STATE-CASH-RESET-1 — Forensikbericht

Datum: 2026-06-03 22:35 UTC **Modus:** READ-ONLY (kein State-Write, kein Code-Touch) **Root-Cause-Status:** ✅ **EXAKT IDENTIFIZIERT**

---

1. Executive Summary

Was passierte: Am **2026-06-03 13:37:10 UTC** wurde nach erfolgreichem `AVAX/USDT`-SELL (außerhalb des 2 h-Outage-Fensters) `_sync_balance()` aufgerufen. Binance-Testnet lieferte `USDT.free = 0.00` (anstatt ~9 829 USDT). Der Bot überschrieb seinen tracked-cash mit `0.00`. Ab dem Zeitpunkt blockiert der Max-Drawdown-Check alle neuen BUYs (100 % Drawdown vs 20 % Threshold).

Bug-Typ: kein Code-Bug im Bot — der Bot vertraut der Exchange-Antwort. **Kein externer State-Edit. Kein Watchdog-Reset.**

Eigentliche Ursache: Binance-Testnet-API hat einen inkonsistenten Balance-Wert während Outage-Recovery-Phase geliefert. Der Bot hat **keinen Sanity-Check** gegen absurde Cash-Reduktion (z.B. 100 % Drop).

---

2. Forensische Timeline

2.1 Cash-Trajectory aus Balance-Sync-Logs (2026-06-03)

Zeit	Event	State-Cash	Exchange-USDT	Delta
01:24:15	MORPHO BUY	10 029.79 →	9 829.21	−200.58 (Cost)
04:20:09	MORPHO SELL	9 829.21 →	10 030.30	+201.09 (Proceeds)
05:48:38	AVAX BUY	10 030.30 →	9 829.51	−200.79 (Cost)
07:58:34	RENDER BUY	9 829.51 →	9 628.91	−200.60 (Cost)
11:03–13:37	**Binance-Testnet 502-Outage** (~2 h 34 min)	unverändert	–	–
13:37:10	**AVAX SELL recovery → SYNC**	**9 628.91 → 0.00**	**0.00 (BOGUS!)**	**−9 628.91 ANOMALIE**
13:44:38	RENDER SELL recovery	0.00 → 0.00	0.00	kein Log (Delta < 0.01)
14:50:32	risk_manager triggert Drawdown-Warning (erstes Sichtbar)	0.00	0.00	–

Die ersten 4 Syncs sind perfekt konsistent mit Bot-internen Cost-/Proceeds-Buchungen. Der **5. Sync (13:37:10)** ist die Anomalie.

2.2 Erster „Max Drawdown" Trigger

```
2026-06-03 14:50:32 | execution.risk_manager | WARNING | 🚨 Max Drawdown erreicht: 100.0% >= 20.0%
```

Dies ist die **Konsequenz**, nicht die Ursache. Die Drawdown-Berechnung in `risk_manager.py:98`:

```
drawdown = (starting_capital - current_value) / starting_capital
# = (10000 - 0) / 10000 = 1.00 = 100%
```

Vorherige Scans (14:30, 14:40, 14:45) zeigen kein Drawdown-Warning → Drawdown-Check wertet `current_value` aus, welches **erst zu dem Zeitpunkt cash=0 + 0 open positions** war.

---

3. State-Files Diff

File	Mtime	Cash	Open	Closed	Size
`live_portfolio.20260601.bak`	2026-06-01 00:01	8 324.14	5	249	201 086
`live_portfolio.20260602.bak`	2026-06-02 00:02	9 841.46	1 (ETH)	266	212 712
`live_portfolio.20260603.bak`	2026-06-03 00:45	**10 029.79**	0	277	220 566
`live_portfolio.json` (current)	2026-06-03 14:50:32	**0.00**	0	280	222 541

Letzter guter State: `live_portfolio.20260603.bak` (00:45 UTC) **Erster schlechter State:** kein Backup — direkte Live-File-Mutation um 13:37:10 (im Speicher), erst um 14:50:32 als File geschrieben (via nächsten `_save_state()` nach Scan-Aktivität).

Erwarteter Cash: 10 029.79 + 0.31 (MORPHO) + 0.42 (AVAX) + 0.35 (RENDER) = **10 030.87 USDT Tatsächlicher Cash:** 0.00 USDT **Anomalie:** −10 030.87 USDT

---

4. Code-Pfad-Identifikation

4.1 Smoking-Gun-Stelle

Datei: `trading/execution/live_trade.py` **Methode:** `_sync_balance` (Zeilen 459–482)

```
def _sync_balance(self):
    out = self._call_read(self._exchange.fetch_balance, op_label="fetch_balance.sync")
    if not out.ok:
        logger.warning(...)
        return    # fail-soft: state.cash bleibt unverändert
    bal = out.result
    usdt = float(bal.get('USDT', {}).get('free', 0.0))   # ← bogus 0.0 möglich
    old_cash = float(self.state.get('cash', 0))
    if abs(old_cash - usdt) > 0.01:
        logger.info(f"⚖️ Balance-Sync: state {old_cash:.2f} → exchange {usdt:.2f} USDT")
    self.state['cash'] = round(usdt, 2)   # ← ÜBERSCHREIBT BLIND, kein Sanity-Check!
    if 'starting_capital' not in self.state or not self.state['starting_capital']:
        self.state['starting_capital'] = round(usdt, 2)
    self._save_state()
```

Aufrufstellen (alle nach Trade-Mutationen): - Z. 137: `__init__` (initialer Sync) - Z. 917: nach `execute_buy` Erfolg - Z. 1075: nach `execute_sell` Erfolg - Z. 1253: nach `execute_sell` (anderer Pfad) - Z. 1363: nach T3-Sell - Z. 1463: nach `execute_partial_sell_t1` Erfolg

4.2 Drawdown-Trigger-Quelle

Datei: `trading/execution/risk_manager.py:90-101`

```
def check_max_drawdown(self, portfolio, max_drawdown=0.20):
    starting_capital = portfolio.get('starting_capital', 10000)
    current_value = portfolio.get('total_value', starting_capital)
    drawdown = (starting_capital - current_value) / starting_capital
    if drawdown >= max_drawdown:
        logger.warning(f"📉 Max Drawdown erreicht: {drawdown:.1%} >= ...")
        return True
    return False
```

Konsumenten: - `main.py:1025` : vor jeder BUY-Decision im Legacy-Pfad → blockt BUYs - `main.py:1165` : zusätzliche Web-Signal-Pfad-Schutzschicht

4.3 Was NICHT involviert war

- ✅ **Kein State-Save-Pfad-Bug:** `_save_state()` schreibt nur was im RAM ist
- ✅ **Kein Crontab-Reset:** Crontab hat keine Einträge gegen `live_portfolio.json`
- ✅ **Kein Watchdog-Modifier:** `bot_watchdog.sh` referenziert State nicht für Mutation
- ✅ **Kein Container-Restart:** Container-StartedAt `2026-05-31T15:56:35Z` , ununterbrochen `Up 3 days (healthy)`
- ✅ **Keine Tracebacks:** 0 Errors im Log über 81 h
- ✅ **Outage-Resilience korrekt:** Pending-Marker-Rollback, Circuit-Breaker, alle Mechanismen wirkten

---

5. Exchange-Side-Verifikation (READ-ONLY)

Aktueller Binance-Testnet-State:

```
USDT: free=0.00, used=0.00, total=0.00
```

Aber das Wallet enthält etliche Testnet-Tokens (FRAX 1379, FOGO 18446, ZAMA 14818, CHIP 12658, etc.) — Reste aus historischen Trades + Testnet-Faucet-Drops.

Interpretation: Die Binance-Testnet-Antwort um 13:37:10 war konsistent mit dem aktuellen Sandbox-Zustand. Das Testnet hat die USDT-Balance nicht korrekt nach SELLs gutgeschrieben (möglicherweise Sandbox-Bug, oder andere Operatoren auf demselben Konto, oder Testnet-Faucet-Reset).

Wichtig: Ein einfacher `restore from exchange` würde wieder `cash=0` ergeben.

---

6. Wahrscheinlichste Root Cause

Single-Root: Binance-Testnet hat um 13:37:10 UTC einen inkonsistenten `fetchBalance` -Response geliefert (USDT-free = 0 statt ~9 829),

wahrscheinlich als Folge der ~2.5 h Outage-Recovery.

Sekundär (Bot-Side): `_sync_balance()` hat **keinen Sanity-Check** für plausible Cash-Werte: - Keine Min-Threshold (z. B. „> 50 % des starting_capital wenn 0 open positions") - Kein Delta-Check (z. B. „kein single-shot drop > 50 %") - Kein Fallback auf Bot-internes Accounting wenn Exchange offensichtlich bogus liefert

Tertiär (Design): Auf einem **Testnet** macht es konzeptionell wenig Sinn, sich auf die Exchange-Balance als Wahrheit zu verlassen — die Testnet-Sandbox ist eine geteilte Ressource und nicht zuverlässig.

---

7. Sichere Restore-Optionen

Option A — Restore aus Backup + Closes nachziehen (EMPFOHLEN)

1. **Code-Fix zuerst** (Sanity-Check in `_sync_balance`)
2. State-Edit (read-only verifiziert via Backup-Diff): `cash: 0.00 → 10030.87` = `live_portfolio.20260603.bak.cash` (10029.79) + Σ(MORPHO 0.31, AVAX 0.42, RENDER 0.35) = 10030.87
3. Bot wieder im Trading-Mode

Risiken: - Wenn Code-Fix nicht zuerst kommt: Nächster `_sync_balance()` setzt cash wieder auf 0 (passiert bei jedem SELL/BUY/Partial!) - State-Edit ist write — verstößt gegen current READ-ONLY-Boundary

Option B — Cash aus Exchange-Testnet-Balance neu setzen

Nicht möglich: Exchange-Balance = 0. Würde nichts ändern.

Option C — Bot im Idle-Mode lassen + Code-Fix zuerst

1. Bot bleibt offline für neue BUYs (SELLs würden funktionieren bei offenen Positionen — aber wir haben keine)
2. Code-Fix `_sync_balance` mit Sanity-Check
3. Cutover SOT-1d
4. State-Edit für cash-restore
5. Bot resumes

Risiken: keine — clean Plan-vor-Code-Pfad. Aber Bot bleibt idle länger.

---

8. Empfohlene Phase-Reihenfolge

#	Phase	Zweck
1	**SYNC-BALANCE-SANITY-1 (P0)**	Code-Fix: `_sync_balance` darf cash nicht auf < 50 % starting_capital reduzieren wenn keine open_positions vorhanden. Telegram-Alert bei verdächtigen Werten. Konfigurierbar via env: `SYNC_BALANCE_MIN_PCT=0.5`
2	**CASH-RESTORE-1 (P0)**	One-shot state-edit: cash = 10030.87 (mit Operator-GO und Pre-Edit-Backup)
3	**EXIT-REASON-FIX-1 (P1)**	LABEL-1 Misclassification (separat)
4	später	CORE-SLOT-RISK-LIMIT-1 (P2)

---

9. STOP / Operator-GO erwartet

Optionen: - **A** GO Plan komplett (SYNC-BALANCE-SANITY-1 + CASH-RESTORE-1) — Standard-Pfad mit Code-Fix zuerst - **B** Nur SYNC-BALANCE-SANITY-1 jetzt, CASH-RESTORE-1 später nach Verify - **C** Cash sofort restoren ohne Code-Fix (NICHT empfohlen — würde sich wiederholen) - **D** Anderes

Default-Empfehlung: A.

Code-Fix dauert <30 min: 1 File geändert + Test + SOT-1d-Cutover. Danach State-Edit + Bot operiert wieder normal.

STOP. Keine Änderung jetzt.
