# **Pre-Coding Essentials (Component: crates/vm\_pipeline/src/build\_run\_record.rs, Version/FormulaID: VM-ENGINE v0) — 60/89**

## **1) Goal & Success**

Goal: Assemble a **RunRecord** that attests to reproducibility—inputs/IDs, engine+FID, determinism settings (incl. RNG seed if used), timestamps, and pointers to produced artifacts.

Success: Object satisfies DB spec; contains everything needed to reproduce the run; no network/time dependencies; with same inputs ⇒ byte-identical after canonical serialization & SHA-256 hashing.

## **2) Scope**

In scope: Populate **RunRecord** fields; echo IDs (REG/TLY/PS), **FormulaID** & **EngineVersion**, determinism settings, UTC timestamps, result\_id and optional frontier\_map\_id.

Out of scope: Computing canonical JSON and hashes (handled in I/O layer); executing pipeline stages.

## **3) Inputs → Outputs (with schemas/IDs)**

Inputs:

From pipeline context: **FormulaID**, **EngineVersion**, reg\_id, ballot\_tally\_id, parameter\_set\_id; determinism settings (rounding/order, **rng\_seed** if used).

From previous step(s): result\_id, optional frontier\_map\_id.

From caller/orchestrator: started\_utc, finished\_utc (UTC strings).

Output: **RunRecord** entity (DB **VM-DB-007**) with id = RUN:<utc\_timestamp>-<short-hash>. Example format: RUN:2025-08-11T14-07-00Z-a1b2c3.

## **4) Entities/Tables (minimal)**

## **5) Variables (used here)**

No VM-VARs alter structure. If ties used policy random, **record** the **rng\_seed** supplied by params, per Doc 3A.

## **6) Functions (signatures only)**

rust

CopyEdit

pub fn build\_run\_record(

ctx: &PipelineCtx, // carries FID, engine, input IDs, determinism settings

result\_id: ResultId,

frontier\_id: Option<FrontierId>,

started\_utc: &str, // UTC, provided by caller

finished\_utc: &str // UTC, provided by caller

) -> RunRecord;

// helpers

fn validate\_utc(ts: &str) -> Result<()>;

fn must\_have\_seed\_if\_random(ctx: &PipelineCtx) -> Result<()>;

fn make\_run\_id(started\_utc: &str, short\_hash: &str) -> String; // "RUN:<utc>-<short>"

(Precondition: a **Result** exists.)

## **7) Algorithm Outline**

**Prechecks:** ensure result\_id present; if tie policy random, assert seed present in ctx.

**Assemble struct:** copy IDs, **FormulaID**, **EngineVersion**, determinism settings (include rng\_seed if any), timestamps, and pointers.

**Canonicalization/Hash (downstream):** writer will serialize with sorted keys, LF, UTC, then **SHA-256**; **RunRecord hash** becomes the <short-hash> part of id. (Builder provides content; I/O computes hash.)

**ID formation:** RUN:<utc\_timestamp>-<short-hash>; note: UTC in ID uses the repo’s ID-friendly timestamp form per example.

Return **RunRecord** ready for serialization and hashing.

## **8) State Flow**

**BUILD\_RESULT → BUILD\_RUN\_RECORD** (final pipeline step for audit). Single-writer; ordering deterministic.

## **9) Determinism & Numeric Rules**

No clock reads; timestamps are **inputs**. Canonical JSON: UTF-8, LF, **sorted keys**, omit unset optionals; hash with **SHA-256**. Stable ordering rules apply globally.

## **10) Edge Cases & Failure Policy**

Missing seed while policy=random ⇒ configuration error (surface upstream).

Invalid UTC strings (non-Z, not YYYY-MM-DDTHH:MM:SSZ) ⇒ reject. (Format per canonical rules.)

No result\_id ⇒ error (precondition).

## **11) Test Checklist (must pass)**

Cross-OS determinism: same inputs on Windows/macOS/Linux yield **identical RunRecord hashes**.

Hashing/ID: canonical writer produces 64-hex hash; id matches RUN:<utc>-<short>.

RNG: if random ties used, **rng\_seed** is recorded; TieLog references appear in Result (separate test).

Reproducibility: with identical inputs, builder outputs identical content ready for canonicalization.