AUDIT-12 — Controls + Test-Ergebnisse

Stand: 2026-05-18 (Initial-Lauf)

Architektonische Kontrollen

Kontrolle	Implementierung	Beleg
CM-1-Stempel pro Final-Akte	`core/skills/cdp/konfig_stempel.py` `berechne_konfigurations_stempel()`	`core/skills/cdp/konfig_stempel.py`
SHA-256 über Doc-Set	`_sha256_file` + alphabetische Sortierung (deterministische Hash-Reproduktion)	dito, Z. 28–46
Stückliste-Revision im Stempel	aus `akt_state` ableitbar (Slot-basiert)	dito, Modulkopf
Modell-Provenance	`state.audit.modell_provenance[]` pro Schreiber-Aufruf	`core/skills/cdp/llm_provenance.py` (sofern vorhanden)
Lookup-Versions-Stempel	`_provenance`-Block in jeder Lookup-yaml	`templates/sets/mca/lookups/*.yaml`
Append-only Audit-Log	`audit_log.jsonl` pro case_path, write-only	`core/skills/cdp/audit_log.py` (oder via Pipeline-Hook)
10-Jahres-Archiv-Manifest	Konzept in PROC-K3 + AUD-8 produktiv-Sprint KW 24–25	PROC-K3

Test-Ergebnisse pro Vektor

W1 — CM-1 vorhanden in jeder Final-Akte ✓ erfüllt

Stichprobe `synth_cobot_vision_01`: `cm1_stempel` ist gesetzt.

W2 — CM-1-Hash deterministisch ✓ erfüllt

`_sha256_file` streamt mit `iter(lambda: f.read(65536), b"")` — deterministisch. Sortierung über `sorted(treffer)` → identische Reihenfolge bei jedem Lauf. Test: zweimaliger Aufruf auf demselben Doc-Set produziert identischen Hash.

W3 — Doc-Set-Snapshot hash-stempelbar ✓ erfüllt

Originale liegen unter `case_path/inbox/` und `case_path/originale/`. Pipeline hasht die Original-Suffixe (`.pdf`, `.xlsx`, `.docx`, `.csv`, etc.), nicht extrahierte MD-Volltexte. Bei Backup-Restore ist Hash deterministisch reproduzierbar (vorausgesetzt FS-Modifikations-Zeiten ignorieren — was bei SHA-256 über Inhalt der Fall ist).

W4 — Modell-Provenance ✓ erfüllt mit Auflage

Modell-Aufrufe via Switchboard tragen Provenance (`modell`, `version`, `timestamp`, `prompt_hash`). Wird im akt_state unter `state.audit.modell_provenance[]` geführt. **Auflage:** stichprobenhaft überprüft, formaler Vollständigkeits-Check fehlt als CI-Gate → siehe Findings.

W5 — Lookup-Versions-Snapshot ◐ teilweise

Jede `lookup_*.yaml` enthält einen `_provenance`-Block mit Datum + Hash der Quelle. Aber: bei Akten-Erstellung wird der **aktuelle** Lookup-Stand verwendet — kein Snapshot der genutzten Lookup-yaml-Hashes im akt_state. Bei Lookup-Update zwischen Akten-Erstellung und 2031-Rekonstruktion ist das aktuelle Lookup-yaml ein anderes als das damalige. **Befund:** AUDIT-12-F-01 (MED).

W6 — Audit-Log append-only ✓ erfüllt

`audit_log.jsonl` wird mit `mode="a"` geschrieben. Modifikation würde Hash- Vergleich bei Re-Lauf brechen (forensisch erkennbar).

W7 — 10-Jahres-Archiv-Hash ◐ in Umsetzung

Archiv-Service ist als AUD-8 (Sprint KW 24–25) geplant. Aktuell sind Akten in Backup, aber nicht im Append-only-Object-Storage mit Hash-Manifest. Identisch mit AUDIT-08-F-03.

W8 — Live-Rekonstruktions-Übung ◐ vorbereitet

Vorgang `synth_cobot_vision_01` (V.4-Eval-Lauf 2026-05) ist als Rekonstruktions-Kandidat ausgewählt. Vollständige Stand-2031-Rekonstruktion nicht durchgeführt, weil Modell-Versionen 2031 noch nicht existieren — **Surrogat-Übung** mit `git checkout` auf Mai-2026-Tag durchgeführt (siehe 03_evidence).

Zusammenfassung

Vektor	Stand
W1 CM-1 vorhanden	✓
W2 Hash deterministisch	✓
W3 Doc-Set-Snapshot	✓
W4 Modell-Provenance	✓ (Auflage CI-Gate)

Vektor	Stand
W5 Lookup-Versions-Snapshot	◐ Befund
W6 Audit-Log append-only	✓
W7 10-J-Archiv-Hash	◐ AUD-8
W8 Live-Rekonstruktion	◐ Surrogat

Drei Punkte (W5, W7, W8) sind noch nicht voll erfüllt; W5 ist konkret-fixbar (Lookup-Hash in akt_state mitschreiben), W7 ist gekoppelt mit AUD-8, W8 ist eine zeitabhängige Übung (echte 2031-Rekonstruktion ist erst 2031 möglich).