G10-4.2 Scope-Bestätigung — Lieferbericht

Datum: 2026-05-09 (UTC 12:05)
Status: Scope-Phase, kein Code, kein Apply, kein Worker. Read-only Planung.

Vorgeschlagener Scope

G10-4.2 lifted die `dry_run=false`-Reject-Bedingung im Bot-Handler `_handle_apply_profile_testnet` und verdrahtet erfolgreiche Validation mit `RuntimeConfigWriter.apply()`. Der laufende Bot übernimmt den Override beim nächsten Cycle-Snapshot via bestehender G10-4.1-Etappe-2-Infrastruktur. **Keine** weiteren Bot-Code-Pfade ändern sich.

Empfohlene Teilung: G10-4.2a / G10-4.2b

Etappe	Inhalt	Live-Action	Backup-Pflicht
G10-4.2a	Bot-Handler-Branch + Idempotency-Helper im Writer + neue Tests; isolierter Test-Run; Commit	nein	nein (rein lokal)
G10-4.2b	Backup-Checkpoint + Container-Sync `command_worker.py` (+ ggf. Writer) + Worker `--once` mit echtem `dry_run=false`-Command + Post-Verify (1–2 Cycles bis Bot-Heartbeat den Override widerspiegelt)	ja	**ja, Pflicht**

Begründung: Dieselbe Aufteilung wie G10-3b/c. G10-4.2a kann lokal getestet, gereviewt, gecommittet werden ohne irgendeinen Live-Effekt. G10-4.2b ist ein eigener kontrollierter Aktivierungs-Schritt.

Datei-Plan

Datei	Änderung	Umfang
`trading/command_worker.py`	`_handle_apply_profile_testnet`: `dry_run=false`-Pfad nach Validation-Cascade ergänzen. Aufruf `RuntimeConfigWriter.apply(...)`, Idempotenz-Check, Result-Composition.	~120 LoC + ~30 LoC für 2 neue Helper
`trading/runtime_config_writer.py`	Optional: neue Method `apply_idempotent(...)` oder Idempotency-Pre-Check als Helper-Function im Writer-Modul. Gibt `ApplyResult(ok=True, no_effect=True)` zurück, wenn Inhalt bereits identisch.	~40 LoC + ~5 Tests
`trading/tests/test_g10_4_2_apply_runtime_config.py`	**NEU** — 30+ Tests	~700 LoC
`trading/tests/test_g10_3b_apply_profile_testnet.py`	**Update** — bestehender Test `test_dry_run_false_rejected` muss angepasst werden. Plus neue Tests, dass dry_run=true unverändert funktioniert.	~50 LoC
`trading/tests/run_g10_4_2_tests.sh`	**NEU** — Sibling-Script analog G10-3b-Pattern	~80 LoC

Bot-Side `main.py` / `risk_manager.py` /etc.: NICHT angefasst. PHP-Side: NICHT angefasst (G10-3.5 hat `createApplyCommand` bereits gebaut).

Handler-Flow (G10-4.2)

```
_handle_apply_profile_testnet(claim, payload):
    audit_emit("apply_started")  # bestehender state-transition-event

    # 1. Validation cascade (UNCHANGED from G10-3b/c):
    ctx = {...}
    _g10_3b_validate_payload_basic(payload, ctx)
    profile, risk_rows = _g10_3b_load_profile(...)
    _g10_3b_validate_profile_state(profile, payload, ctx)
    _g10_3b_validate_schema(payload, ctx)

    # 2. Branch on dry_run:
    if payload["dry_run"] is True:
        return _g10_3b_dry_run_succeed(...)  # G10-3b path UNCHANGED

    # 3. NEW: dry_run=false path
    # Idempotency pre-check via canonical sha256:
    new_sha = sha256(canonical_json(summary, risk))
    existing_sha = read_existing_target_sha256()
    if existing_sha == new_sha:
        audit_emit("apply_succeeded", {**ctx, "no_effect": True})
        return {"ok": True, "no_effect": True, ...}

    # 4. Mainnet-Guard: Writer enforces L1 (BINANCE_TESTNET=True);
    #    handler does not duplicate, but catches MainnetBlockedError.

    # 5. Call writer:
    apply_result = writer.apply(summary, risk, meta)

    # 6. ApplyResult handling:
    if not apply_result.ok:
        audit_emit("apply_failed")
        raise ValueError(f"apply_writer_failed: {apply_result.error}")

    # 7. Optional post-write verify (re-read sha256):
    if sha256(target) != apply_result.sha256:
        restore_from_backup(...)
        audit_emit("apply_rolled_back")
        raise ValueError("apply_post_write_sha_mismatch")

    # 8. Success — split apply_effective vs stored_only:
    audit_emit("apply_succeeded", {
        ...ctx,
        "applied_keys":         all_13_keys,
        "apply_effective_keys": [9 keys per ActiveConfigProvider.APPLY_EFFECTIVE],
        "stored_only_keys":     [4 keys = log_level, decision_log_verbosity, daily/weekly_loss_lir
        "previous_values":      old_runtime_config_or_empty,
        "new_values":           {summary, risk},
        "runtime_config_path":  apply_result.target_path,
        "backup_path":          apply_result.backup_path,
        "sha256":               apply_result.sha256,
        "runtime_mutation":     True,
        "would_apply":          True,
        "handler_phase":        "G10-4.2"
    })

    return {
        "ok": True, "dry_run": False, "validated": True,
        "would_apply": True, "runtime_mutation": True,
        "applied_keys": ..., "runtime_config_path": ...,
        "backup_path": ..., "sha256": ...,
        "handler_phase": "G10-4.2", "elapsed_ms": ...
    }
```

Mainnet-Guard (5-Layer Defense-in-Depth, durable)

Layer	Check	Implementiert in
L1	`Settings.BINANCE_TESTNET is True`	`RuntimeConfigWriter._assert_testnet()` (G10-4.1 Etappe 1) — bereits aktiv
L2	`BOT_ENVIRONMENT == "testnet"`	`command_worker.ALLOWED_ENVIRONMENTS["apply_profile_testnet"]={"testnet"}` (G10-3b)
L3	`command.payload.environment == "testnet"`	`_g10_3b_validate_payload_basic` (G10-3b)
L4	`profile.environment == "paper"`	`_g10_3b_validate_profile_state` (G10-3b)
L5	`exchange_connection_id is None`	`_g10_3b_validate_payload_basic` (G10-3b)

LIVE_TRADING_ENABLED ist NICHT Teil der Mainnet-Sperre — durable rule per LIVE_TRADING_ENABLED-Audit. Optional als Layer L6 für Bot-Run-State-Sanity-Check.

Handler ruft KEINEN expliziten BINANCE_TESTNET-Check auf — der Writer (Layer L1) raised `MainnetBlockedError`, Handler surfaced den Reject-Token.

Payload / Apply-Keys-Handling

Aspekt	Verhalten
Welche Keys werden geschrieben?	**alle 13** (2 Summary + 11 Risk) — wenn im Payload präsent
Apply-Effective (9)	`max_open_positions`, `max_risk_per_trade_pct`, `max_total_exposure_pct`, `min_position_value_usdt`, `cash_reserve_pct`, `fee_buffer_pct`, `tier_low/mid/high_allocation_pct`
Stored-only / Phase-1.1 (4)	`log_level`, `decision_log_verbosity`, `daily_loss_limit_pct`, `weekly_loss_limit_pct`
Source-of-Truth für Split	`ActiveConfigProvider.APPLY_EFFECTIVE` Map (G10-4.1 Etappe 1)
Result + Audit	`applied_keys`, `apply_effective_keys`, `stored_only_keys` separat ausgewiesen
Validierung	`_g10_3b_validate_schema` durchläuft alle 13 Keys gleich

Wenn Payload nur Teilmenge enthält → Writer schreibt nur diese; nicht-präsente Keys bleiben in `runtime_config.json` ungetouched.

Audit-Events (Shape)

Event	Trigger	context_json wesentliche Felder
`apply_started`	Handler-Entry (gleicher Event-Type für dry-run + apply, `dry_run` -Flag in metadata)	profile_id, version, dry_run, environment, handler_phase
`apply_succeeded`	nach erfolgreichem Writer-Call ODER no-op-Idempotency	15 Felder inkl. previous_values, new_values, runtime_config_path, backup_path, sha256, apply_effective_keys, stored_only_keys, runtime_mutation, no_effect (falls idempotent)
`apply_failed`	Validation-Error / Writer-Error / Mainnet-Block	profile_id, version, dry_run=false, reason, error_detail
`apply_rolled_back`	post-write sha-mismatch ODER Restore-Versuch	profile_id, version, backup_path, restore_status, original_error

Plus existierende state-machine-Events (claimed, started, result_written) aus G6.5.

Writer- / Rollback-Verhalten

Fehler-Stadium	Verhalten
Validation-Fail vor Writer-Call	kein Writer-Call, kein Backup, kein Write. command_status=failed, audit apply_failed mit reason=validation_*. runtime_config.json bleibt unverändert.
MainnetBlockedError (Writer L1)	wie oben, plus reason=mainnet_blocked.
Writer-IO-Fehler vor `os.replace`	Writer cleant tmp-File, Backup bleibt erhalten falls erstellt, Target unverändert (POSIX-atomic). ApplyResult(ok=False). Handler: command_status=failed.
Writer-IO-Fehler nach Backup	wie oben — Target unverändert, Backup als Restore-Punkt verfügbar.
Post-Write-Verify-Mismatch (extrem unwahrscheinlich)	Handler ruft restore_from_backup, audit apply_rolled_back, command_status=failed.
Backup-Copy-Fail	ApplyResult(ok=False, error="backup_copy_failed"). Handler: failed, kein Write.

Writer auto-restored NICHT (G10-4.1 Etappe 1 dokumentiert). Restore ist Handler-Verantwortung im post-Write-Verify-Pfad.

Idempotenz-Entscheidung

Vorschlag: Pre-Write SHA256-Compare.

```
new_sha = sha256(json.dumps({"summary": payload_summary,
                             "risk": payload_risk},
                            indent=2, sort_keys=True))
existing_sha = sha256(target.read_bytes()) if target.is_file() else None

if existing_sha == new_sha:
    return ApplyResult(ok=True, no_effect=True,
                       target_path=target, backup_path=None,
                       sha256=existing_sha, error=None)
```

Effekt:
- Identischer Re-Apply → ok=True, no_effect=True, **kein Backup**, **kein Write**.
- Audit-Event apply_succeeded mit no_effect=true — auditierbar, kein Failure.
- Backup-Files füllen sich nicht.

Caveat: `_meta` aus Idempotency-Compare ausschließen (timestamp/command_id würden sonst jedes Apply als „verschieden" markieren). Compare nur über `summary` + `risk`.

Implementation-Ort: im Writer (neue Method `apply_idempotent` ODER Pre-Check in bestehender `apply`). Cleaner: bestehende `apply` mit pre-check ergänzen.

Testplan (≥30 Tests in 6 Klassen)

Dry-run Regression

- existierende G10-3b-Suite läuft ohne Anpassung weiter
- `test_dry_run_true_path_unchanged` (idempotent re-asserted)
- `test_dry_run_false_was_rejected_now_accepted` — UPDATE existing test

Apply-Success

- `test_apply_writes_runtime_config_json`
- `test_apply_result_shape_correct` (alle 11 result_json keys)
- `test_apply_audit_started_then_succeeded`
- `test_apply_effective_keys_split_correct` (9 vs 4)
- `test_apply_meta_includes_command_id_profile_version_checksum`
- `test_apply_creates_backup_when_target_exists`
- `test_apply_no_backup_on_first_write`
- `test_apply_sha256_present_and_matches_disk`
- `test_apply_runtime_config_path_is_state_dir`

Mainnet / Environment Guards

- `test_apply_blocked_when_BINANCE_TESTNET_false`
- `test_apply_blocked_when_BINANCE_TESTNET_missing`
- `test_apply_blocked_when_BINANCE_TESTNET_string_true_not_bool`
- `test_apply_NOT_blocked_when_LIVE_TRADING_ENABLED_false` ← positiv-Beweis
- `test_apply_blocked_when_BOT_ENVIRONMENT_paper`
- `test_apply_blocked_when_payload_environment_not_testnet`
- `test_apply_blocked_when_profile_environment_not_paper`
- `test_apply_blocked_when_exchange_connection_id_not_null`

Writer / Rollback

- `test_writer_io_error_command_failed`
- `test_writer_io_error_target_unchanged`
- `test_writer_io_error_no_tmp_left_behind`
- `test_writer_validation_error_before_write`
- `test_post_write_verify_mismatch_triggers_rollback`
- `test_apply_rolled_back_audit_emitted`

Idempotenz

- `test_idempotent_apply_returns_no_effect`
- `test_idempotent_apply_no_new_backup`
- `test_idempotent_apply_audit_succeeded_with_no_effect_flag`
- `test_meta_timestamp_does_NOT_break_idempotency`
- `test_different_summary_creates_new_apply_not_idempotent`

Boundary

- `test_no_live_portfolio_writes_in_handler` (AST)
- `test_no_env_writes_in_handler` (AST)
- `test_no_orders_or_ccxt_in_handler` (AST)
- `test_no_worker_daemon_started`
- `test_no_gui_apply_button_enabled`

Integration

- `test_full_apply_flow_with_g10_3a_test_profile`
- `test_apply_then_dry_run_with_same_profile_independent`
- `test_concurrent_apply_command_lock_works`

Total: ~33 Tests, plus AST-Boundary-Tests.

Live-E2E-Abgrenzung (G10-4.2b)

Aspekt	G10-4.2a	G10-4.2b
Code-Änderung	ja	nein
Tests laufen	im throwaway Container	nein (Live-E2E)
Container-Sync	nein	ja, `command_worker.py` (+ ggf. Writer)
Backup vorab	nein	**ja, Pflicht**
PHP-Side `createApplyCommand`	nein	ja (G10-3.5 wird live verwendet)
Worker `--once`	nein	ja, einmalig mit echtem dry_run=false-Command
`runtime_config.json` wird geschrieben	nein	**ja** — erstmals real
Bot-Override greift	nein	**ja**, innerhalb 1–2 Cycles via ActiveConfigSnapshot
Bot-Restart	nein	nein (Override greift cycle-snapshot)
Test-Profile	nein	ja, gleiches Profile wie G10-3c

Risiken

ID	Risiko	Mitigation
R1	dry_run=true Regression	volle G10-3b-Suite muss grün bleiben; Test `test_dry_run_true_path_unchanged`
R2	runtime_config.json falscher Inhalt	post-write-sha256-verify; canonical-json (sort_keys); test
R3	Writer schreibt Keys außerhalb Validation	Writer-Whitelist (Etappe 1); M2 AST-Test (Worker↔Writer-sync)
R4	stored_only Keys fälschlich apply_effective	source-of-truth `APPLY_EFFECTIVE` ; Test
R5	Mainnet-Guard falsch	5-Layer Defense-in-Depth + LIVE_TRADING_ENABLED-Test
R6	Idempotenter Apply unnötige Backups	Pre-Write-SHA256-Compare; Test
R7	Backup-Restore-Pfad unklar	post-write-verify-mismatch path explizit; Test
R8	Worker mehr als einen Command	--once-Mode in G10-4.2b; FOR-UPDATE-SKIP-LOCKED
R9	Bot liest halbfertige Datei	os.replace POSIX-atomar; Reader fällt auf {} (G10-1)
R10	Audit unvollständig	strikte Tests für Event-Shape

Stop-Regeln (G10-4.2b Live-E2E)

Trigger	Aktion
Bot-PID wechselt unerwartet	STOP, Restart-Forensik
`runtime_config.json` falscher Inhalt	STOP, Restore from backup_path
`.env` mtime ändert sich	STOP, Forensik
`live_portfolio.json` manuell verändert	STOP (Boundary verletzt)
Worker als daemon statt --once	STOP, kill Worker
Orders / ccxt-Calls sichtbar	STOP, Boundary verletzt
`BINANCE_TESTNET=false`	STOP, Mainnet-Guard verletzt
Mainnet-Pfad sichtbar	STOP
Command failed unerwarteter reason	STOP, untersuche; eventuell rollback
Audit fehlt/unvollständig	STOP, fix audit-emit

Offene Fragen

ID	Frage	Default-Vorschlag
Q-1	Idempotent (no_effect): `previous_values` / `new_values` gleich oder leer?	**gleich** + zusätzlich `no_effect: true` Flag
Q-2	`apply_idempotent` -Logik im Writer ODER Handler?	**Writer** — Single-Point-of-Truth
Q-3	Bei Idempotency Backup-Schritt überspringen?	**ja** — kein Backup wenn no_effect
Q-4	`apply_effective_keys` in Result/Audit-Liste — alle Phasen?	**alle Phasen** — durable Audit-Feld
Q-5	Nur Phase-1.1-Keys (stored_only) im Payload — Apply trotzdem schreiben?	**ja** — `runtime_mutation=true` aber `apply_effective_keys=[]`, `stored_only_keys=[…]`
Q-6	Test-DB für G10-4.2: gleicher G10-3b-Pattern oder separat?	**separat** `tradingbot_gui_g10_4_2_test` (Isolation)
Q-7	Writer Restore-Helper exportieren oder Handler ruft direkt?	**Writer-Helper** `restore_from_backup(backup_path, target_path)` für Test-Coverage

Boundary-Bestätigungen (Scope-Phase)

Check	Status
Read-only Scope	ja
Code-Änderung	nein
Tests geschrieben	nein
Worker	nein
Apply	nein
`runtime_config.json` Write	nein (nicht existent)
Bot-Restart	nein, PID 4246 stable
Orders	nein
Mainnet	nein, BINANCE_TESTNET=true
Push	nein
Backup-Status	für Scope nicht nötig (kein Live-Action). G10-4.2b braucht Pre-Live-Backup per durable Rule.

Finale Empfehlung

Scope-Bestätigung: Vorschlag oben akzeptieren mit den 7 offenen Fragen Q-1 bis Q-7.

Sequenz:

1. **G10-4.2a:** Code + Tests (≥33 neue Tests + Update G10-3b reflection-test). Lokal, kein Live-Effekt. Commit.
2. **Approval-Schleife** für G10-4.2a-Code-Review.
3. **G10-4.2b:** Backup → Container-Sync → Worker `--once` → Live-E2E Apply mit Test-Profile → Verify Bot-Override greift in 1–2 Cycles → Lieferbericht.

Geschätzter Umfang G10-4.2a: ~300 LoC Bot-Code + ~700 LoC Tests, 1 Commit im G10-3b-Stil.
Geschätzter Umfang G10-4.2b: ~6–8 Schritte analog G10-3c.