# Bilko — Financial Audit Trail & Retention Architecture

# Bilko — Financial Audit Trail & Retention Architecture

**Parent:** MC #105872 · **Plan:** `~/.claude/plans/greedy-greeting-flame.md` (CEO-approved 2026-07-17) · **Design team:** Petter Graff (lead), Martin Kleppmann (event-log semantics), Bruce Momjian (Postgres)

CEO mandate (2026-07-17): Bilko is serious accounting software — every mutation of a financial record must leave a trace ("if someone deletes a file and claims we lost their data, we must be able to prove what happened"). Files themselves do not have to be kept once their legal retention window expires, but the **trace of what happened must survive forever**.

## Status at time of writing (2026-07-17)

Phase A (foundation) is **merged to** `azdo/main` via Azure DevOps PR #178 (commit `c44ee7c9`, squash-merge of the full A2–A8 + C2 chain, 24 files, 4707 insertions) — genuinely a two-parent merge commit (`git cat-file -p c44ee7c9` shows parents `3f144e82`/`ce7ee96f`), consistent with a real PR merge, though this specific instance doesn't carry azdo's usual auto-generated "Merge pull request N from branch into main" message text that its neighboring commits do (those PRs were completed via REST rather than the azdo web UI the same evening, per team lead). Three sibling PRs from the same wave: **#179** (Lexicon legal docs, commit `a71a0b15`), **#180** (D1/D2 demo-country fix, commit `3f144e82`), **#181** (B4 storage consolidation, commit `ce64b93b`). PR numbers/attribution here are as reported by the team lead; this session could not independently query the azdo PR REST endpoint (`az repos pr show` fails with a local keychain error, `-25308 Can't fetch password from system`, unrelated to the PAT itself — see `~/.claude/projects/-Users-makinja/memory/technical_azdo_git_auth_paths_sp_not_in_org_2026-07-17.md`, PAT was rotated and confirmed working earlier the same day). The merge-commit hashes and their file contents **are** independently tool-verified (`git log`/`git show --stat` against `azdo/main`). This page documents what is **live in** `main` **today** and flags explicitly what is still open.

| Item | Status |
| --- | --- |

| | |
|---|---|
| A1 — RS/BA/HR retention law verification | Done (MC #105873) |
| A2/A3 — `financial_audit_log` + `document_retention_manifest` schema | Merged main (V124, V125), PR #178 |
| A4/A5 — capture trigger, pilot + 6-table fan-out | Merged main (V126, V128), PR #178 |
| A6 — actor/request-id context plumbing | Merged main (3 adversarial rounds, see below), PR #178 |
| A7 — backfill of existing rows | Merged main (V129), PR #178 |
| A8 — activity feed + entity timeline read API | Merged main (V130), PR #178; **web UI panel not built**, flagged to Vizu |
| C1 — `deleted_by` / `deletion_reason` columns | Merged main (V127), PR #178 |
| C2 — `InvoiceService.deleteInvoice` USER_DELETE snapshot | Merged main, PR #178 |
| B4 — Storage: R2/GCS → Azure Blob SDK, `deleteBlob()` | Merged main (`ce64b93b`), PR #181 |
| B1 — retention citation fixes + per-type override table | **MC #105880, in progress** — not yet in code |
| B5 — upload path writes `document_retention_manifest` row | **MC #105884, in progress** |
| B2 — purge worker, dry-run mode | **MC #105881 created, blocked** — red-zone (Momjian+Graff), waiting on A3+B1 |
| B3 — purge worker, live mode | **MC #105882 created, blocked** — waiting on B2 + ≥1 clean dry-run sprint on stage + explicit CEO sign-off before first live activation |
| D1/D2 — demo-country display bug (parallel workstream, same wave, not part of audit trail scope) | Fixed and merged (`3f144e82`), PR #180; documented separately, referenced here only for context |

**Do not treat B1/B2/B3/B5 as done.** MC #105881/#105882 exist but are deliberately blocked, not started — the purge worker does not run anywhere today; nothing purges blobs or applies retention. This page's "purge runbook" section below describes the *design*, not a currently operable procedure.

# Why a third audit table

Two audit mechanisms already existed before this program:

- **V1** `logged_actions` — append-only JSONB, but only ever used for `ENTRA_JIT_LINK` / `SECURITY_VIOLATION` events, and has no `org_id` / `country_code` (any RLS check against it needs an expensive subquery).
- **V51** `audit_log` — append-only, but scoped to admin-portal actions only.

Neither covers ordinary financial-entity mutations (invoices, expenses, contacts, transactions). Before this program, `InvoiceService.deleteInvoice` was a **hard delete with no audit trail at all** ( `InvoiceService.kt:1004-1018` pre-change) — the exact gap the CEO mandate calls out.

Kleppmann's arbitration: build a **third** table, not a consolidation of V1/V51 and not per-domain tables. It carries `org_id` + `country_code` from day one (unlike V1), so ADR-017 Phase 2B-style partitioning/tenant-query patterns work immediately.

# Schema

## `financial_audit_log` (V124, `apps/api/src/main/resources/db/migration/V124__financial_audit_log.sql`)

Partitioned, append-only audit trail for financial-entity mutations and read-sensitive actions.

```
event_id       BIGINT GENERATED ALWAYS AS IDENTITY
org_id         UUID NOT NULL REFERENCES organizations(id)
country_code   VARCHAR(10) NOT NULL      -- denormalized, indexed, NOT the partition key
entity_type    VARCHAR(32)
entity_id      UUID
action         VARCHAR(24) NOT NULL      -- INSERT | UPDATE | DELETE | EXPORT | DOWNLOAD |
                                          -- RESTORE | USER_DELETE | RETENTION_PURGE
actor_user_id  UUID REFERENCES users(id) ON DELETE SET NULL
actor_label    TEXT
before_data    JSONB
after_data     JSONB
changed_fields TEXT[]
reason         TEXT
request_id     TEXT
client_ip      INET
is_backfilled  BOOLEAN NOT NULL DEFAULT false
occurred_at    TIMESTAMPTZ(6) NOT NULL DEFAULT now()

PRIMARY KEY (event_id, occurred_at)   -- occurred_at required in PK: it's the partition key
```

Design decisions (Momjian + Kleppmann arbitration, all tool-verified against the live schema, not assumed):

- **`PARTITION BY RANGE (occurred_at)`, monthly.** Purge is a time predicate (`DROP PARTITION`, never a million-row `DELETE`). `country_code` is a plain indexed column, not the partition key — this is the explicit rebuttal of ADR-017 Phase 2B's original country-list partitioning idea

(Momjian: YAGNI, purge doesn't need country as a physical boundary).

- `retain_until` **is deliberately NOT a column.** Retention differs by jurisdiction (HR/RS/BA_FED/BA_RS) and by entity type within a jurisdiction (see retention table below), so it's computed at purge time from `organizations.legal_retention_years` plus a per-type override table (B1, not yet built), not stored redundantly on every row.
- **Append-only enforcement is two layers, not one:** `REVOKE UPDATE/DELETE FROM bilko_app` (grants: INSERT+SELECT only) **and** `BEFORE UPDATE/DELETE RAISE EXCEPTION` triggers on the partitioned parent. The trigger layer exists specifically to also catch `bilko_admin`, which is `BYPASSRLS` but *not* superuser — `BYPASSRLS` skips RLS policies, it does not skip triggers. PG16 propagates parent triggers to all partitions automatically, including ones created later by the (not-yet-built) partition-maintenance job.
- **RLS is PERMISSIVE + FORCE**, consistent with the current Phase 2A org-isolation era — `org_isolation_select`/`org_isolation_insert` scoped to `app.current_org_id`, plus `platform_admin_full_read`. A RESTRICTIVE flip is an explicit, separate, later CEO-gated decision; this migration does not touch it.
- `country_code` **is** `VARCHAR(10)`**, not** `VARCHAR(8)`**.** This was a real bug caught by Momjian's design review of the first commit: the source column (`organizations.country`) was widened to `VARCHAR(10)` back in V16 specifically to hold `BA_FED`/`BA_RS` (6 chars), and V120 did the same for `compliance_deadlines.country` "to keep country-code columns consistent." The original `VARCHAR(8)` would have fit today's values with zero headroom — any future jurisdiction code longer than 8 characters would have thrown inside the audit trigger and aborted the *parent* invoice/expense mutation transaction, which is a worse failure mode than a normal app-level validation error on a table explicitly designed to be an invisible safety net. Fixed pre-merge; a dedicated `BA_FED`/`BA_RS` width test fixture was added (previously the 30/30 green suite only exercised `HR`/`RS`/`BA`, none of which are close to the old 8-char limit).
- Initial monthly partitions cover 2026-07 through 2026-10 only. **A partition-maintenance job to create future months does not exist yet** — this is an open operational gap, not a documentation oversight; inserts for November 2026 onward will fail with "no partition found for row" unless a job is built before then.

# `document_retention_manifest` (V125, generalizes the existing V78 `hr_einvoice_archive` WORM pattern to all uploaded files)

```
manifest_id       BIGSERIAL PRIMARY KEY
org_id            UUID NOT NULL REFERENCES organizations(id)
entity_type       VARCHAR(32)
entity_id         UUID
original_filename TEXT NOT NULL
```

```
content_type      TEXT
file_size         BIGINT
sha256_hex        CHAR(64) NOT NULL
storage_backend   TEXT
storage_path      TEXT
uploaded_by       UUID REFERENCES users(id) ON DELETE SET NULL
uploaded_at       TIMESTAMPTZ(6) NOT NULL DEFAULT now()
blob_purged_at    TIMESTAMPTZ(6)
blob_purge_reason TEXT
purged_by_job     TEXT
```

This table is **permanent by design** — it has no `retain_until` because it never expires. It is the direct, literal answer to the CEO scenario: a row here proves a file existed, who uploaded it, its exact SHA-256, and — once the blob itself is purged — exactly when, why (`RETENTION_PURGE` vs `USER_DELETE`, matching `financial_audit_log.action`), and by which job run. `bilko_app` has INSERT+SELECT only; there is no append-only trigger here (unlike `financial_audit_log`) because a legitimate, narrow future UPDATE path exists — the not-yet-built purge worker marking `blob_purged_at` — and that grant is deliberately deferred to the Phase B2/B3 migration that creates the `bilko_purge_worker` role, since granting to a role that doesn't exist yet is a Flyway failure.

# Capture mechanism (V126 pilot on `invoices`, V128 fan-out to 6 more tables)

Arbitration point 1 (Graff + Momjian, overruling a pure application-level design): **capture is a generic `AFTER` DB trigger** (`capture_financial_audit()`, `to_jsonb(OLD/NEW)`, keyed off `TG_TABLE_NAME`), synchronous and transactional — not `pg_notify`/async, which would create a durability gap. This was chosen specifically because this same repo already proved app-level capture gets forgotten: `deleteInvoice` never called `logged_actions` before this program. Covers `invoices`, `expenses`, `contacts`, `transactions`, `bank_accounts`, `bank_transactions`, `invoice_items`.

Read-sensitive actions the trigger structurally cannot see — `export`, `download`, `restore` — are Kleppmann's second arbitration point: those are **application-level** events written into the same table from the service layer, since a DB trigger only fires on `SELECT`... it never fires at all, it can't be the mechanism for read events.

# C1/C2 — soft-delete and hard-delete audit coverage

- **V127** adds `deleted_by` + `deletion_reason` to every table with `deleted_at` — the actual list was tool-verified via a live `information_schema.columns` query against a full V1..V126 Testcontainers replay, not assumed from the task spec, and correctly caught tables the

spec missed (`bank_accounts`, `chat_conversations`, `exchange_rates`, `expense_documents`, `offer_items`, `offers`, `travel_orders`).

- **C2**: `InvoiceService.deleteInvoice` still does a hard delete (DRAFT-only semantics unchanged), but now writes a `financial_audit_log` row with `action='USER_DELETE'` — a full invoice+line-items JSON snapshot — in the same transaction, before the delete. This is deliberately **not deduplicated** against the V128 trigger's own low-level `DELETE` row for the same statement: `USER_DELETE` is the business-level "someone claims we lost their invoice, prove what happened" answer; the trigger's `DELETE` row is the safety net that fires even for deletions that bypass the service entirely (e.g. direct SQL).

# A8 — read API

`GET /orgs/{id}/activity` (org-wide feed) and `GET /{entity}/{id}/timeline` (single-entity history), both paginated, both using the composite indexes created in V124. Gated by a new `activity:read` permission (V130), seeded **only** for `owner`/`admin` — not `viewer`, not `accountant` — because `before_data`/`after_data` carry full row snapshots including PII (e.g. a contact's OIB/JMBG-equivalent tax ID). Both endpoints filter directly on `financial_audit_log.org_id = principal.organizationId` rather than the existing `ResourceAccessFilter.requireOrgOwnership` helper, because that helper's table coverage predates this program and is missing 3 of the 7 audited tables. Cross-org access returns a 404 for the org feed and an empty list (not a 403/404) for the entity timeline, specifically to avoid leaking entity-existence across tenants via response-code side channel. **The web UI panel for this API was explicitly scoped out** — flagged to Vizu as a separate task rather than built as a rushed stub, per the implementing agent's own assessment that building it to the existing app's quality bar is realistically over an hour of frontend work.

# A6 saga — three adversarial rounds, two real bugs caught

A6 (`SET LOCAL app.current_user_id` / `app.current_request_id`, MC #105877) is the most important lesson from this program for anyone touching Ktor plugin install-order in this codebase. It went through three rounds of Momjian build → Parisa Tabriz (Securion) adversarial verify before landing.

**Round 1 finding (ThreadLocal vs. coroutine dispatcher hop):** the original implementation used a plain `ThreadLocal` to carry actor context. `dbQuery{}` dispatches onto `Dispatchers.IO`, which is a thread pool — a `ThreadLocal` set on the request-handling thread is simply not visible on whatever IO-pool thread the query actually executes on. Result: `actor_user_id` silently NULL for essentially all real requests. Fixed by switching to a kotlinx.coroutines `ThreadContextElement`, which is designed to survive exactly this kind of dispatcher hop.

**Round 2 finding (install-site / phase-ordering — the more interesting one):** even with the coroutine-context mechanism now correct in isolation, it was still not effective end-to-end, because `installOrgScopePlugin()` was installed via `intercept(ApplicationPhase.Plugins)` at the

`Application.module()` level — which runs **before** Ktor's routing tree even dispatches into the `authenticate("bilko-jwt") { }` block, i.e. before `BilkoPrincipal` is resolved. So `call.principal<BilkoPrincipal>()` inside the interceptor was **always** null in production, authenticated request or not.

This is the same class of bug as **MC #104962**, which had already found and fixed the identical mistake for the sibling `TrialGatePlugin`: install at `Application` level and the plugin sees a permanently-null principal, no matter how correct its internal logic is. `TrialGatePlugin`'s own doc comment says this explicitly — but `installOrgScopePlugin()` had not been given the same fix.

The org_id side of the same mechanism was *not* affected, and understanding why is the actual transferable lesson: `orgTransaction(organizationId: String, ...)` takes `organizationId` as an explicit function parameter, sourced by every real call site via `effectiveOrgId(principal)` called **inside** the route handler (which does run after auth). So org_id never depended on the broken plugin's principal read at all — `currentOrgIdThreadLocal` turned out to be dead code in production (zero call sites). `actor_id` had no equivalent parameter — its only source was the broken plugin — so only the actor side silently failed.

**Fix:** `installOrgScopePlugin()` was changed from an `Application` extension to a `Route` extension, intercepting `ApplicationPhase.Call` (not `Plugins`), called as the literal first statement inside `authenticate("bilko-jwt") { }` in `Routing.kt` — before `install(TrialGatePlugin)` and all route registrations. `TrialGatePlugin` couldn't be copied verbatim as a pattern because it uses the `on(AuthenticationChecked)` hook, which is a plain synchronous callback with no `proceed()` — fine for a check-and-throw, but unable to keep a coroutine context element alive forward into a later route handler's `dbQuery` call, which is exactly what A6 needs.

Round 3's regression test (`OrgScopeActorContextHttpIntegrationTest.kt`) is itself a lesson worth keeping: it deliberately goes through the *real* HTTP client, the *real* JWT verifier, and the *real* routing tree — not the `withActorContextForTest` seam that round 2's test used, which structurally could not have caught this bug because it never calls `installOrgScopePlugin()`'s actual install path at all.

**Lesson for future Ktor plugin work in this codebase:** if a plugin needs `call.principal<T>()` to be non-null, verify empirically (not by inspection) which phase it actually observes the principal at — `ApplicationPhase.Plugins` at the `Application` level is provably too early for anything gated by `authenticate(...)`, regardless of where in the source file the `intercept()` call is textually nested. This has now bitten two different plugins (`TrialGatePlugin` in #104962, `OrgScopeSessionVariable` here) with the same root cause.

# Retention law — verified findings (MC #105873, primary sources)

| Jurisdiction | Books (journal/ledger) | Auxiliary books | Financial statements | e-invoice (fiscalized) | Payroll records |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| Croatia (HR) | at least 11 years (ZOR NN 78/2015 čl. 10(2)) | at least 11 years | permanent* (pin at B1) | 6 years (Zakon o fiskalizaciji NN 89/2025 čl. 35) | ≥6 years / analytics permanent |
| Serbia (RS) | 10 years (Zakon o računovodstvu, "Sl. glasnik RS" 73/2019 čl. 28) | 5 years | 20 years | n/a (SEF rules separate — pin at B1) | permanent |
| BiH — Federation (BA_FED) | at least 11 years (Zakon o računovodstvu i reviziji FBiH, "Sl. novine FBiH" 15/2021 čl. 49) | pin at B1 | pin at B1 | n/a | pin at B1 |
| BiH — Republika Srpska entity (BA_RS) | at least 10 years ("Sl. glasnik RS" 115/2025, article number TBD — not yet pinned from primary text) | at least 5 years | permanent | n/a | pin at B1 |

Two compliance findings from this verification are **not yet fixed in code** — they are B1's job, not done:

- **Code cites a dead law:** `PluginRS.kt:325` cites "Sl. glasnik RS br. 62/2013, čl. 24" — that law was replaced by 73/2019 in its entirety. The 10-year figure for books happens to still be correct, but the legal basis citation is wrong and must be updated to 73/2019 čl. 28.
- □□ **Compliance gap, not just a citation issue:** `V50` currently sets `organizations.legal_retention_years = 10` for BA orgs generally. FBiH law requires **at least 11**. Any `BA_FED` organization in Bilko today is configured with a retention period one year short of its legal minimum. This is a live data-value fix, not a comment fix, and is the highest-priority item inside B1.

The BA_RS article number in the new 115/2025 law could not be pinned from available sources (target site had a TLS failure); the 10/5/permanent figures are confirmed via secondary sources, but B1's implementer must pin the exact article from the official Sl. glasnik RS 115/25 text before it is cited in any docs or code comment — this is a repeat of the same "don't propagate an unverified article number" discipline that caught the RS 62/2013→73/2019 gap in the first place.

# GDPR — erasure vs. retention

Position (plan arbitration point 7, not yet formalized in Privacy Policy/DPIA — that's Lexicon's MC #105277, linked, separate from this page): under **GDPR Art. 17(3)(b)**, the right to erasure does not apply where processing is necessary for compliance with a legal obligation — here, the accounting-law retention requirements in the table above. A financial audit snapshot ( `financial_audit_log.before_data`

/`after_data`, or a `document_retention_manifest` row) is **not** deleted on a user erasure request during the applicable retention window. The only erasure-adjacent action available is pseudonymizing `actor_label`/user-identifying fields after the retention window closes, and that is explicitly scoped as: every erasure request goes through Securion/legal review, never a self-serve deletion path, and never automatic.

# Purge worker — design (not yet built; B2/B3 are open tasks)

This section is a design skeleton for the not-yet-implemented `RetentionPurgeWorker`, so it is documented in the same place as the schema it operates on. **Nothing described below exists in the running system today.**

- **Primary path:** `DROP TABLE <partition>` for a `financial_audit_log` monthly partition once every row in it is past its jurisdiction's retention window. Fallback for mixed-retention partitions (a partition spanning rows from different orgs/countries with different retention lengths): batched `DELETE` with `LIMIT 5000` + commit per batch + `pg_sleep` throttle.
- **Order of operations is fixed and load-bearing: blob first, then row.** Deleting the `document_retention_manifest`/audit-row pointer before the blob would create an orphaned blob with no trace pointing to it — the opposite of this program's purpose. The blob delete must succeed (or be confirmed already-gone, 404-as-success, matching `ReceiptService.deleteBlob()`'s existing idempotent contract from B4) before the manifest row is marked `blob_purged_at`.
- **Dedicated `bilko_purge_worker` role**, `BYPASSRLS`, least-privilege, narrower than `bilko_admin`. A `SECURITY DEFINER purge_expired_partition()` function owned by `bilko_admin` is the intended privilege-elevation boundary rather than granting the worker role broad table access directly.
- **The purge worker's own actions are themselves audited** ("audit the auditor") in a `purge_worker_log` table — any trigger-disable needed for a legitimate purge happens inside the same transaction as the log write.
- **Dry-run is the default mode, and is required before any live run.** A dry-run must print the candidate rows/partitions it *would* purge and delete nothing; the acceptance test for B2 is specifically that a row still exists after a dry-run pass.
- **Live mode (B3) requires explicit CEO sign-off before its first activation**, and only after at least one full sprint of clean dry-run output on stage. This is a □□ red-zone item in the parent plan — senior-only, never a locally-dispatched builder task.
- **Cron mechanism is not yet confirmed.** `pg_cron` availability on Azure Flexible Server has not been verified (flagged to FlowForge); the fallback is the existing app-level cron pattern already used by `InvoiceCronRoutes.kt` (`COMPLIANCE_CRON_SECRET`).
- **How to read a dry-run log once B2 exists:** each dry-run entry will list, per candidate partition/row set: org, country, entity type, computed `retain_until` (from `organizations.legal_retention_years` + the not-yet-built per-type override table from B1),

and the action that would be taken (`DROP PARTITION` vs. batched `DELETE`). Approval for a live run means a human has read that candidate list and confirmed it matches expectation — not just that the dry-run exited zero.

# Relationship to ADR-017 Phase 2B

ADR-017 Phase 2B originally proposed country-code-list partitioning. It was never implemented — Flyway's actual `HEAD` was `V116` at the start of this program (V36/V37 slots referenced in the old ADR discussion were long since consumed by unrelated migrations), so this program's `financial_audit_log` design is effectively a green-field implementation of "partition a big multi-tenant table," not a migration of an existing partitioned table. Momjian's rebuttal, adopted here: partition by `occurred_at` (RANGE, monthly) with `country_code` as a plain indexed column, not a partition key — purge is fundamentally a time predicate, and a country-list partition scheme would need to be extended by hand every time a new market is added, for no purge-performance benefit.

# Testing discipline (applies to every migration in this program)

Every migration listed above has a companion `V1xxMigrationTest.kt` (Testcontainers `postgres:16`) that does a **full `V1..V1xx` replay**, not a fresh-schema shortcut — this is the direct lesson from the V121 incident (Exposed's `SchemaUtils.create()`-based test schemas do not see `CHECK` constraints introduced by a migration; the only way to prove a constraint is real is to replay the actual migration chain and show that removing it makes the negative test fail). Each migration test includes negative tests (`UPDATE`/`DELETE` on the audit tables must throw, for both `bilko_app` and `bilko_admin`), an RLS cross-org test, and — where relevant — a partition-existence/partition-drop test.

# Evidence index

| Task | Evidence |
|---|---|
| A1 retention law verification | `~/system/evidence/105873/a1-retention-verification.md` |
| A6 build (3 rounds) | `~/system/evidence/105875/momjian-review.md` |
| A6 adversarial verify (Securion, 3 rounds) | `~/system/evidence/105877/parisa-verify.md`, `~/system/evidence/105877/verdict.md` |
| A7 backfill | `~/system/evidence/105878/verdict.md` |
| A8 activity/timeline API | `~/system/evidence/105879/verdict.md` |

| Task | Evidence |
|---|---|
| B4 storage consolidation | `~/system/evidence/105883/verdict.md` |
| C1 soft-delete columns | `~/system/evidence/105885/verdict.md` |
| C2 delete snapshot | `~/system/evidence/105886/verdict.md` |
| D1 demo-country bug root cause | `~/system/evidence/105874/d1-rootcause.md` |
| D2 demo-country bug fix + E2E | `~/system/evidence/105887/d2-verdict.md` |

# Open follow-ups

- MC #105880 (B1) — fix the dead-law citation and the BA_FED 10→11 year compliance gap; build the per-type retention override table.
- MC #105884 (B5) — upload path writes a `document_retention_manifest` row at upload time (blocked on both V125 and the B4 Azure Blob refactor being merged, which they now are — this can proceed).
- B2/B3 purge worker — MC #105881/#105882 created, both blocked (B2 on A3+B1; B3 on B2 + a clean dry-run sprint + CEO sign-off). Dry-run first, live mode requires CEO sign-off.
- Partition-maintenance job for `financial_audit_log` beyond 2026-10 — not started; will start failing inserts in November 2026 if not built.
- Web UI panel for the A8 activity/timeline API — flagged to Vizu, not scheduled.
- MC #105277 (Lexicon) — Privacy Policy/DPIA update for the Art. 17(3)(b) erasure-vs-retention position described above.
- ADR for this design has not been written into the repo yet — outline below, for whoever picks up that task.

# ADR outline (for a future repo-side ADR, not written to the repo by this task)

Proposed title: **ADR-0XX — Financial Audit Trail: append-only partitioned log, not a consolidation of V1/V51**

1. Context: CEO mandate, gaps in V1/V51 coverage, `deleteInvoice` hard-delete-with-no-trail example.
2. Decision: third table (`financial_audit_log`), RANGE-partitioned by `occurred_at` monthly, `country_code` denormalized not partition key; dual append-only enforcement (REVOKE + trigger); hybrid capture (DB trigger for mutations, application-level for read-sensitive actions); permanent `document_retention_manifest` for file-level trace, decoupled from blob lifecycle.

3. Alternatives considered and rejected: consolidating into V1 or V51 (Kleppmann — wrong semantics, V1 has no tenant context); async/`pg_notify` capture (durability gap); country-code partitioning per original ADR-017 Phase 2B sketch (Momjian — no purge-performance benefit, extra operational burden per new market); hash-chained tamper-evidence (deferred, not rejected — revisit only on explicit regulatory/enterprise requirement).
4. Consequences: purge becomes a partition-drop operation once B2/B3 land; every future table needing financial audit coverage follows the V126/V128 trigger-fan-out pattern; A6's install-site lesson should be called out as a standing Ktor-plugin gotcha for this codebase.
5. Status: Phase A implemented and merged; Phase B (retention enforcement) open.

---

Revision #2
Created 2026-07-17 20:27:24 UTC by John
Updated 2026-07-17 20:33:54 UTC by John