# Masterplan POS-CAP

## Operator-steuerbare Position-Caps via Laravel-Backend

**Audit-Datum**: 2026-05-12
**Status**: PLAN ONLY — kein Code-Touch in diesem Dokument
**Verfasser**: Steve-TradingBot Architektur-Audit
**master HEAD**: `ea11637`

---

## 1 — Executive Summary

Aktuelle Lage: Der Bot zeigt im Trading-Log "4/4 Max offene Positionen erreicht", obwohl Settings (Container-`.env`) auf 5 stehen. **Ursache**: Das BEAR-Regime injiziert in `trading/scanner/regime.py:139` einen Hardcoded-Override `max_positions: 4`, der via `max_override` Parameter höchste Priorität in `RiskManager.check_open_positions()` hat — über `active_config` und `Settings`.

Ziel: **Der Operator soll die Position-Caps pro Markt-Regime über das Filament-Admin-Panel steuern können**, ohne die Strategie-/Trading-Decision-Logik anzufassen.

Vorgeschlagene Lösung: Per-Regime-Caps in den bestehenden G9 Risk-Allowlist + G10 Apply-Path integrieren. Bot-Side regime.py-Refactor: statt Hardcode aus `ActiveConfigProvider.max_positions_<regime>` lesen. Wenn keine Operator-Override gesetzt → bestehende Defaults bleiben (rückwärts-kompatibel).

**Aufwand-Schätzung**: 4-6h verteilt auf 6 Mini-Phasen. Reversibel via `git revert` + runtime_config clear. Kein Mainnet-Risiko (TESTNET-only durable).

---

## 2 — Audit-Befund (Recap aus read-only Analyse)

### 2.1 Effektive Limit-Quelle

```
Bot-Log (live):
  [Regime] Markt: BEAR | Min Score: 7.5 | Positions: 4 | Bias: MEAN_REVERSION
  execution.risk_manager | Max offene Positionen erreicht: 4/4
```

### 2.2 Code-Pfad

```
trading/scanner/regime.py
  Zeile 130: BULL     → max_positions: 5
  Zeile 139: BEAR     → max_positions: 4   ← aktuell aktiv
  Zeile 148: SIDEWAYS → max_positions: 4
  Zeile 157: default  → max_positions: 4

trading/main.py:698,1081
  max_pos = regime.get('max_positions', 5) if regime else 5
  risk_mgr.check_open_positions(open_positions, max_pos, active_config=...)

trading/execution/risk_manager.py:105-122
  Priority-Kaskade:
    1. max_override (regime-imposed)        ← BEAR injiziert HIER 4
    2. active_config.max_open_positions     (5 via fallback)
    3. self.max_open_positions = Settings   (5 via .env)
```

### 2.3 Sekundärer Cap

```
trading/execution/paper_trade.py:356  if len(state['positions']) >= 5 and not allow_add:
trading/execution/live_trade.py:534   if len(state['positions']) >= 5 and not allow_add:
```

Beide Files haben einen **hardcoded** `>= 5` als Defense-in-Depth. Würde mit Regime-Cap 4 nicht relevant; bei Operator-Lift auf > 5 wäre das ein zweiter Touchpoint.

### 2.4 Klassifikation

Es ist **kein Bug**, sondern intentionale konservative Regime-Adaptierung. Im BEAR-Markt fewer Slots, im BULL-Markt mehr.

---

## 3 — Architektur-Ziele

| | Ziel |
|---|---|
| **G1** | Operator kann pro-Regime Caps im Filament-Panel ändern (keine SSH-/CLI-Schritte) |
| **G2** | Änderungen werden via bestehende CommandBus + G10 Apply-Path ausgerollt |
| **G3** | Bot übernimmt neue Werte per Scan-Cycle (kein Restart nötig nach Apply, analog G10-4.2) |
| **G4** | Strategie-Logik (Regime-Detection, Score-Threshold, MEAN_REVERSION-Bias) bleibt unangetastet |
| **G5** | Rückwärts-kompatibel: ohne Operator-Override behält Bot exakt heutiges Verhalten |
| **G6** | Mainnet-Block intakt (5-Layer-Guard) |
| **G7** | Audit-Trail: jede Cap-Änderung als `risk_setting.set` / `risk_setting.cleared` Audit-Event |
| **G8** | Schema-bounded: min=1, max=8 (oder ähnlich konservativ, Operator wählt Phase-1-Bounds) |
| **G9** | Defense-in-Depth: paper_trade.py + live_trade.py hardcoded ceiling sollte ≥ Schema-max bleiben |

---

## 4 — Design-Optionen

### Option A — Single global cap

Der Operator setzt **einen** Wert `max_open_positions` (z.B. 6). Regime ignoriert dieses Feld; der einzelne globale Wert gilt für alle Regimes.

**Vorteile**:
- Einfachste Implementierung — 1 Allowlist-Eintrag, 1 Settings-Override
- Bestehender G9 Schema-Eintrag `max_open_positions` (cap 8) kann 1:1 verwendet werden
- Bot-Side: 2 Zeilen in `main.py` ändern (regime-fallback entfernen)

**Nachteile**:
- **Verliert** regime-adaptive Konservativität (BEAR könnte überfüllen, BULL bleibt konservativ)
- Operator muss manuell tracken, wie sich Regime auf Risk auswirkt
- Bricht das durable Pattern "Regime-driven Risk-Profile" (Phase R0-N regime-aware)

### Option B — Per-Regime caps

Der Operator setzt **drei** Werte: `max_positions_bull`, `max_positions_bear`, `max_positions_sideways`. Default beim ersten Apply = aktuelle Hardcodes (5/4/4); Operator kann pro Regime nach oben/unten justieren.

**Vorteile**:
- Erhält regime-adaptive Verhalten (auf Bot-Side fast unverändert, nur Quelle der Werte wechselt)
- Symmetrisch zur bestehenden Architektur (Regime-Config liefert per-Regime parameters wie `sl_atr_mult`, `tp_atr_mult`, `position_size_mult` — Cap fällt in dieselbe Kategorie)

- Operator kann gezielt experimentieren (z.B. BULL auf 7, BEAR bei 4 lassen)
- G9 Schema-Erweiterung 3 neue Keys mit identischer Bound-Logik

**Nachteile**:
- 3 Felder statt 1 → UI etwas länger
- Allowlist + Schema-Tests +9 Cases statt +3

### Empfehlung: Option B

Begründung:
1. Architektur-konsistent mit bestehenden Regime-Parametern (sl/tp/sizing pro Regime in regime.py:118-160)
2. Bewahrt das defensive Verhalten — Operator kann nicht versehentlich BEAR-Cap erhöhen ohne es zu wollen
3. Schema-Erweiterung trivial; G9-3 Filament-UI rendert automatisch alle Schema-Keys (Reusable Component)
4. Default-Werte beim ersten Apply spiegeln heutige Hardcodes wider → echte Backward-Compat

---

## 5 — Phasen-Roadmap

### POS-CAP-0: Decision-Lock (15 min, doc-only)

Operator entscheidet: Option A oder B. Defaults pinnt:
- BULL: 5 (heute)
- BEAR: 4 (heute)
- SIDEWAYS: 4 (heute)
- Min/Max bounds: [1..8] (analog Phase1RiskAllowlist.max_open_positions)

**Output**: Decision-Pin in `pos_cap_decision.md` Memory.

### POS-CAP-1: PHP-Side Schema-Erweiterung (45 min)

#### Files

```
gui/app/Services/ConfigProfile/Schema/Phase1RiskAllowlist.php
  → +3 SchemaField entries:
      'max_positions_bull'      (TYPE_INT, min=1, max=8, riskLevel=MEDIUM)
      'max_positions_bear'      (TYPE_INT, min=1, max=8, riskLevel=MEDIUM)
      'max_positions_sideways'  (TYPE_INT, min=1, max=8, riskLevel=MEDIUM)
  → Optional: deprecate 'max_open_positions' single key (oder beibehalten als
    legacy-fallback; gentle migration)
```

Per Spec G9-1 ist Phase1RiskAllowlist die kanonische Quelle für key/min/max. Service-Layer (G9-2 `RiskSettingService`) übernimmt automatisch ohne Code-Change. UI (G9-3 Filament Risk Settings Section) rendert ebenfalls automatisch.

#### Tests

```
gui/tests/Unit/Services/ConfigProfile/Schema/Phase1RiskAllowlistTest.php
  → +5 new tests:
      test_max_positions_bull_field_in_allowlist
      test_max_positions_bear_field_in_allowlist
      test_max_positions_sideways_field_in_allowlist
      test_max_positions_per_regime_bounds_1_to_8
      test_max_positions_keys_have_medium_risk_level
```

## POS-CAP-2: Bot-Side ActiveConfigProvider-Mapping (30 min)

### File

```
trading/active_config_provider.py
  SETTINGS_FALLBACK ergänzen:
    "max_positions_bull":     "MAX_POSITIONS_BULL",       # default 5
    "max_positions_bear":     "MAX_POSITIONS_BEAR",       # default 4
    "max_positions_sideways": "MAX_POSITIONS_SIDEWAYS",   # default 4
```

### Settings ergänzen

```
trading/config/settings.py
  + MAX_POSITIONS_BULL: int = int(os.getenv('MAX_POSITIONS_BULL', '5'))
  + MAX_POSITIONS_BEAR: int = int(os.getenv('MAX_POSITIONS_BEAR', '4'))
  + MAX_POSITIONS_SIDEWAYS: int = int(os.getenv('MAX_POSITIONS_SIDEWAYS', '4'))
```

Defaults spiegeln aktuelle Hardcodes in `regime.py` 1:1.

## POS-CAP-3: Bot-Side regime.py-Refactor (45 min)

### File

```
trading/scanner/regime.py
  → in den dict-Generatoren statt Hardcode:
      BULL:     max_positions = active_config.max_positions_bull
                                ?? Settings.MAX_POSITIONS_BULL
      BEAR:     max_positions = active_config.max_positions_bear
                                ?? Settings.MAX_POSITIONS_BEAR
      SIDEWAYS: max_positions = active_config.max_positions_sideways
                                ?? Settings.MAX_POSITIONS_SIDEWAYS
```

**Wichtig**: regime.py muss `active_config` Parameter erhalten (oder ActiveConfigProvider-Lookup pro Call). Pattern bereits bei `risk_manager._resolve_max_open` etabliert (G10-4.1). Strikt: KEIN Touch an `regime_detect`-Logik (BULL/BEAR/SIDEWAYS Klassifikation bleibt unangetastet — nur die config-dict-Konstanten werden parametrisiert).

### Tests

```
trading/tests/test_pos_cap_regime_lookup.py (NEW)
  Pin:
    test_bear_max_positions_uses_runtime_override_when_set
    test_bear_max_positions_falls_back_to_settings_when_no_override
    test_settings_falls_back_to_default_4_when_no_env
    test_active_config_priority_over_settings
    test_main_py_still_passes_max_pos_to_risk_manager
```

### POS-CAP-4: Filament UI (0 min — auto)

Die G9-3 Risk Settings Section in `ConfigProfileResource/Pages/EditConfigProfile` rendert automatisch alle Schema-Keys aus Phase1RiskAllowlist. Keine UI-Änderung nötig. Operator sieht nach POS-CAP-1:

```
Risk Settings:
  max_risk_per_trade_pct      [0.005]   bound 0.0025–0.01
  max_total_exposure_pct      [0.20]    bound 0.10–0.40
  max_open_positions          [4]       bound 1–8       (legacy single)
  max_positions_bull          [5]       bound 1–8       (NEW)
  max_positions_bear          [4]       bound 1–8       (NEW)
  max_positions_sideways      [4]       bound 1–8       (NEW)
  daily_loss_limit_pct        [0.02]    bound 0.005–0.03
  ...
```

### POS-CAP-5: Test-Surface vollständig (30 min)

```
PHP-Side:
  gui/tests/Unit/Domain/Allowlists/Phase1RiskAllowlistTest.php   +5 tests
  gui/tests/Feature/G9-3-UI-RenderingTest.php                    (regression — auto-rende

Bot-Side:
  trading/tests/test_pos_cap_regime_lookup.py                    +6 tests (siehe POS-CAP-1
  trading/tests/test_pos_cap_active_config_provider.py           +3 tests (mapping correct

Expected: ~14 neue tests
Total Bot-side: 81+9 = ~90
Total GUI-side: 587+5 = ~592
```

### POS-CAP-6: Pre-Deploy Drift-Audit + Deploy (40 min)

Per durable rule `feedback_pre_docker_cp_drift_check.md`:

1. 6-Punkt-Audit für betroffene Bot-Files: `regime.py`, `settings.py`, `active_config_provider.py`
2. Backup-Checkpoint
3. `docker cp` 3 Bot-Files + 1 neuer test-File
4. Bot SIGTERM + watchdog respawn
5. Smoke: scan-cycle läuft, regime.max_positions kommt aus Settings (nicht Hardcode mehr)
6. Closure-Pin

### POS-CAP-7: Operator-Drill (15 min, optional)

1. Filament: Edit Config Profile → Risk Settings → `max_positions_bull` von 5 auf 7
2. Apply Profile (Hard-Confirm Pattern)
3. Worker `--once` (oder Daemon)
4. runtime_config.json wird geschrieben
5. Bot picks up override per-cycle
6. Log: `[Regime] Markt: BULL | Min Score: 6.5 | Positions: 7 | ...`
7. Clear Runtime Config
8. Log: `[Regime] Markt: BULL | ... Positions: 5` (Default zurück)

---

## 6 — Datenmodell

### Keine DB-Migration nötig

Die `risk_settings`-Tabelle existiert bereits (G9-2). Neue Keys werden einfach via Schema-Allowlist hinzugefügt. Persistenz funktioniert automatisch über bestehende `RiskSettingService::setRiskSettings()`.

### Audit-Events (auto via bestehende G9-2 Logik)

```
risk_setting.set      (per key change)
risk_setting.cleared  (via clear_runtime_config command)
```

Bereits G-DR-11 prefix-separated von baseline. / *runtime_config.* / managed.*.

### runtime_config.json Schema-Erweiterung

```
{
  "schema_version": 1,
  "applied_at": "2026-05-12T...Z",
  "command_id": "...",
  "max_risk_per_trade_pct": 0.005,
  "max_open_positions": 4,                (legacy single)
  "max_positions_bull": 7,                (NEW)
  "max_positions_bear": 5,                (NEW)
  "max_positions_sideways": 5,            (NEW)
  ...
}
```

Bot's `ActiveConfigProvider` liest pro Cycle, Bot-Verhalten ändert sich ab nächstem Scan.

---

## 7 — Bot-Side Änderungen im Detail

### File 1: `trading/config/settings.py` (+3 Konstanten)

Additive Erweiterung, keine bestehenden Felder geändert. Env-overridable per `MAX_POSITIONS_BULL` / `_BEAR` / `_SIDEWAYS`.

### File 2: `trading/active_config_provider.py` (+3 Mapping-Zeilen)

Erweitert `SETTINGS_FALLBACK` dict um die 3 neuen Keys → Settings-Konstanten. Pattern identisch zu bestehenden 9 Keys.

### File 3: `trading/scanner/regime.py` (Cap-Constanten parametrisieren)

Statt:

```
'max_positions': 4,  # Hardcoded BEAR
```

wird:

```
'max_positions': self._resolve_max_positions(regime='bear', active_config=active_config),
```

oder pragmatischer ohne method-Signature-Change:

```
'max_positions': active_config.get('max_positions_bear') \
                 if active_config else Settings.MAX_POSITIONS_BEAR,
```

Detail-Ausarbeitung in POS-CAP-3 Code-Phase.

### File 4: `trading/main.py` (1 Zeile + active_config passing)

`regime.get('max_positions', 5)` bleibt unverändert weil regime.py das Feld jetzt dynamisch füllt. Möglicherweise muss `regime`-Detection-Aufruf `active_config` als Parameter erhalten (Detail-Klärung in POS-CAP-3).

### Keine Touches an

- `risk_manager.py` (Priority-Cascade bleibt unverändert; `max_override` aus regime ist immer noch korrekt)
- `paper_trade.py` (Hardcoded ceiling 5 bleibt als Defense-in-Depth — solange Schema-max bei 8 bleibt, ist die Defensive bei 8 zu setzen wäre eigene Mini-Phase POS-CAP-8 optional)
- `live_trade.py` (analog paper_trade.py)
- Strategie-Files (`strategies/mean_reversion.py`, `strategies/pattern_recognition.py`, etc.) — keine Touches

---

## 8 — PHP-Side Änderungen im Detail

### File 1: `gui/app/Services/ConfigProfile/Schema/Phase1RiskAllowlist.php`

3 neue `SchemaField`-Einträge im `allowlist()`-Array. Analog zum bestehenden `max_open_positions`-Eintrag.

### File 2: `gui/tests/Unit/Services/ConfigProfile/Schema/Phase1RiskAllowlistTest.php`

+5 tests pro POS-CAP-1.

### Keine Touches an

- G7 Config Profile Editor (Schema-driven, auto-pickup)
- G9-2 RiskSettingService (key/value-agnostisch)
- G9-3 Filament UI (Form auto-rendert via Schema)
- G10-4.2 Apply Path (Schema-bounded, picks up automatically)
- G10-5 Clear Path (key-agnostisch, picks up automatically)
- G10-6 Status Widget (read-only, picks up automatically)

**→ Mit nur 2 PHP-File-Changes (1 Allowlist + 1 Test) ist der UI-Pfad komplett.**

---

## 9 — Apply-Flow aus Operator-Sicht

```
1. Admin meldet sich in Filament an
2. Config Profiles → "Default" (oder aktives Profil)
3. Risk Settings Section
4. Neuer Eintrag "Max Positions BULL" → 7 eintippen
5. Speichern → Audit-Event risk_setting.set
6. Apply Profile → Hard-Confirm Pattern (Profilname tippen)
7. Background Command "apply_profile_testnet" → Pending
8. Operator triggert Worker (manuell --once oder Daemon greift auto)
9. Worker schreibt runtime_config.json:
     "max_positions_bull": 7,
     ...
10. Bot's ActiveConfigProvider sieht den neuen Wert ab nächstem Scan
11. Im nächsten BULL-Scan: "[Regime] Markt: BULL ... Positions: 7"
12. Bei BEAR-Scan immer noch: "Positions: 4" (unverändert)
```

Revert:

```
1. Operator → Runtime Config Status Widget → "Clear"
2. Hard-Confirm "clear-runtime-config"
3. Background Command "clear_runtime_config" → Worker
4. runtime_config.json wird gelöscht
5. Bot fällt auf Settings-Defaults zurück (5/4/4)
```

---

## 10 — Test-Plan-Übersicht

| Test-Datei | Anzahl | Type |
|---|---|---|
| `Phase1RiskAllowlistTest.php` | +5 | Unit |
| `G9-3 Risk Settings UI` regression | +0 (auto via schema) | Feature |
| `test_pos_cap_regime_lookup.py` | +6 | Unit |
| `test_pos_cap_active_config_provider.py` | +3 | Unit |
| Bot regression (regime.py callers) | full surface | Regression |
| **Total neu** | **~14** | |

Per durable rule: Bot-Side Tests müssen ohne psycopg2/ccxt laufen (host-side). GUI-Tests laufen in Container.

---

## 11 — Boundaries / Stop-Rules

### Was POS-CAP NICHT tut

- ❌ Keine Strategie-Logik-Änderung (Regime-Detection bleibt unverändert)
- ❌ Keine Buy/Sell-Logik-Änderung
- ❌ Keine Risk-Formel-Änderung (max_risk_per_trade bleibt unangetastet)
- ❌ Keine Ordergröße-Änderung
- ❌ Keine Score-Threshold-Änderung
- ❌ Keine Mainnet-Aktivierung (5-Layer-Guard bleibt intakt)
- ❌ Kein Managed Transfer
- ❌ Keine DB-Migration (risk_settings table existiert)
- ❌ Kein Container-Rename
- ❌ Kein neuer CommandType (apply_profile_testnet bleibt unverändert)
- ❌ Keine MH-1-Scope-Vermischung (paralleler Strang)

### Stop-Trigger während POS-CAP

| | Bedingung | Aktion |
|---|---|---|
| **S1** | Schema-Bounds-Drift (z.B. min > max) | STOP, Phase1Allowlist revertieren |
| **S2** | regime.py Detection-Logic bricht | STOP, sofortiger Code-Revert |
| **S3** | Tests rot | STOP, kein Deploy |
| **S4** | Apply schreibt unerwartete Keys in runtime_config.json | STOP, runtime_config rollback |
| **S5** | Bot-Restart führt zu Tracebacks | STOP, Backup-Restore |
| **S6** | Mainnet-Flag betroffen | sofort STOP |

---

## 12 — Risiken

| Risk | Wahrsch. | Impact | Mitigation |
|---|---|---|---|
| Schema-Erweiterung bricht G9-3 UI auto-render | sehr niedrig | mittel | G9-3 form ist schema-driven, neue keys werden gerendert ohne UI-Code-Touch — regression-test vor deploy |
| Bot-Side regime.py refactor bricht regime detection | niedrig | hoch | regime.py wird strikt minimal getouched (nur dict-value-source, nicht detection-logic). 6 dedicated tests sichern es ab |
| Operator setzt versehentlich BEAR=8 → zu aggressiv im Bear-Markt | mittel | mittel | Hard-Confirm Pattern (G10-6 Profilname-bestätigung) verhindert Klick-Fehler; Audit-Trail dokumentiert |
| runtime_config.json Drift zwischen Worker-Schreibe und Bot-Lese | sehr niedrig | mittel | bestehende G10-4.2 Two-File-Atomic-Pattern (sha256-verified) — wir reuse die Apply-Infra |
| paper_trade.py / live_trade.py Hardcoded `>= 5` ceiling cappt > 5 | sicher | bei Operator-Lift > 5 | dokumentiert; POS-CAP-8 optional als follow-up phase, hebt ceiling auf 8 oder zieht aus Settings |
| Bot-Restart nötig nach POS-CAP-6 | sicher | mittel | SIGTERM + Watchdog Pattern (bewährt durch G1/G1-FU-1a/MH-0.5/Notifier-1 = 5× clean restarts heute) |

---

## 13 — Aufwand-Schätzung

| Phase | Dauer | Risk | Code-Touch |
|---|---|---|---|
| POS-CAP-0 Decision-Lock | 15 min | minimal | doc-only |
| POS-CAP-1 PHP Allowlist | 45 min | klein | 2 Files (Allowlist + Test) |
| POS-CAP-2 Bot Settings + Provider | 30 min | klein | 2 Files (settings.py + active_config_provider.py) |
| POS-CAP-3 Bot regime.py | 45 min | mittel | 2 Files (regime.py + 1 neuer test) |
| POS-CAP-4 Filament UI | 0 min (auto) | minimal | 0 Files |
| POS-CAP-5 Tests vollständig | 30 min | minimal | regression-runs |
| POS-CAP-6 Pre-Deploy + Deploy | 40 min | mittel | docker cp + Restart |
| POS-CAP-7 Operator-Drill (optional) | 15 min | mittel | live apply + clear |
| **Total** | **~3.5h** | klein-mittel | 6-7 Files |

---

## 14 — Backlog-Dependencies + Reihenfolge

### Prerequisites

| Check | Status |
|---|---|
| G6.5 Command Bus | ✅ MVP completed (commit `ed38fa3`) |
| G7 Config Profile Editor | ✅ Phase-1 complete (commit `04cfae9`) |
| G9-1 Risk Allowlist Schema | ✅ approved (commit `2c91a45`) |
| G9-2 RiskSettingService | ✅ approved (commit `7abc462`) |
| G9-3 Risk Settings UI | ✅ approved (commit `49390c5`) |
| G10-4.2 Apply Profile to Testnet | ✅ closed (commit `f75c0e8`) |
| G10-5 Clear Runtime Config | ✅ closed (commit `d88d34c`) |
| G10-6 GUI/Operator Finalization | ✅ closed (commits `d69ccf2/aaa03c2/592bf37`) |
| ActiveConfigProvider in Bot | ✅ G10-4.1 deployed |
| MH-0.5 Worker-Daemon | ✅ closed (commit `e745171`) |

**Alle Prerequisites ✅ — keine Blockaden.**

### Compatibility mit Parallel-Strängen

| Strang | Kompatibel mit POS-CAP? |
|---|---|
| MH-1 Capability-System (just committed `ea11637`) | ✅ orthogonal — MH-1 ist managed-holdings, POS-CAP ist risk-config; getrennte Code-Pfade |
| MH-2-9 RECON-MH Folge | ✅ orthogonal |
| Container-Rename | ✅ orthogonal (kein Touch an compose) |
| MH-NAMING-CLEANUP Layer C/D | ✅ orthogonal (kein paper/testnet flag touch) |
| B-FEE-FIX Phase 2 | ✅ orthogonal |
| G1-FU-1b run_scan_cycle bulk-hoist | ✅ orthogonal (lazy-import lifts, kein regime.py touch) |

### Empfohlene Einbettung

POS-CAP kann **vor oder nach MH-1 weitere Sub-Phasen** ausgeführt werden — keine Reihenfolge-Abhängigkeit. Operator wählt nach Risiko-/Operator-Bedürfnis.

---

## 15 — Mainnet-Implikationen

**Keine in Phase 1**:
- Allowlist allowedEnvironments bleibt `['testnet']` (analog G9 / G10)
- Apply-Path bleibt testnet-only (G10-3.5 Service-Hardcode `dry_run_must_be_true` + `environment=testnet`)
- Mainnet-Block-Layer 1-5 unverändert
- BINANCE_TESTNET=true intact

**Falls später Mainnet-Aktivierung kommt**:
- POS-CAP-Werte würden in Mainnet-Profil separat persistiert (config_profiles.environment=mainnet)
- Mainnet-Defaults sollten konservativer sein (BULL 3, BEAR 2 z.B.)
- Diese Entscheidung fällt in Mainnet-Sign-Off-Phase, **nicht in POS-CAP-1**

---

## 16 — Memory-Pin Vorschlag (nach Closure)

```
pos_cap_closure.md
  - Commit-SHAs der 6 Phasen
  - Test-Bilanz vor/nach
  - Drift-Audit Ergebnis
  - Backup-Pfad
  - Operator-Drill-Outcome
  - Boundary-Snapshot
  - BACKLOG: POS-CAP-8 ceiling-lift auf 8 in paper_trade.py + live_trade.py (optional)
```

## 17 — Empfehlung Operator-Entscheidung

| | Frage | Empfehlung |
|---|---|---|
| 1 | Option A oder B? | **B (per-regime)** — architektur-konsistent, defensiv |
| 2 | Min/Max-Bounds | `1..8` (analog bestehendes max_open_positions) |
| 3 | Default-Werte beim ersten Apply | BULL=5, BEAR=4, SIDEWAYS=4 (Backward-Compat) |
| 4 | Bot-Restart akzeptabel? | ja (Pattern bewährt durch 5× Watchdog-Respawn heute) |
| 5 | POS-CAP vor oder nach MH-2? | egal, orthogonal — Operator-Präferenz |
| 6 | Optional POS-CAP-8 Ceiling-Lift jetzt mit oder später? | später (separate Mini-Phase, low-prio) |

## 18 — Abschlussempfehlung

POS-CAP ist eine **niedrig-risiko, hoch-leverage** Operational-Hardening-Phase:
- Verteilt 3 Hardcodes (4/4/5 in regime.py) auf operator-steuerbare Settings
- Nutzt bestehende G9 + G10 Infra zu 100% — kein neues Pattern
- Reversibel via runtime_config clear (G10-5)
- Test-Coverage existiert für alle berührten Komponenten
- 0 Mainnet-Risiko, 0 DB-Migration, 0 new command-types
- Backward-Compat garantiert (no operator override → exakt heutiges Verhalten)

**Empfehlung**: Operator-Decision auf Option B, dann sequenziell POS-CAP-1 bis POS-CAP-7 ausführen. Ca. 3.5h verteilt, jeder Schritt einzeln approbierbar mit eigenem Operator-GO.

---

*Ende Masterplan POS-CAP*