

CRM ARCHITECTURE / DATA INTEGRITY

Fixing 885 broken records with a *timing* problem

A multi-location medical spa's HubSpot pipeline had quietly desynchronised from its point-of-sale system. Deals were advancing to "Appointment Scheduled" with no appointment attached to them. The cause was not the automation logic. It was the order events arrived in.

885

ORPHANED DEALS
IDENTIFIED

6⁺

COMPETING
WORKFLOWS

4^x

DAILY POS SYNC
(NOT REAL-TIME)

2

DEFECTS CAUGHT
PRE-LAUNCH

DESIGNED & ENGINEERED BY

Manan Asim

CRM, RevOps & Automation Engineer

HUBSPOT OPERATIONS HUB
CUSTOM-CODED ACTIONS · NODE.JS
CRM V3 / V4 API

The context

The client is a multi-location medical spa running daily aesthetic appointments across several centres. HubSpot is their marketing and sales system of record. A separate point-of-sale platform owns the booking calendar, and a third-party integration pushes appointment records across into HubSpot as a custom object.

How the model is meant to work

A prospect submits a lead form from an ad or the website. Routing automation creates a deal in the Sales Pipeline at **New Lead**. When that person books, an appointment record is created in the POS and syncs into HubSpot against their contact.

The deal and the appointment are then supposed to be associated to one another. That single link is load-bearing: downstream automation reads it to advance the deal through the pipeline, and marketing reporting depends on it to attribute booked revenue back to the campaign that generated the lead.

What was actually happening

A large and growing population of deals had reached **Appointment Scheduled** with no appointment object attached and no appointment date populated. The pipeline reported activity that could not be traced to a real booking.

Six or more overlapping workflows had accumulated over time, each attempting to move deals forward on a slightly different signal. None of them reliably created the association itself.

THE REPORTING CONSEQUENCE

Because attribution flows through the deal-to-appointment link, every orphaned record represented booked revenue that could not be traced back to its originating campaign. The integrity problem was upstream of every marketing performance question the business wanted to answer.

ENGAGEMENT

Sector	Medical aesthetics
Footprint	Multi-location
Discipline	RevOps / CRM
Engagement	Diagnostic + build
Deal pipelines	3 (1 relevant)
Appt. statuses	19 (1 "open")

SCOPE OF WORK

- Root-cause diagnosis
- Object & property audit
- Association engine build
- Workflow consolidation plan
- Historical remediation script
- Monitoring & QA reporting

STACK

HubSpot Operations Hub

Node.js 20 CRM v3 API

CRM v4 Assoc. Custom objects

Private apps

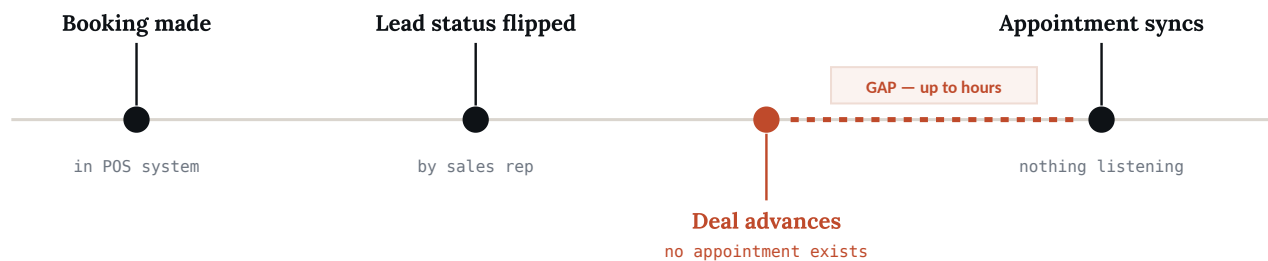
A race condition, not a logic error

Every workflow attempting to advance a deal was triggered by a **deal-side** event. The record it needed to find did not exist yet at the moment it ran.

When a representative books a client, three things happen in sequence. The booking is entered into the point-of-sale system. The representative then updates the contact's lead status, which immediately fires automation and advances the deal. Only later, on a batch cycle that runs four times per day, does the appointment record actually arrive in HubSpot.

By the time the appointment landed, every workflow that could have linked it had already run and exited. Nothing was listening. The deal was left permanently stranded in a stage that implied a booking it had no record of.

BROKEN SEQUENCE



The automation was never wrong. It was simply asking a question the system could not yet answer, and then never asking again.

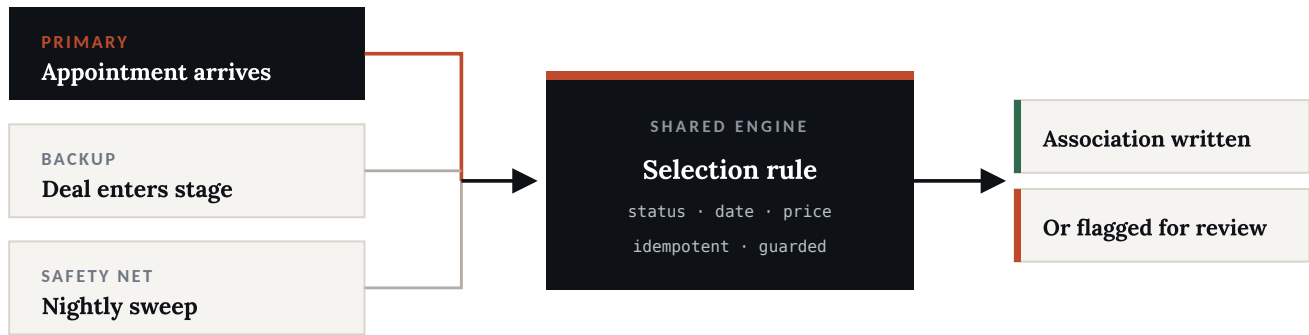
DIAGNOSTIC FINDING

Why adding more workflows would not have worked

The intuitive remedy is to tighten the existing stage-movement workflows. That approach fails for a structural reason: no amount of refinement to a deal-triggered workflow changes *when* it runs. The correct fix was to invert the trigger and drive association from the appointment side, so the link is created at the moment the missing data finally arrives.

Inverting the trigger

One association engine, reached by three independent paths. Association became the source of truth, and pipeline stage became a consequence of it rather than a competing signal.



PRIMARY — APPOINTMENT-TRIGGERED

Fires the moment an appointment record is created or updated. This is the path that resolves the root cause: it acts precisely when the previously-missing data becomes available, with re-enrolment configured so records are re-evaluated if status or price is populated later.

BACKUP — DEAL-TRIGGERED

Enrols deals entering the relevant stages, then deliberately waits before evaluating, giving the batch sync time to deliver the appointment. Without that delay, this path would simply reproduce the original defect on a different trigger.

SAFETY NET — SCHEDULED SWEEP

A nightly pass over deals still missing an association, catching anything lost to sync timing, a paused workflow, or a transient API failure. This is what prevents the backlog from silently rebuilding after remediation.

DESIGN PRINCIPLE

All three paths execute the same logic. A single rule, reached three ways, rather than three implementations that drift apart over time.

The selection rule

A contact may hold several appointments at once, some pre-booked months ahead, some cancelled, some already delivered. The engine resolves which single appointment belongs to the deal.

- 1 Restrict to the correct pipeline and to deals in the two qualifying stages, ignoring the two unrelated pipelines entirely.
- 2 Exit immediately if the deal already carries an appointment. The engine never overwrites existing links.
- 3 Filter the contact's appointments to the single status meaning "not yet delivered", out of nineteen possible values.
- 4 Discard any appointment already serving another deal, so one booking is never double-counted.
- 5 Keep only appointments dated today or later, sorted nearest first.
- 6 Select the nearest appointment carrying a sale value above zero.
- 7 If none carries a value, select the nearest one regardless.

GUARDRAIL

Where a contact holds more than one qualifying deal, the engine does not choose. It halts and flags the record for human review.

The client had confirmed only one such deal should ever exist. Rather than silently resolving a contradiction, the system surfaces it as a data-quality signal.

PROPERTIES RESOLVED

Custom object	Appointment
Date field	Start time
Status field	POS status
Value field	Final sale price

WHY IT NEEDS TO BE A RULE AT ALL

The naive approach links whatever appointment happens to be nearest. That breaks the moment a client pre-books several visits at once, which this business does routinely, or when a cancellation sits in the calendar looking superficially valid. Encoding value and status into the decision is what separates a link that is merely present from one that is correct.

Two defects caught before launch

The engine was tested against live records with the workflow held offline. Two failures surfaced that no amount of code review would have exposed, both caused by real-world data shape rather than logic.

01 Appointments carrying multiple contacts

The first test returned no deals for a contact that visibly had several. The appointment was associated to more than one contact, and the engine had trusted whichever the API returned first. Rewritten to evaluate every associated contact and proceed with the one holding a qualifying deal.

02 Date-only values compared as timestamps

A same-day appointment was rejected as already past. Appointment dates are stored without a meaningful time component, and timezone offset arithmetic was pushing them backwards across the day boundary. Comparison was moved to whole calendar days, which is what the business rule actually describes.

WHY THIS MATTERED

Both defects would have presented in production as silent non-events: deals simply failing to link, with no error raised and nothing to alert on. They would have been indistinguishable from the original problem, and would have discredited the fix. Testing against live records before activation is what made them visible.

HOW THE ENGINE WAS VALIDATED

The workflow was built and left deactivated throughout. Records were enrolled manually, one at a time, so every execution could be inspected in isolation. Each run emitted a structured decision trace, recording not only the outcome but the reasoning behind it: which contact was resolved, which deal qualified, how many appointments were considered, and which branch of the rule selected the winner.

CASES DELIBERATELY TESTED

- Single open future appointment
- Multiple future appointments, nearest unpriced
- Deal already correctly associated
- Only past-dated open appointments
- Contact holding several qualifying deals

WHAT THE TRACE EXPOSED

- Which contact the engine resolved
- Candidate appointments considered
- Rule branch that selected the winner
- Explicit reason for every skip
- Distinction between failure and correct no-op

That last distinction mattered most. A system that declines to act looks identical to one that is broken, unless it explains itself.

Remediation, safely

Repairing 885 historical records is not a code problem. It is a blast-radius problem.

The existing estate contained automation listening for exactly the event the remediation would generate. Writing 885 associations in a single pass risked cascading stage changes, internal notifications, and client-facing messages referencing appointments that had long since passed.

CONTROL	PURPOSE
Dry run	Full pass with writes disabled, producing a reviewable report of every proposed pairing before a single record is touched.
Client review	Sign-off on a sample weighted toward ambiguous cases, not just the straightforward ones.
Suppression flag	A dedicated property stamped on remediated deals, excluded from downstream automation so historical repairs never trigger live communications.
Staged rollout	A small batch verified in production before the full run proceeds.
Audit trail	Every write logged with both record identifiers, giving a complete and reversible record of the operation.

STATUS AT TIME OF WRITING

Built

ENGINE DEPLOYED
AND VALIDATED

Tested

VERIFIED AGAINST
LIVE RECORDS

Staged

REMEDICATION HELD
FOR CLIENT SIGN-OFF

WHAT THE ENGAGEMENT PRODUCED

- Root-cause diagnosis of a long-standing data integrity failure
- A single association engine replacing competing automation
- Consolidation plan for the inherited workflow estate
- Reversible remediation tooling with review gates
- A recurring safety net preventing recurrence
- A health metric making the problem visible if it returns

Manan Asim

CRM, RevOps & Automation Engineer

HUBSPOT ARCHITECTURE
SYSTEMS INTEGRATION
REVENUE OPERATIONS