CAP-1-PREFLIGHT — Profit-Reinvestment / Profit-Reserve Mode — Audit-Bericht

Datum: 2026-05-09 (UTC 11:20)
Status: Read-only Audit, keine Mutation, durable Architecture-Vorschlag
Scope: Profit-Reinvestment / Profit-Reserve Mode für den Trading-Bot

---

1. Aktueller Kapitalfluss als Baum

```
.env STARTING_CAPITAL=10000 USDT
        │
        ▼
PaperTrader._load_state()                                  (paper_trade.py:42)
  state['starting_capital'] ──────────────────────►  STATIC, never updated
  state['cash']             ──────────────────────►  DYNAMIC, mutated per trade
  state['positions']        ──────────────────────►  open positions
  state['closed_trades']    ──────────────────────►  history (no aggregate!)
  state['tier3_budget']     ──────────────────────►  T3 SEPARATE pool ⚠
  state['tier3_used']       ──────────────────────►  T3 already has split!
  state['_audit_log']

execute_buy()    →  state['cash'] -= total_cost      (paper_trade.py:209)
execute_sell()   →  state['cash'] += net_proceeds    (paper_trade.py:322)
execute_t3_buy() →  state['cash'] -= (size + fee)    (paper_trade.py:416)
execute_partial_sell → state['cash'] += net          (paper_trade.py:514)

get_portfolio() returns:                               (paper_trade.py:86-110)
  cash             = state['cash']                   (current, after-trade)
  positions_value  = Σ qty × current_price
  total_value      = cash + positions_value
  starting_capital = state['starting_capital']  (static)
  pnl              = total_value − starting_capital  (= unrealized + realized)
  pnl_pct          = pnl / starting × 100

Sizer-Inputs (alle Pfade — main.py + sizers):
  portfolio['cash']           ← used for "available cash"
  portfolio['total_value']    ← used for "portfolio_value" (sizing-base)
  portfolio['positions_value']← used for exposure-check

Drawdown-Check (risk_manager.py:90-100):
  drawdown = (starting_capital − total_value) / starting_capital

Daily-Loss-Limit (main.py:192):
  daily_limit = starting_capital × 0.02
```

Befund (Aufgabe 1)

Frage	Antwort
Wird heute Compound-Mode implizit verwendet?	**JA**. `cash` wächst durch Profits, wandert direkt in den nächsten Trade. Sizer rechnen mit `total_value` (= cash + positions), das schließt unrealized + realized PnL **automatisch ein**.
Gibt es heute eine Reserve-Logik?	**NEIN**. Nur statische `RESERVE_PCT=0.20` in `position_sizer.py` (= 20% Cash-Buffer pro Trade). Das ist KEINE Profit-Reserve, nur ein Sizing-Buffer.
Wird Exchange-Balance vs state['cash'] gemischt?	Per `BalanceGuard`: bei live `_exchange.fetch_balance()`, fallback `state['cash']`. Bei Paper: nur `state['cash']`.
starting_capital wandert je in Profit/Reserve?	**NEIN**. Static. Nur PnL-Anzeige + Drawdown-Floor.
Hat T3 bereits Capital-Split?	**JA** (`tier3_budget`, `tier3_used`, `tier3_positions`). Vorbild für T-SPLIT.

Read-Sites Tabelle

Datei:Zeile	Funktion	Wert	Sizing?	Anzeige?
`paper_trade.py:92`	`get_portfolio()`	`cash + positions_value`	**ja**	ja
`paper_trade.py:96`	PnL	`total_value − starting`	nein	ja
`paper_trade.py:209,322,416,514`	execute_*	`cash ±= …`	n/a	n/a
`risk_manager.py:95-98`	drawdown	`starting_capital`	gates buy	nein
`main.py:192`	daily_limit	`starting × 0.02`	gates buy	nein
`main.py:618,652,764,849,1023`	sizer-input	`cash` direkt	**ja**	nein
`multi_strategy_position_sizer.py:152`	sizer	`cash`	**ja**	nein
`position_sizer.py:78-86`	sizer	`cash`, `total_value`	**ja**	nein
`balance_provider.py:142`	usable_balance	exchange balance ⇨ `cash` fallback	**ja**	nein
`dashboard.py:714`	UI	`starting`, `closed_trades`, PnL	nein	**ja**
`reporter.py:82,149`	UI	`starting_capital`	nein	ja

---

2. Datenmodell-Befund

Bot-State (`live_portfolio.json`)

Feld	Vorhanden?	Bedeutung
`cash`	ja	aktueller Cash inkl. realized PnL
`starting_capital`	ja	static, init aus .env
`positions`	ja	offene Positionen
`closed_trades[]`	ja	Historie mit `pnl` , `fee_buy` , `fee_sell` , etc.
`tier3_*`	ja	T3 hat eigenen Capital-Pool
`_audit_log`	ja	Mutation-Audit-Trail
`realized_pnl_total`	**fehlt**	wäre `Σ closed_trades[].pnl`
`profit_reserve`	**fehlt**	wäre der CAP-1-Kandidat
`investable_capital`	**fehlt**	wäre derived aus cash − reserve
`base_capital`	**fehlt**	(= starting_capital? oder eigener Marker?)
`profit_mode_activated_at`	**fehlt**	Aktivierungs-Marker
`reinvestment_mode`	**fehlt**	`compound` / `reserve_profits` / `partial_reinvest`

GUI-DB

Tabelle	profit-relevante Spalten
`trade_logs`	`realized_pnl` , `realized_pnl_pct` (per Trade, nicht aggregiert)
`bot_statuses`	`total_pnl` , `exposure` (pro Heartbeat)
`position_snapshots`	`unrealized_pnl` , `unrealized_pnl_pct`
`config_profiles` / `risk_settings`	KEINE profit-spezifische Spalte
`commands` / `audit_events`	KEINE profit-spezifische Spalte

Source-of-Truth-Empfehlung

Bot-State (`live_portfolio.json`) ist die Source-of-Truth für Cash + Positions. **GUI-DB** ist Read-only Spiegelung via Bot-Emitter (G6.1).

CAP-1 sollte **abgeleitete Werte** im Bot-State **berechnen** (z.B. `realized_profit_total = Σ closed_trades.pnl`), zusätzliches `profit_reserve` als persistierten Wert, plus den `mode` . Die GUI würde diese Werte **nur anzeigen**, nicht selbst berechnen.

---

3. Modell-Vergleich

Aspekt	A) Compound	B) Reserve Profits	C) Partial Reinvest
Status quo?	**JA, implizit**	nein	nein
Investierbarer Pott	`cash` (= start + realized)	`min(cash, base_capital − locked)`	`base + (1−p) × realized_pnl`
Wenn realized=0	`cash = base`	`cash = base`	gleich
Wenn realized > 0	wächst (riskanter)	bleibt base	wächst um `(1−p) × realized`
Wenn realized < 0	schrumpft	schrumpft auch	schrumpft
Drawdown-Risk	hoch (Profits werden geriskt)	niedrig	mittel
Sizing-Komplexität	trivial (status quo)	mittel (1 neue Var)	hoch (1 mode + 1 pct)
Test-Aufwand	(existiert)	mittel	hoch
Mainnet-Tauglichkeit	risikoreich	konservativ	mittel
Testnet-Tauglichkeit	gut für Wachstums-Sim	OK für Profit-Schutz	OK

Wichtige Beobachtung Reserve-Mode: Wenn der Bot Verluste macht, fällt `cash < starting_capital`. Soll dann die Reserve „aufgegessen" werden (= Reserve = max(0, realized_pnl_total))? **Empfehlung: Ja**, Reserve ist nur **realized profit**, nie negativ. Bei Verlust gibt es einfach keine Reserve. Drawdown ist dann separater Mechanismus.

Empfehlung Aufgabe 3

- **Modell B (Reserve Profits Mode) zuerst**, **A bleibt der Compound-Default**.
- Modell C ist Phase 1.1, weil der zusätzliche `pct`-Parameter neue Bounds + Cross-Field-Validation braucht (analog G9 Risk Settings).
- **Default**: für Testnet `compound` (status-quo, kein Behavior-Change), für Mainnet `reserve_profits` (sicherer).

---

4. Begriffe + Formeln (eigener Vorschlag)

Minimaler Begriffs-Satz

Begriff	Bedeutung	Berechnung
`base_capital`	Kapital-Anker, ab dem Profit gezählt wird	persistiert; default = `starting_capital`
`realized_pnl_total`	Summe aller Trade-PnL	`Σ closed_trades[i].pnl`
`reinvestment_mode`	enum	`compound` | `reserve_profits`
`investable_capital`	was Sizer als Pott sieht	siehe Formeln unten

Verworfen als redundant: `current_cash` (= state['cash']), `realized_profit_total` / `realized_loss_total` (split überflüssig, `realized_pnl_total` bereits signed), `net_realized_pnl` (gleich wie realized_pnl_total), `reinvestable_profit` (derived).

Formeln

```
realized_pnl_total = Σ closed_trades[i].pnl                # signed
locked_in_positions = Σ positions[s].quantity × entry_price # cost basis

# Compound-Mode (status quo):
investable_capital_compound = state['cash']

# Reserve-Mode:
profit_reserve = max(0, realized_pnl_total)                # ≥ 0, never negative
investable_capital_reserve = state['cash'] − profit_reserve

# Wenn Verluste angefallen sind:
#   realized_pnl_total < 0 → profit_reserve = 0 → investable = cash (= base − loss)
#   → Bot kann mit verbleibendem Kapital weitertraden, Reserve schützt nichts mehr.
```

Sicherheits-Eigenschaften

- `investable_capital ≥ 0` durch `max(0, …)` und `cash ≥ 0`-Invariante (paper_trade hat keine Negativ-Cash-Logik).
- `investable_capital ≤ cash` immer (Reserve ≤ realized_pnl_total ≤ cash − base + positions_value … fast).
- Kein Doppel-Counting: `investable_capital` wird NUR vom Sizer gelesen; `cash` bleibt für Bookkeeping (Buy/Sell-Mutation) zuständig.
- Reserve bei Verlust: schrumpft automatisch auf 0, kein Negativ-Wert, keine doppelte Drawdown-Wirkung.

---

5. Sizing-Integration

Heutiger Read	Empfehlung
`portfolio['cash']` als available-cash für Sizing	`min(portfolio['cash'], investable_capital_after_reserve)`
`portfolio['total_value']` als sizing-base	`min(total_value, base_capital + locked_in_positions)` für Reserve-Mode; unverändert für Compound
`BalanceGuard.usable_balance` (live exchange)	`min(real_free_balance, investable_capital)`
`cash_reserve_pct` interagiert	bleibt orthogonal — schützt 5% vom **investable**, nicht vom gesamten cash
`max_total_exposure_pct` interagiert	bleibt orthogonal — Ausgangsgröße ist `investable_capital`, nicht `total_value`
`max_open_positions` interagiert	unverändert (Anzahl, nicht Größe)
DCA / T3	T3 hat eigenen Pool (`tier3_budget`); CAP-Mode soll **nicht in T3 eingreifen** in Phase 1

Kritische Regel: Wenn Reserve-Mode aktiv und `exchange_balance > investable_capital`, darf der Bot **nicht** den Exchange-Wert nutzen. `BalanceGuard` capt auf `investable_capital`.

6. GUI-Vorschlag

Empfehlung: Eigener Bereich „Capital Management" in Filament, NICHT in Risk Settings.

Begründung: Risk Settings sind Per-Trade-Limits (max_risk, exposure, etc.). Capital-Management ist ein **architektonischer Mode-Switch** plus Dashboard-Anzeige. Eigener Resource bleibt sauber.

```
Filament Resources:
├── Risk Settings (G9)             ← per-trade gates
├── Config Profiles (G7)            ← G10-Apply-Profile
└── Capital Management (CAP)        ← NEW
    ├── Mode Selector: [compound | reserve_profits | partial_reinvest (1.1)]
    ├── Base Capital (editable, audit-logged)
    ├── Realized PnL (computed, read-only)
    ├── Profit Reserve (computed, read-only)
    ├── Investable Capital (computed, read-only)
    └── History tab: mode-changes mit timestamps + by-user
```

Dashboard-Widget: „Capital Allocation"-Donut: cash / positions_value / profit_reserve.

Feld	Editierbar?	Audit nötig?
`mode`	ja, admin	ja, `capital.mode_changed` event
`base_capital`	ja, admin	ja, `capital.base_changed` event
`realized_pnl_total`	nein (derived)	n/a
`profit_reserve`	nein (derived)	n/a
`investable_capital`	nein (derived)	n/a

G10 Apply-Required? Mode-Wechsel ist substantieller Behavior-Change → **JA**, sollte über Apply-Mechanik laufen (analog G9). Konkret: `mode` als 14. Phase-1.1-Key in G7-Allowlist UND/ODER neue G9-Allowlist-Erweiterung.

7. G10 / Runtime Config — Empfehlung

Frage	Antwort
Gehört in G10 Runtime Config?	**Ja**, mode + base_capital sollten via runtime_config.json apply-bar sein
In G9 Risk Settings?	**Nein** — semantisch eigener Bereich; eigene Allowlist `Phase1CapitalAllowlist` (analog Phase1RiskAllowlist)
In G7 Summary?	**Nein** — Mode ist mehr als ein „verbosity"-Feld
runtime_config.json soll enthalten?	`{"capital": {"mode": "compound", "base_capital": 10000.0}}` als 3. Top-Level-Block neben `summary + risk`
Bot-Read-Sites bei Apply	(a) BalanceGuard (read mode + reserve) (b) Sizer (use investable_capital statt cash) (c) Dashboard (anzeigen)
Wann wirkt Wechsel?	**Per-Cycle-Snapshot** (analog G10-4.1 Etappe 2) — nächster Cycle nach Apply

Audit-Schicht: jeder Mode-Wechsel schreibt `audit_events.event_type='capital.mode_apply_requested'` analog `profile.apply.requested`.

---

8. T-SPLIT-Auswirkung

Frage	Antwort
Profit-Reinvest **global** oder per T-Bereich?	**Beides möglich** — empfohlen Phase 1: global; Phase 2: per-T-bereich
T3 hat schon eigenen Pool	richtig (`tier3_budget` / `_used` / `_positions`) — CAP soll T3 zunächst NICHT überschreiben
Pump & Dump (T2) Reserve	empfehlenswert, da T2 hochvolatil → eigener Reserve-Modus später
Copy Trading (T3)	**niemals automatisch reinvestieren** — empfohlener Hard-Default `reserve_profits` für Copy

Phase-Reihenfolge: CAP-1 → T-SPLIT-Detection. Nach T-SPLIT kann CAP-2 (per-T-bereich-Pools) folgen.

CAP-1 / T-SPLIT sind **orthogonal**. Reihenfolge offen.

---

9. Safety / Defaults

Setting	Empfehlung
Sicherer Default (Mainnet)	**`reserve_profits`**
Sinnvoller Default (Testnet)	**`compound`** (status quo, kein Behavior-Change)
Mainnet-Hard-Restrict?	**Compound nur mit explizitem Admin-Apply** in Mainnet (zusätzliche Bestätigung im Filament)
Max-Profit-Reinvest-Cap?	**JA** für `partial_reinvest` (Phase 1.1) — `pct ≤ 0.50` als upper bound
Aufhebeln verhindern	invariant `investable_capital ≤ cash` durchgängig — geprüft via Test
Initial-Activation	`mode = compound` (default), `base_capital = starting_capital`, `profit_mode_activated_at = NOW()`

---

10. Migration / Backfill

Frage	Empfehlung
Historische closed_trades nutzen?	**NUR FÜR REPORTING**, nicht für rückwirkenden Reserve-Aufbau
Rückwirkend reservieren?	**Nein** — gefährlich (verzerrt Audit-Trail)
Ab Aktivierung starten	**JA** — `profit_mode_activated_at` Marker setzen, alte Trades als „pre-cap" reportieren
`base_capital_snapshot`	**JA** — beim Activate `base = current_cash + Σ position[i].entry_value` (= Buchwert)
Bestehende offene Positionen	bleiben unverändert, werden bei nächstem Sell mit aktuellem Modus abgewickelt

---

11. Phasenplan (eigener Vorschlag)

Phase	Scope	LoC	Tests	Risiko
CAP-1-PREFLIGHT	dieser Audit (jetzt)	0	0	none
CAP-1.1	`realized_pnl_total` + `base_capital` + `profit_reserve` + `investable_capital` als Computed-Properties in `PaperTrader.get_portfolio()`. **Bot-only**, kein Sizer-Effekt.	~80	~10	niedrig (read-only Erweiterung)
CAP-1.2	Sizer-Migration: `cash` → `investable_capital` analog G10-4.1 Etappe 2. Mode = compound (status-quo) → kein Behavior-Change.	~150	~15	mittel (Sizer-Touch)
CAP-1.3	`Phase1CapitalAllowlist` PHP-side + `runtime_config.json["capital"]` Erweiterung. Writer + Provider erweitern (mode, base_capital).	~200	~20	mittel
CAP-1.4	Filament „Capital Management" Resource + Dashboard-Widget. Read-only Anzeige.	~250	~25	niedrig (UI)
CAP-1.5	G10-Apply-Integration: `command_type='apply_capital_mode'` analog `apply_profile_testnet`. Live-E2E Dry-Run + Apply.	~300	~30	hoch (Apply-Phase)
CAP-1.6 (optional)	partial_reinvest mode mit reinvestment_pct + Bounds (0–0.5)	~100	~15	mittel

Gesamt: ~5–6 Etappen, jede 1–2 Sessions, ~1080 LoC + ~115 Tests.

Empfohlene Position in Roadmap: NACH G10 Closure, **vor T-SPLIT**. Begründung: G10-Apply-Mechanik ist Voraussetzung für CAP-1.5; T-SPLIT braucht globalen CAP-Mode als Default, bevor per-T-Pools zugeschaltet werden.

---

12. Testplan

Test-Kategorie	Anzahl	Schwerpunkt
Unit: Capital-Formeln	~10	`realized_pnl_total`, `profit_reserve`, `investable_capital`
Compound-Mode	~5	status-quo Verhalten, keine Regression
Reserve-Mode	~10	Sizer cap auf `investable_capital`; Reserve-Schutz
Partial-Mode (1.6)	~10	pct-Bounds, edge cases
Negative PnL	~5	reserve = 0, kein Negativ-Spiral
Profit-nach-Verlust	~3	base_capital-Anker bleibt stabil
Bestehende Positionen bei Activation	~3	locked_in_positions korrekt subtrahiert
`investable < exchange_cash`	~3	Reserve cap greift
`investable > exchange_cash`	~3	exchange-Limit greift, kein Bot-Double-Spend
`cash_reserve_pct` interaction	~3	bleibt orthogonal
`max_total_exposure_pct` interaction	~3	bleibt orthogonal
Dashboard-Anzeige	~5	Zahl-Korrektheit
Audit-Events	~5	mode-Wechsel logged
G10-Apply (CAP-1.5)	~10	Dry-Run + real Apply
Boundary AST	~5	keine Orders, keine `live_portfolio.json` Mutation in Preflight

Total: ~95–115 Tests über alle CAP-Phasen.

13. Risiken + Stop-Regeln

Risiko	Mitigation
Falsches Sizing nach Mode-Switch	Per-Cycle-Snapshot; alter Cycle bleibt im alten Mode konsistent
Doppelte Kapitalzählung	Invariant-Test: `investable_capital ≤ cash` ; Zentralisierung in `PaperTrader.get_portfolio()`
Gewinne aus Versehen riskiert	Reserve-Mode default für Mainnet + Apply-Confirm-Dialog
Reserve aus Versehen für Drawdown	Reserve never < 0; Drawdown-Check separater Pfad (risk_manager)
Dashboard zeigt falsche Zahlen	Computed-Properties zentral in get_portfolio(), Tests pro Feld
Backfill verfälscht Historie	KEIN Backfill — Activation-Marker setzt Schnittlinie
G10-Apply ändert Kapitalbasis unerwartet	Apply-Confirm-Dialog (Filament v4 Modal) + 24h-Cooldown
T1/T2/T3 Pools mischen	Phase-1: nur global; Per-T-Pools erst nach T-SPLIT
Mode-Switch mid-cycle Race	Snapshot pattern (analog G10-4.1)

Stop-Regeln:

- Falls `investable_capital < 0` rechnerisch → Bot abort + Audit
- Falls `realized_pnl_total != Σ closed_trades.pnl` → Bot Warning + Reconcile
- Falls Mode-Wechsel ohne Audit-Event → Test-Failure (boundary AST)

---

14. Eigene Empfehlung — Kurzfassung

1. **CAP-1 ist sinnvoll und gehört nach G10-Closure**, weil G10-Apply-Mechanik Voraussetzung für saubere Mode-Wechsel ist.
2. **Eigene CAP-Serie**, nicht in G9 oder G7 einbauen — semantisch klar getrennt.
3. **Modell B (Reserve Profits) zuerst** als optional-aktivierbarer Mode, **Compound bleibt Default** für minimal-invasive Migration.
4. **Architektur-Schwerpunkt:** neue Computed-Properties in `PaperTrader.get_portfolio()` + Sizer-Migration `cash → investable_capital` (analog G10-4.1 Etappe 2).
5. **Phasen:** CAP-1.1 bis CAP-1.5 (oder 1.6 mit partial), gleiches Etappen-Muster wie G10-4.x.
6. **Default Testnet:** compound (status-quo). Default **Mainnet: reserve_profits + admin-confirm-required**.
7. **T-SPLIT-Interaktion:** orthogonal — CAP-1 global zuerst, per-T-bereich-Pools später (T-SPLIT ist Vorbedingung).
8. **Backfill ablehnen** — Activation-Marker setzt Schnittlinie.

---

15. Offene Fragen (für späteren CAP-1-Start)

- **Q-1:** Soll `base_capital` an `STARTING_CAPITAL` (.env) gekoppelt sein, oder unabhängig editierbar?
- **Q-2:** Soll T3 in CAP-1 explizit ausgeklammert werden (eigener Pool)? Oder T3 auch im global-Pool?

- **Q-3:** Bei Mode-Switch mit offenen Positionen: alle bestehenden Positions in „pre-CAP" Bucket markieren? Oder bei nächstem Sell wie-bisher behandeln?
- **Q-4:** Soll `realized_pnl_total` auch fees abziehen (= net) oder nur PnL? Aktuell `closed_trades.pnl` ist already-netted (siehe paper_trade.py:208).
- **Q-5:** Reserve-Wert als **separates Feld** in `live_portfolio.json` persistieren, oder **always derived** aus `closed_trades`? (Performance: 32 Trades sind trivial; bei 10 000 Trades wird Recompute teuer.)

Boundary-Bestätigungen

Check	Status
Read-only	ja — nur Code/State/DB lesen
Code-Änderung	nein
Tests geändert	nein
Migration	nein
`live_portfolio.json` Mutation	nein
`runtime_config.json` Mutation	nein (nicht existent)
`.env` Mutation	nein, mtime `1777991334` unverändert
Bot-Restart	nein, PID 4246 stable
Worker	nein
Orders	nein
Mainnet	nein, BINANCE_TESTNET=true
Push	nein

Empfohlene Roadmap-Position: Post-G10 Backlog Item #3 (zwischen T-SPLIT-PREFLIGHT und D-DEPLOY).