

PLAN_DCA_LOG_ORPHANS_CLEANUP

Status: planned **Priority:** P1 **Erstellt:** 2026-05-18 (Nightly-Audit-Finding) **Quelle:** reports/nightly/2026-05-18/ALERTS.md P1-B **Roadmap-ID:** DCA-LOG-ORPHANS-CLEANUP

1. Befund

dca_log.json enthält **26 Einträge**, davon **25 Orphans** (keine korrespondierende offene Position):

- Schatten aus geschlossenen Trades (CBBTC , PNUT , CHIP , 1000CHEEMS , BABY etc.)
- Schatten aus alten Bot-Sessions
- Schatten aus CommandBus-manual-close (kein Cleanup-Hook)
- Schatten aus State-Drift-Reconcile-Aktionen

Aktuell offene Positionen: **nur TON** (1 Eintrag matched) → 25/26 = 96 % Orphan-Rate.

2. Root cause

Cleanup-Hook existiert nur in:

```
# trading/execution/dca_manager.py
def cleanup(self, symbol): ... # ruft del self.log[symbol]
```

Aufrufer: - ✓ trading/execution/paper_trade.py:execute_sell — beim SL/TP-Trigger - ✗
trading/command_worker.py:_handle_close_position — fehlt - ✗ State-Drift-Reconcile (z. B. SHIB heute) — fehlt - ✗ Beim Bot-Boot (alte Schatten aus voriger Session) — fehlt

3. Auswirkung

Effekt	Severity
Forensik unscharf	mittel — Orphans täuschen DCA-Historie vor
DCA-Erfolgsquote-KPI verfälscht	mittel — falsche dca_count -Aggregate
dca_log.json wächst monoton	niedrig — Datei klein, kein Performance-Risk
Operator-Verwirrung	mittel — JSON-Inspektion zeigt „Geister“

4. Fix-Strategie

Phase 1 — Cleanup-Hook-Erweiterung (Code)

Drei Stellen Cleanup einfügen:

```
# command_worker.py:_handle_close_position (nach erfolgreichem SELL)
if hasattr(self.dca_mgr, 'cleanup'):
    self.dca_mgr.cleanup(symbol)

# live_trade.py:_handle_phantom_position (im PHANTOM-SELL-LOOP-FIX)
if self.dca_mgr:
    self.dca_mgr.cleanup(symbol)
```

Phase 2 — Boot-Time-Reconcile

```
# beim Bot-Boot: dca_log entries ohne open position → orphan-warning
def _reconcile_dca_log_at_boot(self):
    open_symbols = set(self.state.get('positions', {}).keys())
    log_symbols = set(self.dca_mgr.log.keys())
    orphans = log_symbols - open_symbols
    if orphans:
        logger.info(f"dca_log boot-reconcile: {len(orphans)} orphans found – moving to dca_log_orphans.json archive")
        self._archive_orphans(orphans)
```

Archive statt Delete: Forensik bleibt erhalten in dca_log_orphans/YYYY-MM-DD_HHMMSS.json .

Phase 3 — One-Shot Backfill für aktuelle 25 Orphans

```
# manuell, mit Backup
cp dca_log.json dca_log.json.pre_orphan_cleanup
python3 archive_orphans.py
```

5. Boundaries

- Bot-Touch: ja (3 Code-Pfade)
- DB-Migration: keine
- 0x live_portfolio.json-Touch
- 0x Mainnet
- Cutover SOT-1d

6. Cutover-Plan

1. Pre-Cutover Snapshot
2. Watchdog freeze `CUTOVER_FREEZE_DCA_ORPHANS_CLEANUP`
3. `docker compose build clawbot`
4. Container-Tests
5. Recreate clawbot
6. 3-Way MD5
7. Boot-time reconcile fires → 25 Orphans archiviert
8. Verify `dca_log.json` hat nur noch Einträge für offene Positionen
9. Verify `dca_log_orphans/2026-05-XX.json` als Backup
10. Watchdog re-enable

7. Tests

```
trading/tests/test_dca_log_orphans_cleanup.py:- test_cleanup_called_after_command_worker_close -
test_cleanup_called_after_phantom_position_handler - test_boot_reconcile_detects_orphans -
test_boot_reconcile_archives_no_delete - test_open_position_not_treated_as_orphan
```

8. Acceptance Criteria

- Nach Cutover: `dca_log.json` ≤ N offene Positionen
- `dca_log_orphans/` -Verzeichnis mit Backup
- 0 zukünftige Orphans nach close-Event
- Operator-Forensik kann historische DCA-Daten weiter aus orphan-archive ziehen

9. Bundle-Empfehlung

Mit PHANTOM-SELL-LOOP-FIX + TRACK-SELL-FAILURE-ALERT gemeinsam ausrollen (alle drei berühren `live_trade.py` + `command_worker.py` Code-Pfade).

10. STOP

Operator-GO + Bundle-Cutover mit PHANTOM-SELL-LOOP-FIX abwarten.