Baseline handoff design and overwrite prevention

Finanzas SD – Architecture, Flows & SOPs

Arquitectura, Flujos y Procedimientos

January 9, 2026

1 Baseline handoff design and overwrite prevention

1.1 Root cause of the overwrite

Historically the handoff endpoint trusted the `projectId` provided in the path even when the incoming baseline belonged to a different project. When the resolver detected a different `projectId` for the baseline it only swapped the identifier but continued to reuse the metadata fetched for the **path project**, so the final `PutCommand` wrote a new baseline into the wrong PROJECT#…/METADATA item. Because the request context (`created_by = system`) became the source of truth, the prior project owner, acceptance metadata, and SDM assignment were overwritten. The new baseline-aware helper now returns both the correct `projectId` **and** the resolved project metadata so downstream writes preserve original authorship and SDM ownership.

1.2 Design goals

- **Idempotency:** repeated handoffs with the same idempotencyKey return the original {handoffId, projectId, baselineId} tuple and never create duplicate projects.
- **Baseline isolation:** every baseline maps to a single PROJECT#…/METADATA record; different baselines cannot collide even if the UI reuses a stale project ID.
- **Metadata preservation:** ownership fields (`created_by`, `created_at`, `sdm_manager_name`) are taken from the resolved project metadata whenever it exists so system contexts do not clobber user-entered values.
- **Auditable writes:** every handoff produces a HANDOFF#… item plus an `audit_log` entry that records the before/after payloads and the actor.

1.3 Data flow

1. **Baseline normalization** – the handler normalizes incoming baseline/handoff payloads and extracts `baselineId`, deal inputs, and acceptance metadata.
2. **Project resolution** – `resolveProjectForHandoff` is called with `baselineId`, the incoming (path) `projectId`, and the `idempotencyKey`.

   - Checks idempotency cache and returns the cached tuple when the key matches the same baseline.
   - If the path project already carries the baseline, it is reused.
   - If the path project carries a **different** baseline, the helper searches for an existing project with the incoming baseline; if none is found it generates a new P-<uuid> project ID to avoid collisions.
   - The helper returns both `resolvedProjectId` and any `existingProjectMetadata` for that baseline so the handler does not rely on the path project's metadata.

3. **Metadata hydration** – the handler replaces `existingProject.Item` with the resolver's metadata (or fetches it when the resolver chose a different project ID) so subsequent writes use the correct `created_by`/`created_at`/`sdm_manager_name` values.
4. **Project write** – a `PROJECT#…/METADATA` item is written with:

   - Baseline details (id, status, acceptance timestamp).
   - Project identity (code derived from baseline for long UUIDs) and client/name fields.
   - Ownership fields preserved from existing metadata when present.

5. **Handoff record** – a `HANDOFF#…` item captures the normalized payload and owner, and an idempotency record stores the tuple for future retries.
6. **Audit trail** – `audit_log` receives a `HANDOFF_UPDATE` entry with `before`/`after` snapshots to aid investigations.

1.4 Components touched

- **`services/finanzas-api/src/handlers/projects.ts`** – consumes resolver metadata, builds the project item with preserved ownership, writes handoff and audit entries.
- **`services/finanzas-api/src/lib/projects-handoff.ts`** – performs baseline-aware project selection, idempotency checks, and metadata retrieval.
- **DynamoDB tables** – `projects` (for `PROJECT#…/METADATA`, `HANDOFF#…`, `IDEMPOTENCY#HANDOFF`) and `audit_log` are written during handoff.

1.5 Behavioral expectations

- Adding a new baseline to an existing project ID will **not** overwrite another project's metadata; a new project is generated or the correct baseline-specific project is reused.
- Retrying the same handoff with the same idempotency key returns the original identifiers without creating extra projects.
- Ownership fields remain attributed to the original user even when handoff runs under `system` credentials.
- Audit logs always reflect the state transition so regressions can be traced.

1.6 Testing performed

- `npm run test:unit` (tsx) – validates auth role resolution, API service coercion, baseline creation error handling, cost utilities, role routing, and project normalization logic relevant to SDMT handoff flows.

1.7 Defensive measures against baseline collisions

1.7.1 Pre-write METADATA baseline check

As an additional safety layer, `projects.ts` includes a defensive check immediately before writing PROJECT#…/METADATA:

1. Read the current METADATA record for the resolved project ID
2. Extract any existing `baseline_id` or `baselineId` field
3. If an existing baseline is present and differs from the incoming baseline:

   - **Refuse the write with HTTP 409 Conflict**
   - Return error details: `{ error: "baseline collision detected", existingBaselineId, newBaselineId, projectId }`

   This prevents cross-baseline overwrites even if the resolution logic has edge cases or receives incorrectly routed requests.

1.7.2 Idempotency conflict detection

The `resolveProjectForHandoff` helper enforces baseline-scoped idempotency:

- If an idempotency key was previously used with a different baseline, it throws `IdempotencyConflictError`
- The handler catches this error and returns HTTP 409 Conflict with: `{ error: "idempotency conflict", message }`
- This ensures the same idempotency key cannot be reused across different baselines

1.7.3 QA project handling

When a project exists without a baseline (e.g., a QA test project):

- The **first** baseline handoff can claim that project
- Subsequent handoffs with **different** baselines will create new project IDs
- This prevents multiple baselines from accidentally sharing a single QA project

1.8 Error responses

1.8.1 409 Conflict: Baseline collision

```
[] { "error": "baseline collision detected: metadata already exists for a different baseline", "projectId": "P-1b6309be-bf75-4994-a332-097bdfc63ae4", "existingBaselineId": "base_eac8ddf69dbb", "newBaselineId": "base_b8566fa19c08" }
```

1.8.2 409 Conflict: Idempotency conflict

[] { "error": "idempotency conflict", "message": "Idempotency key \"key-123\"was previously used with baseline \"base_old\"but is now being used with baseline \"base_new\""

}