G10-3c — Scope-Bestätigung

Steve Trading Bot — Live-E2E Dry-Run-Verifikation des apply_profile_testnet-Handlers

Datum: 2026-05-09

Master-Tip: `62cf7c7`

Bot-PID: 14381

Test-Profile: `b4033597-0f90-4c6a-9b55-2a31648f10d5` (G10-3a)

Status: vor Implementierung — wartet auf GO mit Antwort auf Q1.

Befund vor Scope: Payload-Schema-Drift

Beim Audit der PHP-side `ApplyProfileService::createDryRunCommand()` habe ich Folgendes gefunden:

```
PHP baut Payload mit:
  $payloadCore = [
      'profile_id'             => ...,
      'profile_version'        => (int) $profile->version,   // <- profile_version
      'requested_by'           => $userId,
      'dry_run'                => true,
      'environment'            => 'testnet',
      'exchange_connection_id' => null,
      'config_summary'         => $normalisedConfig,         // extra Daten
      'risk_settings'          => $normalisedRisk,           // extra Daten
  ];
  $checksum = PayloadCanonicaliser::checksum($payloadCore);
```

Bot-Handler aus G10-3b erwartet:

```
ver = payload.get("version")  # <- "version", NICHT "profile_version"
```

Live-E2E würde fehlschlagen mit `validation/missing_or_bad_version`. Spec-Drift zwischen User-Spec (G10-3b sagte „version") und der bestehenden PHP-Implementation (sagt „profile_version" seit G10-2).

Lösungsoptionen (vor G10-3c-Live-Run zu entscheiden)

Option	Action	Risiko	Empfehlung
A	Bot-Handler akzeptiert beide: `payload.get("profile_version") or payload.get("version")`	minimal — 1 Zeile + 1 Test	empfohlen
B	Bot-Handler von `version` -> `profile_version` umbenennen	Test-Aktualisierungen nötig	klar, aber mehr Test-Drift
C	PHP-Service von `profile_version` -> `version` umbenennen	berührt G10-2 + Audit-Schema	schlechte Idee, retro-aktiv

Empfehlung **A** — Bot-Handler defensiv erweitern (1 Zeile, 1 neuer Test), bevor Live-Run.

Geplanter Ablauf G10-3c

Vorbereitungs-Mini-Phase G10-3c-pre (falls Option A gewählt)

1. Edit `command_worker.py:_g10_3b_validate_payload_basic` — `payload.get("profile_version") or payload.get("version")`
2. Add 1 Test in `test_g10_3b_apply_profile_testnet.py`: `test_payload_profile_version_field_accepted`
3. Re-run G10-3b suite (42/42)
4. Commit kleines diff

G10-3c Hauptphase

```
Stage 1 - Code-Sync:
  docker cp /projekte/.../command_worker.py clawbot:/home/.../command_worker.py
  Hash-Verify host vs container
  Python-Import-Sanity

Stage 2 - Pre-State Capture:
  - commands count, command_audit_log count, audit_events count
  - bot_statuses count
  - live_portfolio.json sha256
  - runtime_config.json existence (false expected)
  - .env mtime
  - Bot main.py PID 14381 + cmdline
  - Worker container/PID (false expected)

Stage 3 - Command-Erzeugung via PHP-Service:
  docker exec steve-tradingbot-gui php -r "
    require '/var/www/html/vendor/autoload.php';
    $app = require '/var/www/html/bootstrap/app.php';
    $app->make(Illuminate\\Contracts\\Console\\Kernel::class)->bootstrap();
    $svc = app(App\\Services\\Apply\\ApplyProfileService::class);
    $profile = App\\Models\\ConfigProfile::where('profile_id','b4033597-...')->firstOrFail();
    $cmd = $svc->createDryRunCommand($profile, 23);
    echo 'cmd_id=' . $cmd->command_id . ' status=' . $cmd->status . PHP_EOL;
  "
  Erwartet:
  - 1 neue commands row (status=pending)
  - 1 neue audit_events row (event_type=profile.apply.dry_run_requested)

Stage 4 - Worker one-shot:
  docker exec -e GUI_DB_HOST=gui-db -e GUI_DB_NAME=tradingbot_gui \
              -e GUI_DB_USER=tradingbot_gui -e GUI_DB_PASSWORD=tradingbot_gui \
              -e WORKER_ID=g10-3c-test-worker \
              -e COMMAND_BUS_VERSIONS_PATH=/home/.../gui/command_bus_versions.json \
    clawbot python3 -m trading.command_worker --once
  Erwartet:
  - eine Iteration, claim+execute, exit 0
  - kein Daemon

Stage 5 - Post-Verify:
  - command status = succeeded
  - command result_json: ok=true, dry_run=true, validated=true, would_apply=false
  - command error_message: NULL
  - command_audit_log >=4 events fuer diese command_id:
      "started" (Framework)
      "apply_profile_testnet.dry_run_started" (Handler)
      "apply_profile_testnet.dry_run_succeeded" (Handler)
      "result_written" (Framework)
  - live_portfolio.json sha256: UNVERAENDERT
  - runtime_config.json: existiert NICHT
  - .env mtime: UNVERAENDERT
  - Bot main.py PID 14381: UNVERAENDERT
  - Worker container: nicht aktiv (one-shot exit)

Stage 6 - Cleanup (passiv):
  - Test-command bleibt durable
  - audit_events bleiben
  - command_audit_log rows bleiben
  - kein Rollback noetig
```

1. Worker-Ausführungsmodus

Bestehend, sicher: `python3 -m trading.command_worker --once`

Modus	CLI	Verhalten
One-shot (G10-3c)	`--once`	claim_next() + execute() einmalig, exit 0. Kein Daemon.
Loop (Out-of-Scope)	(kein Flag) oder `--max-iterations N`	endlose Schleife — ungewollt für G10-3c

G10-3c nutzt `--once` ausschließlich. Kein `docker compose --profile workers up -d clawbot-worker`.

2. Command-Erzeugungs-Pfad

Bevorzugt: PHP-Service `ApplyProfileService::createDryRunCommand($profile, $userId)` via inline Artisan-Skript (gleicher Pattern wie G10-3a-Profile-Anlage).

Vorteile:

- volle PHP-side Validation (Layers 1–9): Phase1Allowlist, Phase1RiskAllowlist, Cross-Field, SecretPatternFilter
- echter `PayloadCanonicaliser::checksum()` (sha256 über kanonisches JSON)
- echtes `idempotency_key` Pattern
- 1 audit_events Eintrag (profile.apply.dry_run_requested)

Alternative (NICHT bevorzugt): raw `INSERT INTO commands (...)` — würde PHP-Validation überspringen + falschen checksum erzeugen.

3. Payload (PHP-Service-erzeugt)

```
{
  "profile_id":             "b4033597-0f90-4c6a-9b55-2a31648f10d5",
  "profile_version":        1,
  "requested_by":           23,
  "dry_run":                true,
  "environment":            "testnet",
  "exchange_connection_id": null,
  "config_summary": {
    "log_level": "info",
    "decision_log_verbosity": "full"
  },
  "risk_settings": {
    "max_risk_per_trade_pct":   0.005,
    "max_total_exposure_pct":   0.20,
    "max_open_positions":       4,
    "daily_loss_limit_pct":     0.02,
    "weekly_loss_limit_pct":    0.05,
    "tier_low_allocation_pct":  0.01,
    "tier_mid_allocation_pct":  0.03,
    "tier_high_allocation_pct": 0.05,
    "cash_reserve_pct":         0.10,
    "fee_buffer_pct":           0.002,
    "min_position_value_usdt":  10
  },
  "checksum":   "<sha256 hex>",
  "expires_at": "<ISO8601 +5min>"
}
```

`config_summary` + `risk_settings` werden vom Bot-Handler nicht validiert (Bot liest aus DB für Source-of-Truth — defensiv gegen Payload-Tampering).

4. Vorher/Nachher-Checks

Metrik	Quelle	Pre	Post (erwartet)
`commands` count	gui-db	1	2 (+1)
`command_audit_log` count	gui-db	4	>=8 (+4)
`audit_events` count	gui-db	2	3 (+1)
`config_profiles` count	gui-db	1	1 (unverändert)
`risk_settings` count	gui-db	11	11 (unverändert)
`bot_statuses` Wachstum	gui-db	growing	weiter wachsend (G6.1)
`live_portfolio.json` sha256	clawbot	X	X (unverändert durch G10-3c)
`runtime_config.json` existiert	clawbot	NEIN	NEIN
`.env` mtime	host	`1777991334`	`1777991334`
Bot main.py PID	clawbot	14381	14381
Worker daemon-Container	docker	nicht aktiv	nicht aktiv
`command_worker.pid` Datei	clawbot	nicht vorhanden	kann existieren (Diagnose), Prozess tot

5. Erwarteter Zustand nach G10-3c

```
1 neue commands row:
  command_type = "apply_profile_testnet"
  status       = "succeeded"
  result_json  = {"ok":true,"dry_run":true,"validated":true,
                  "profile_id":"b4033597-...","version":1,
                  "validated_summary_keys_n":2,"validated_risk_keys_n":11,
                  "would_apply":false,"runtime_mutation":false,
                  "handler_phase":"G10-3b"}
  error_message = NULL

4 neue command_audit_log rows (chronologisch):
  1. event_type = "started"  (claimed -> running)            [Framework]
  2. event_type = "apply_profile_testnet.dry_run_started"    [Handler]
  3. event_type = "apply_profile_testnet.dry_run_succeeded"  [Handler]
  4. event_type = "result_written"  (running -> succeeded)   [Framework]

1 neue audit_events row:
  event_type = "profile.apply.dry_run_requested"  (PHP-Service)

KEINE Mutation:
  - kein live_portfolio.json Write
  - keine runtime_config.json
  - keine .env-Mutation
  - keine Orders
  - kein clawbot-worker Daemon
  - Bot main.py weiter unverändert
```

6. Stop-Regeln

Bedingung	Action
Bot main.py PID 14381 ändert sich	sofort STOP, Forensik
live_portfolio.json hash drift unerklärbar (nicht Bot)	sofort STOP
`runtime_config.json` wird erzeugt	sofort STOP — G10-3 darf das NICHT
`.env` mtime ändert sich	STOP
Worker verarbeitet >1 Test-Command	STOP — `--once` sollte nur 1
Command bleibt failed mit unerwartetem reason	STOP, Forensik
Orders/ccxt-Calls im Worker-Log sichtbar	STOP — AST-Test sollte das verhindert haben
Mainnet-Pfad sichtbar (BINANCE_TESTNET=false)	STOP

7. Cleanup/Rollback-Plan

Default: kein Rollback nötig — alle G10-3c-Aktionen sind read-only-only-write-to-audit, keine bot-state-Mutation.

Artefakt	Rollback bei Bedarf
1 neue commands row	optional `DELETE FROM commands WHERE command_id = '<uuid>'` (kaskadiert auch command_audit_log via FK CASCADE) — aber besser nicht löschen, audit-trail behalten
audit_events row	nicht löschen — durabler Beweis
`command_worker.pid` Datei	optional `docker exec clawbot rm logs/command_worker.pid`
Worker-Prozess (one-shot)	beendet sich von selbst nach exit

Bei kritischem Stop: G10-3c stoppt vor Worker-Start, command bleibt pending. `DELETE FROM commands WHERE command_id = '<uuid>'` löscht ihn sauber zurück.

8. Open Questions (vor G10-3c)

#	Frage	Vorschlag
Q1	Payload-Schema-Drift `profile_version` vs `version`	Option A: Bot-Handler akzeptiert beide via fallback (1-Zeilen-Fix + 1 Test in einem G10-3c-pre-Commit)
Q2	Worker one-shot vs daemon	One-shot (`--once`) for G10-3c; daemon-Aktivierung als separate G10-3d-Phase
Q3	Command-Erzeugung über PHP-Service vs raw INSERT	PHP-Service — gleicher Pattern wie G10-3a-Profile-Anlage
Q4	Test-command in DB lassen oder löschen	lassen — durable audit-trail; kein Cleanup nötig

9. Hard-Boundaries für G10-3c (durable)

```
KEIN dry_run=false       (Handler weist es ab; PHP-Service erzeugt nur dry_run=true)
KEIN G10-4
KEIN Runtime-Config-Switch
KEIN live_portfolio.json Write
KEIN runtime_config.json Write
KEINE Orders
KEINE .env-Mutation
KEIN Mainnet
KEIN GUI-Apply-Enable
KEIN Rollback (G10-5)
KEIN Push (Repo hat kein Remote)
KEIN Daemon-Worker (nur --once)
```

Stop

Scope-Bestätigung mit Payload-Schema-Drift-Befund als Q1.

Wartet auf:

- **GO G10-3c-pre** -> Bot-Handler erweitert um Doppel-Field-Akzeptanz, dann Live-E2E
- ODER **GO G10-3c direkt** + Variante (z.B. testen wie es ist, dann sehen wir das Mismatch-Failure live)
- ODER **Anpassung X**

Stand: 2026-05-09 07:10 UTC. Vor Implementierung. Erfordert explizites GO.