FULLSTACK-ARCH-ERROR-REVIEW — strategy_group-Konsistenz T1/T2/T3/MS-Runner

Stand: 2026-05-16 15:10 UTC · Branch `master` · HEAD `7e5cf6e` **Scope:** read-only Review · keine Code-/DB-/Container-Mutation.

1. Executive Summary

Die Operator-Hypothese ist **technisch bestätigt**: Der `MultiStrategyRunner` schreibt für **jede** seiner 5 Strategien (`trend_follow`, `breakout`, `mean_reversion`, `vwap_mean_reversion`, `volatility_sweep`) ohne Unterscheidung `strategy_group='t1_core'`.

Aber: dieses Verhalten ist im aktuellen Code **bewusst gewählt** und durch einen **Regression-Test** (`test_t_split_2_emitter_wiring.py`) zementiert. Der Backlog-Pin **STRATEGY-GROUP-CONFIG-1** dokumentiert, dass T1/T2 *später* als getrennte Profile aufgesetzt werden sollen — die aktuelle Code-Variante ist die Interim-Lösung der Phase T-SPLIT-2.

Ergebnis: kein klassischer "Bug" (Code = Spec = Test), aber ein **Architektur-Designfehler** im Sinne der Fullstack-Dokumentation, die T2 als "aggressivere Momentum-/Pump-Strategien" beschreibt. Operator-Hypothese ist berechtigt.

Empfehlung: **Option C (strategy_id → strategy_group Mapping)** als P1-Phase **STRATEGY-GROUP-CONTRACT-1** vor jeder MS-Aktivierung (`MULTI_STRATEGY_DRY_RUN=false`).

2. Gefundener Hauptfehler

Aspekt	Befund
Mismatch	Architektur-Dokumentation (mein Full-Stack-PDF heute) sagt: "T2 Pump & Dump = aggressivere Momentum-Strategien". MS-Runner schreibt aber pauschal `t1_core`.
Ort	`trading/strategies/multi_strategy_runner.py:409` (emit_decision) + `:438` (execute_buy)
Schwere	P1 vor `MULTI_STRATEGY_DRY_RUN=false`. P2 solange Dry-Run.
Daten-Wirkung	Wenn MS-Runner jemals live tradet, würden Breakout-/Volatility-Sweep-Trades als T1 Standard erscheinen — verfälscht GUI-KPIs (StrategyGroupKpiWidget), PnL-Aggregation pro Gruppe, spätere Cap-/Risk-Profile pro Gruppe.
Test-Status	`test_multi_strategy_runner_tags_execute_buy_as_t1_core` **enforced** das aktuelle Verhalten — Fix muss diesen Test mit anpassen.

3. Belege aus Code & Dokumentation

3.1 MS-Runner schreibt pauschal `t1_core`

Datei: `trading/strategies/multi_strategy_runner.py`

```
# Line 409 (emit_decision)
strategy_group='t1_core',

# Line 438 (execute_buy)
# T-SPLIT-2: MultiStrategyRunner is the Standard / Core trading pipeline
# (Binance, Multi-Strategy). All positions opened from this path are
# tagged `t1_core` so subsequent decision/snapshot/trade emits carry
# the origin-tag.
strategy_group='t1_core',
```

Das Comment-Statement ist explizit: "*Standard / Core trading pipeline*" — unabhängig von der konkreten `decision['strategy_id']` (eine aus 5).

3.2 Regression-Test zementiert das Verhalten

Datei: `trading/tests/test_t_split_2_emitter_wiring.py`

```
def test_multi_strategy_runner_tags_execute_buy_as_t1_core(self):
    src = (_TRADING / "strategies" / "multi_strategy_runner.py").read_text()
    # Runner has exactly one execute_buy call site (the production
    # path); it must carry strategy_group='t1_core'.
    self.assertIn("execute_buy", src)
    self.assertIn("strategy_group='t1_core'", src,
        "MultiStrategyRunner.execute_buy call must tag t1_core.")
```

→ Wenn ich heute `t1_core` durch ein dynamisches Mapping ersetzen würde, schlägt dieser Test fehl. Der Test ist also Teil des Architekturvertrags und muss bei einem Fix mit aktualisiert werden.

3.3 Backlog STRATEGY-GROUP-CONFIG-1 sagt es selbst

Datei: `memory/backlog_strategy_group_config_1.md`

> "Operator möchte T1/T2 künftig als **echte unabhängige Strategieprofile** mit eigenen Risk-/Entry-/Exit-/Quellen-Parametern, **nicht nur als Klassifizierungs-Labels**."

→ Die heutige Implementierung ist **Interim** (Klassifizierungs-Label), nicht Endzustand. Operator hatte das selbst gepinnt mit harter **Stop-Regel:** `MULTI_STRATEGY_ENABLED=true` **bleibt NO-GO solange DATA-LINK-1-FU2 nicht umgesetzt** — DATA-LINK-1-FU2 ist heute durch (commit `32c95c7`), die T1/T2-Profilseparation aber noch nicht.

3.4 Andere T-SPLIT-2-Stellen

Datei	Zeile	strategy_group	Pfad	Bewertung
main.py	515,585,646,671,701,730,761	`STRATEGY_GROUP_T1`	run_scan_cycle emit_decision (evaluating/reject/buy)	✓ korrekt, das ist der Legacy-Scanner-Pfad
main.py	873	`STRATEGY_GROUP_T1`	_new_pos via execute_buy (Main-BUY tranche_1)	✓ Legacy-Scanner
main.py	997	`STRATEGY_GROUP_T2`	external web-channel BUY	✓ external pump-typical
main.py	1114	`STRATEGY_GROUP_T2`	telegram-bot-channel BUY	✓ external pump-typical
main.py	1310	`STRATEGY_GROUP_T2`	T3-forwarded BUY (`_process_t3_forwarded_signals`)	✓ T3-Bridge
main.py	349,375,427	`t.get('strategy_group')`	emit_trade/snapshot post-close	✓ propagation
multi_strategy_runner.py	409,438	`'t1_core'` hardcoded	MS-Runner emit_decision + execute_buy	**✗ uniform für 5 unterschiedliche Strategien**
paper_trade.py:_new_pos	(T-SPLIT-2)	aus kwarg	wird durchgereicht	✓ pure pass-through
live_trade.py	(T-SPLIT-2)	aus kwarg	wird durchgereicht	✓ pure pass-through

3.5 DB-Ist-Stand

```
trade_logs:
   t1_core         8   (Bot-Scanner-Closes + 1× HISTORY-1-Backfill für KITE)
   t2_pump_dump    3   (T3-forwarded od. external-channel)
   NULL           59   (pre-T-SPLIT-2 historische Rows)

position_snapshots:
   t1_core      1359
   t2_pump_dump  510
   NULL        18522   (pre-T-SPLIT-2)

decision_logs:
   t1_core    100284   (alle Scanner-Cycles + alle MS-Runner-evaluating sind hier — heute ist ALLES als t1_core gemarkt)
   NULL        76289   (pre-DATA-LINK-1-FU2 oder pre-T-SPLIT-2)
```

→ MS-Runner-Trades sind in den 8 trade_logs.t1_core **NICHT enthalten**, weil seit MS-Test-2-Start (DRY_RUN=true) kein MS-Trade real ausgeführt wurde. Die MS-Runner-decision_logs sind ebenfalls 0 wegen Dry-Run-Gate. Falls jetzt MS-Live aktiviert würde, würden ALLE neuen MS-BUYs als `t1_core` in trade_logs landen.

---

4. MS-Runner Datenfluss

4.1 Strategien (5 in STRATEGY_REGISTRY)

```
'trend_follow':         TrendFollowStrategy
'breakout':             BreakoutStrategy
'mean_reversion':       MeanReversionStrategy
'vwap_mean_reversion':  VWAPStrategy
'volatility_sweep':     VolatilitySweepStrategy
```

4.2 Pipeline

```
MultiStrategyRunner.run(symbols, data_by_symbol):
  pro symbol:
    regime = market_regime.detect_extended()
    router_result = StrategyRouter.route(regime)
      → active_strategies (Subset der 5)
    pro strategy_id in active_strategies:
      cls = STRATEGY_REGISTRY[strategy_id]()
      strategy_result = cls.score_signal(df, regime)
      decision = decision_engine.evaluate(strategy_result, ...)
        → {decision: TRADE_CANDIDATE|REJECT, strategy_id, ...}
      _attempt_execution(decision):
        [Safety-Gates]
        decision_log_id = generate_decision_id()
        emit_decision(strategy_group='t1_core', strategy_id=decision['strategy_id'], ...)
        trader.execute_buy(strategy_group='t1_core', strategy_id=..., ...)
```

→ `strategy_id` (der konkrete einer aus 5) **wird korrekt durchgereicht**. Nur `strategy_group` ist konstant `'t1_core'`.

4.3 Antwort auf Operator-Frage

> *Würde ein echter MS-BUY heute als T1 oder T2 in der DB landen?*

Antwort: **T1 (`t1_core`)** — alle 5 Strategien einheitlich.

Auch wenn ein `breakout`-Trade (eher T2-Pump-Charakter) oder ein `volatility_sweep`-Trade (Hochvolatilität-Hunt) live ginge, würde er als T1 markiert. Das **verwischt die T1-Auswertung** mit aggressiven Strategien.

---

5. GUI-Auswirkung

5.1 Widget `StrategyGroupKpiWidget` + `StrategyGroupKpiReader`

```
-- Reader query (vereinfacht aus StrategyGroupKpiReader.php)
SELECT DISTINCT ON (symbol)
   symbol, strategy_group, position_value
FROM position_snapshots
WHERE status='open'
```

→ **Wenn MS-Live geht**, würden ALLE MS-Trades die T1-KPI-Karte aufblähen (Exposure, Open-Positions-Count). Operator sieht "T1 hat 23 offene Positionen / 4500 USDT Exposure" — versteht aber nicht, dass 12 davon breakout/volatility-sweep-Trades aus dem aggressiven MS-Pfad sind.

5.2 TradeLogResource + PositionSnapshotResource

Beide Filter zeigen `strategy_group`-SelectFilter. Operator kann `t1_core` filtern, sieht aber gemischt: - Legacy-Scanner-Trades (konservativ, threshold-basiert) - MS-trend_follow-Trades - MS-breakout-Trades (sollte eher T2 sein) - MS-volatility_sweep-Trades (sollte eher T2 sein)

Operator-Workaround heute: zusätzlich nach `strategy_id` filtern. Aber KPI-Widget aggregiert NICHT pro strategy_id, nur pro strategy_group.

5.3 Legacy-Risiko

`COALESCE(strategy_group, 'legacy_unknown')` im Reader → NULL-Rows werden korrekt als "Legacy / Unknown" gerendert. Die 59 NULL-trade_logs + 18522 NULL-position_snapshots sind pre-T-SPLIT-2 historisch und kommen nicht mehr aus dem aktuellen Code (alle 14 emit-Sites carry einen Wert).

5.4 T3-forwarded BUYs

Bekommen `STRATEGY_GROUP_T2` (`t2_pump_dump`) via `_process_t3_forwarded_signals` in main.py:1310. **Das ist konsistent** mit der Doku ("T2 = Pump-Detect / Volatility / Listing aus tier3 subsystem").

---

6. T3-Subsystem-Analyse

6.1 Ist T3 wirklich ein Execution-Tier?

Nein. Genauer:

- T3-Subsystem-Code (`trading/tier3/`) verwaltet ein **eigenes isoliertes Portfolio** (`tier3_portfolio.json`) mit eigener Cash-Quote (20% des Bot-Kapitals)
- `Tier3Portfolio.buy()` schreibt aber nur den **lokalen JSON-State** — **keine echten Solana-Orders** (keine Jupiter-Swaps, kein sendTransaction)
- Die einzige reale Execution erfolgt für T3-forwarded-Signals via `live_trader.execute_buy(strategy_group='t2_pump_dump', ...)` auf Binance Testnet — also als T2-getaggte Trades, NICHT als T3

6.2 Strategy-Group-Verteilung für T3

Pfad	strategy_group in DB	Wert
T3-Pump.fun-Detection → Bonding-Curve >= 85%	nichts in trade_logs (keine echte Solana-Order)	—
T3-Pump.fun-Detection → Graduation	nichts in trade_logs	—
T3-Signal → Market-Cap-Gate forwards an T1/T2 → Binance Testnet	`t2_pump_dump` ✓	✓ korrekt
T3-Pump.fun in `tier3_portfolio.json`	(kein DB-Write, nur File-State)	—

→ `STRATEGY_GROUP_T3` (`'t3_copy_trading'`) **wird heute NIE geschrieben.** Der Wert ist nur als Konstante reserviert für eine geplante T3-EXECUTION-Phase (echte Solana-Trades).

6.3 Eigene Meinung zu T3

Ja, die Architektur-Bezeichnung "T3 = Execution-Tier" ist heute irreführend:

- Der Name suggeriert ein drittes Trading-Subsystem mit eigener Execution
- In Wahrheit ist T3 ein Signal-/Daten-Layer für Solana-Memecoins
- Real ausgeführt wird nur was via Forward-Brücke nach T1/T2 (Binance) geht
- Das `tier3_portfolio.json` ist semantisch ein "Was wäre wenn"-Tracker, kein realer Portfolio-Stand

Vorschlag: in der Doku klarer kennzeichnen als:

> T3-Subsystem ≠ T3-Strategy-Group. Das Subsystem ist Signal-Layer (Solana monitoring). Die Strategy-Group `t3_copy_trading` ist für eine zukünftige Phase reserviert (T3-EXECUTION via Jupiter/DEX), heute nicht aktiv.

---

7. Fehlerklassifikation

Fehler	Schwere	Risiko	Sofort fixen?
MS-Runner schreibt pauschal `t1_core` für 5 verschiedene Strategien	**P1** (vor MS-Live) / P2 (Dry-Run)	PnL/KPI/Cap pro Gruppe falsch aggregiert	nein, aber vor `MULTI_STRATEGY_DRY_RUN=false` zwingend
T-SPLIT-2 Regression-Test zementiert das Verhalten	P1 (zusammen mit Fix)	Mit-Update beim Fix nötig	nein, kommt mit dem Fix
Dokumentation "T3 = Execution-Tier" irreführend	P3	Operator-Missverständnis	nein, Doku-only
Fehlende per-strategy-Tests	P2	Regression-Risiko bei späteren Strategy-Adds	nein, kommt mit Fix
NULL-Rows pre-T-SPLIT-2 in DB (legacy_unknown)	P3	nur kosmetisch — historisch	nein, Operator-Boundary "keine Massenbereinigung"

---

8. Eigene Architekturmeinung — Option A/B/C-Bewertung

Option A — MS komplett T1

- **Heutiger Zustand**.
- Vorteile: 0 LoC-Änderung, konservativ.
- Nachteile: maskiert Strategy-Group als reines "Pipeline-Origin" statt Risikoprofil. Wenn STRATEGY-GROUP-CONFIG-1 später eigene Cap-/Risk-/Exit-Regeln pro Gruppe einführt, müssten 5 Strategien einzeln umverteilt werden.
- **Bewertung**: pragmatisch heute (Dry-Run blockiert eh), aber gefährlich bei Aktivierung.

Option B — MS komplett T2

- Alle MS-Trades als `t2_pump_dump`.
- Vorteile: trennt MS sauber vom Legacy-Scanner. T1 bleibt "klassischer Scanner".
- Nachteile: `mean_reversion` und `trend_follow` sind **nicht** Pump-Charakter — werden fälschlich aggressiv markiert.
- **Bewertung**: zu grob. Sammelt Konservatives + Aggressives wieder in einer Bucket.

Option C — strategy_id-basiertes Mapping (Operator-Präferenz)

Vorgeschlagenes Mapping:

strategy_id	Charakter	→ strategy_group
`trend_follow`	Conservative trend rider, longer holds	`t1_core`
`mean_reversion`	Counter-trend, RSI extremes	`t1_core`
`vwap_mean_reversion`	VWAP-Range, neutral-conservative	`t1_core`
`breakout`	Momentum on volatility	`t2_pump_dump`
`volatility_sweep`	High-Vol hunting	`t2_pump_dump`
(T3-forwarded)	external pump/listing	`t2_pump_dump`
(Legacy main-scanner)	Score-Threshold-Pipeline	`t1_core`

- **Aufwand**: ~30 LoC (Mapping-Dict + Lookup in MS-Runner) + 1 Test-Update
- **Risiko**: niedrig — kein DB-Schema-Change, kein Bot-Restart außer SOT-1d
- **Datenqualität**: PnL/KPI pro Gruppe wird semantisch korrekt
- **Zukunftsfähigkeit**: matched STRATEGY-GROUP-CONFIG-1 perfekt
- **Testbarkeit**: AST-Tests + unit-tests pro strategy_id-Bucket möglich
- **Bewertung**: **klare Empfehlung**.

Diskussions-Punkte zu Option C

1. **`mean_reversion` und `vwap_mean_reversion` als T1?** Argument FÜR: counter-trend ist nicht Pump-typisch. Argument GEGEN: bei Memecoins ist mean_reversion oft auch riskant. → Operator-Decision-Point.
2. **Was wenn Strategy nicht in Map?** Default-Fallback auf `t1_core` ODER `legacy_unknown` (defensiv).
3. **Kann sich das Mapping zur Laufzeit ändern?** Heute: Pinnen als Konstante. Phase STRATEGY-GROUP-CONFIG-1d würde später GUI-edit + Audit erlauben.

---

9. Fix-Plan — STRATEGY-GROUP-CONTRACT-1

Ziel

strategy_id → strategy_group eindeutig mappen, MS-Runner respektiert das Mapping, T-SPLIT-2-Test aktualisiert, NIE Mainnet, keine DB-Migration, kein Massenupdate historischer Rows.

Geänderte Dateien

Datei	Änderung	LoC
`trading/strategies/multi_strategy_runner.py`	Neue Helper-Funktion `_strategy_group_for(strategy_id)` + 2 hardcoded `'t1_core'`-Strings durch Lookup ersetzen	~15
`trading/strategies/registry.py` ODER neue `strategies/strategy_group_map.py`	`STRATEGY_ID_TO_GROUP` Dict pin (5 Strategien explizit, default-fallback)	~25
`trading/tests/test_t_split_2_emitter_wiring.py`	Test `test_multi_strategy_runner_tags_execute_buy_as_t1_core` ersetzen durch Mapping-Assertion (alle 5 IDs durchgehen)	~30
`trading/tests/test_strategy_group_contract_1.py` NEU	5 Unit-Tests pro strategy_id-Bucket	~80

Bot-/Worker-/GUI-Touch

- **Bot-Recreate**: 1× via SOT-1d (Image-Rebuild für multi_strategy_runner.py)
- **Worker**: nicht betroffen (Worker nutzt LiveTrader.execute_buy direkt, übergibt strategy_group als kwarg vom Caller)
- **GUI**: nicht betroffen (Reader liest nur DB-Spalte; bestehende Labels passen)

Historische DB-Werte

- **Keine Mutation** historischer Rows. Operator-Boundary stay.
- T1-Rows aus Legacy-Scanner bleiben t1_core (korrekt).
- Die 8 t1_core trade_logs (1 davon der KITE-Backfill) bleiben unverändert.
- Die 3 t2_pump_dump trade_logs (web/telegram/T3-forwarded) bleiben.
- Nach Fix-Cutover landen NEUE MS-Trades mit korrektem Mapping.

Stop-Regeln

- **HARTE Stop-Regel** bleibt: `MULTI_STRATEGY_DRY_RUN=false` ist NO-GO solange STRATEGY-GROUP-CONTRACT-1 nicht durch ist.
- Kein Mainnet
- Keine Strategieparameter-Änderung
- Kein Push ohne separates GO

Was zwingend vor MS-Live erledigt sein muss

1. STRATEGY-GROUP-CONTRACT-1 implementiert + getestet ✓
2. T-SPLIT-2 Test angepasst ✓
3. Bot-Recreate via SOT-1d + 3-Way MD5 verifiziert ✓
4. Erste 5 MS-decision_logs-Rows in DB zeigen 5 verschiedene strategy_id, jeweils mit dem erwarteten strategy_group (live verifiziert in Dry-Run mit `MULTI_STRATEGY_ENABLED=true`)
5. Operator-GO für Phase MS-Live-Aktivierung (eigene Decision-Phase)

---

10. Tests

Test	Was	Erwartung
`test_strategy_group_map_complete`	STRATEGY_ID_TO_GROUP enthält alle 5 STRATEGY_REGISTRY-IDs	pass
`test_strategy_group_map_values_in_allowlist`	jeder Mapping-Wert ist in `_STRATEGY_GROUP_ALLOWED` von db_emitter	pass
`test_trend_follow_maps_to_t1_core`	strategy_id='trend_follow' → 't1_core'	pass
`test_mean_reversion_maps_to_t1_core`	'mean_reversion' → 't1_core'	pass
`test_vwap_maps_to_t1_core`	'vwap_mean_reversion' → 't1_core'	pass
`test_breakout_maps_to_t2_pump_dump`	'breakout' → 't2_pump_dump'	pass
`test_volatility_sweep_maps_to_t2_pump_dump`	'volatility_sweep' → 't2_pump_dump'	pass
`test_unknown_strategy_id_falls_back_to_t1`	default für unbekannten strategy_id	pass (defensiv)
`test_msr_emit_decision_uses_lookup`	AST: MS-Runner emit_decision Call-Site enthält `_strategy_group_for(`	pass
`test_msr_execute_buy_uses_lookup`	AST: dasselbe für execute_buy	pass
`test_t_split_2_no_hardcoded_t1_in_msr`	AST: kein literal `'t1_core'` mehr in multi_strategy_runner.py	pass — ersetzt T-SPLIT-2-Regression-Assertion

Regression-Suite-Wirkung

Folgende existierende Tests müssen aktualisiert werden: - `test_t_split_2_emitter_wiring.py::test_multi_strategy_runner_tags_execute_buy_as_t1_core` → ersetzen durch Mapping-Assertion
Folgende bleiben grün: - `test_data_link_1_fu2.py` — decision_id/strategy_id-Wiring unverändert - `test_command_bus_close_testnet_1/2/3.py` — kein MS-Runner-Bezug - `test_decision_log_metadata_emit.py` — Legacy-Scanner-Pfad - alle T-SPLIT-1-Schema-Tests

---

11. GO / NO-GO Empfehlung

GO für STRATEGY-GROUP-CONTRACT-1 Implementation

- Scope klein und scharf umrissen (~150 LoC inkl. Tests)
- Risiken-Profil niedrig (Dry-Run, kein Live-Impact)
- Erfüllt eine **zwingende Vorbedingung** für `MULTI_STRATEGY_DRY_RUN=false`
- Vermeidet künftige Daten-Aufräumarbeit (richtig labeln vom ersten Live-Trade an)

NO-GO bis Operator-Entscheidung

- **Mapping-Tabelle**: bestätige meine 5-zu-2-Aufteilung (trend_follow/mean_reversion/vwap → T1; breakout/volatility_sweep → T2) ODER setze sie anders. Das ist Operator-Domain-Wissen.
- **Default-Fallback**: t1_core (konservativ) oder legacy_unknown (defensiv)?
- **Zeitpunkt**: jetzt (während MS-Test-2 läuft → MS-Test wird reset) ODER nach Ende des aktuellen 24h-Dry-Runs?
- **Test-Replacement**: T-SPLIT-2-Test darf umgeschrieben werden? (Test ist Vertrag, Update erfordert bewusste Vertrags-Änderung)

---

12. Antworten zur Phase-9-Frage des Operators

> Ist die Fullstack-Architektur konsistent?

Nein — die Doku beschreibt T2 als "Pump-Dump-/Momentum-Strategien", der Code labelt aber alle 5 MS-Strategien einheitlich als T1.

> Wenn nein, wo genau nicht?

Im `MultiStrategyRunner._attempt_execution` (Zeilen 409 + 438) + `test_multi_strategy_runner_tags_execute_buy_as_t1_core`.

> Welche Komponente schreibt falsche Daten?

`MultiStrategyRunner` selbst — alle anderen Komponenten propagieren nur durch.

> Wie wirkt sich das auf GUI/PnL/Caps/Strategieanalyse aus?

- GUI `StrategyGroupKpiWidget` aggregiert T1-Karte mit gemischtem Risiko
- PnL-pro-Gruppe-Aggregation verfälscht
- Spätere Cap-/Risk-Profile pro Gruppe (STRATEGY-GROUP-CONFIG-1) lassen sich nicht durchsetzen ohne nachträgliche Datenkorrektur
- Strategieanalyse je strategy_id ist weiterhin möglich (strategy_id ist korrekt propagiert), aber strategy_group-Aggregation ist semantisch unscharf

> Was ist der sicherste Fix?

STRATEGY-GROUP-CONTRACT-1 mit Option C (Mapping pro strategy_id) — ~150 LoC, 11 neue Tests, 1 Test-Update, 1 Bot-Recreate,

0 DB-Mutation, 0 Mainnet, 0 Push, 0 Strategieparameter-Change.

---

13. STOP

Read-only Review abgeschlossen.

Boundaries gehalten: - 0× Code-Mutation - 0× DB-Migration - 0× DB-Write - 0× historische Updates - 0× Strategieparameter-Änderung - 0× Bot-/Worker-/GUI-Restart - 0× Mainnet - 0× Push - 0× Secrets im Output - 0× env dump - 0× docker compose config

Operator-Entscheidung erforderlich:

1. Bestätigung der Mapping-Tabelle (Option C)
2. Default-Fallback-Wahl
3. Zeitpunkt (jetzt während MS-Test-2 oder nach 24h-Ende)
4. GO für STRATEGY-GROUP-CONTRACT-1 Implementation