälschlich als Phantom interpretiert → Position gelöscht - Mitigation: vorher `fetch_balance(base_asset)` aufrufen und prüfen ob tatsächlich kleiner als `pos['quantity']`

Option B — Retry-Limit + Eskalation (defensiver)

In `_track_sell_failure`: - nach N Versuchen (z. B. 5) Position als `_stuck_state` markieren - Telegram-Eskalation - Bot weiter laufend, aber Position aus aktiver SL-Trigger-Schleife genommen - Operator muss manuell entscheiden

Pro: - konservativer - kein Auto-Delete

Contra: - Phantom bleibt im State sichtbar - 5+ fehlgeschlagene API-Calls vor Eskalation

Empfehlung — Hybrid

1. **Retry mit Cap** (Option B): erste 3 Versuche werden geloggt
2. **Auto-Cleanup nach 3 Fehlern** (Option A) mit Pre-Check auf `fetch_balance < pos.quantity`
3. **Telegram-Eskalation** bei Cleanup (Operator weiß sofort)

5. Boundaries

- Bot-Touch: **ja** (`live_trade.py`)
- Cutover SOT-1d nötig
- 0x DB-Migration (synthetic close nutzt existing `trade_logs.exit_reason`)
- 0x Mainnet — Default `BINANCE_TESTNET=true` bleibt
- 0x Strategieparameter

6. Cutover-Plan

1. Pre-Cutover Snapshot
2. Watchdog freeze `CUTOVER_FREEZE_PHANTOM_SELL_LOOP`
3. `docker compose build clawbot`
4. Container-Test im neuen Image:
5. Unit-Test für `_handle_phantom_position`
6. Mock-permanent-fail Test
7. `docker compose up -d --force-recreate --no-deps clawbot`
8. 3-Way MD5 für `live_trade.py`
9. Bot spawn + 0 Tracebacks
10. Watchdog re-enable

7. Tests

```
trading/tests/test_phantom_sell_loop_fix.py:- test_permanent_insufficient_funds_triggers_cleanup -
test_non_permanent_fails_keep_position - test_balance_precheck_prevents_false_cleanup -
test_telegram_alert_fired_on_cleanup - test_synthetic_close_emits_trade_log_with_correct_exit_reason - AST-Guard:
_handle_phantom_position ist defined und referenced
```

8. Acceptance Criteria

- Kein `insufficient_funds` -Loop > 3 Versuche
- Cleanup nur wenn `fetch_balance` bestätigt drift
- Telegram-Alert sichtbar
- `trade_logs` row mit `exit_reason=phantom_position_cleanup`
- `pos['_state_drift_reconcile']` -Meta gesetzt
- 0 Tracebacks nach Cutover

9. Risk

Risiko	Severity	Mitigation
Auto-Cleanup falscher Position bei echtem Reserve-Lock	hoch	Balance-Pre-Check
Telegram-Spam wenn mehrere Phantoms	mittel	Rate-Limit-Aggregator (RISK-GUARD-TELEGRAM-RATE-LIMIT)
State-Race wenn parallel manuelle Close-Action läuft	niedrig	mtime-cookie reload (HISTORY-1 Fix B)
Pre-Check <code>fetch_balance</code> rate-limited	niedrig	TTL-Cache 30s

10. STOP

Kein Code vor: - Operator- G0 PHANTOM-SELL-L00P-FIX - TRACK-SELL-FAILURE-ALERT P0 als Sibling-Plan reviewed (gemeinsamer Cutover) - C1-Monitor stabil + EXTERNAL-CHANNEL-CAP-ALIGN gebündelt überlegt

Bot main.py PID=2061 healthy, manuelle SHIB-Fix gilt bis Code-Fix durch.