# **Pre-Coding Essentials (Component: schemas/run\_record.schema.json, Version/FormulaID: VM-ENGINE v0) — 19/89**

## **1) Goal & Success**

Goal: JSON Schema for **RunRecord** — the signed/attested provenance of one execution.

Success: Validates RUN: ID and UTC timestamp; records **engine/version/FormulaID**, **input IDs + digests**, **tie/RNG policy**, **platform info**, and pointers to produced artifacts (RES:, optional FR:). Strict (additionalProperties:false), integers/booleans only where applicable.

## **2) Scope**

In scope: Immutable audit envelope for a single run: who/what/when, exact inputs, policies affecting outcomes, and output references.

Out of scope: The Result content itself (lives in result.json), Frontier geometry (own file), presentation/HTML.

## **3) Inputs → Outputs**

Inputs: Manifest + loaded artifacts (Registry, ParameterSet, Ballots or Tally, optional Adjacency), engine metadata.

Output: One run\_record.json per run; used by reports and determinism tests.

## **4) Entities/Fields (schema shape to encode)**

**Root**

id **(required, string)** — RUN:<timestamp>-<short-hash> (hash over canonical input bytes + engine metadata)

timestamp\_utc **(required, string, ISO-8601 Z)** — e.g., 2025-08-11T14:07:00Z

engine **(required, object)**

engine\_version **(string)** — semantic version or commit hash

formula\_id **(string)** — hex fingerprint of rule set

formula\_manifest\_sha256 *(string, 64-hex)* — digest of the normative manifest used to compute formula\_id

inputs **(required, object)**

manifest\_id *(string)* — MAN:… if a manifest was used

reg\_id **(string)** — REG:…

parameter\_set\_id **(string)** — PS:…

**exactly one of:**

ballots\_id *(string)* — dataset label if raw ballots had an ID (optional in some pipelines)

ballot\_tally\_id **(string)** — TLY:…

adjacency\_present **(boolean)**

digests **(object)** — map <relative\_path> → { sha256: <hex64> } for every input file loaded

policy **(required, object)**

tie\_policy **(string enum)** — status\_quo | deterministic\_order | random

deterministic\_order\_key *(string, const option\_order\_index when used)*

rng\_seed *(string, 64-hex;* ***required iff*** *tie\_policy = "random")*

platform **(required, object)**

os **(string)** — windows|macos|linux

arch **(string)** — x86\_64|aarch64 etc.

rustc\_version **(string)**

build\_profile **(string)** — debug|release

outputs **(required, object)**

result\_id **(string)** — RES:…

result\_sha256 **(string, 64-hex)**

frontier\_map\_id *(string)* — FR:… if produced

frontier\_map\_sha256 *(string, 64-hex)* — required iff frontier\_map\_id present

tie\_log\_summary *(object)* — optional quick stats:

deterministic\_ties **(integer ≥ 0)**

randomized\_ties **(integer ≥ 0)**

notes *(string, optional)*

## **5) Variables (validators to embed in schema)**

## **6) Functions**

(Schema only.)

## **7) Algorithm Outline (schema authoring steps)**

$schema = JSON Schema **2020-12**; set $id.

$defs: Hex64, RunId, each ID regex, DigestEntry.

Root: type: object, required: ["id","timestamp\_utc","engine","inputs","policy","platform","outputs"], additionalProperties:false.

Encode **one-of** constraint inside inputs: exactly one of ballots\_id or ballot\_tally\_id must be present.

Add if/then for policy.tie\_policy = "random" ⇒ require rng\_seed (hex64).

Require frontier\_map\_sha256 iff frontier\_map\_id present.

For digests, use additionalProperties schema { type:"object", required:["sha256"], properties:{ sha256:{pattern: hex64} }, additionalProperties:false }.

Keep all nested objects strict with additionalProperties:false.

## **8) State Flow**

After pipeline builds Result (and optional FrontierMap), engine assembles RunRecord, computing digests and embedding IDs. Report reads RunRecord (snapshot of VM-VARs is resolved via PS:; tie summary aids audit).

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

RUN: id derived from **canonical bytes** (inputs + engine metadata + FormulaID).

All digests SHA-256 (hex) over canonical bytes (UTF-8, LF, sorted keys).

RNG used **only** if tie\_policy="random"; seed recorded here for reproducibility.

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

Missing rng\_seed while tie\_policy="random" ⇒ **schema fail**.

Both ballots\_id and ballot\_tally\_id present (or neither) ⇒ **schema fail**.

Non-UTC timestamp or non-ISO format ⇒ **schema fail**.

frontier\_map\_id present without frontier\_map\_sha256 ⇒ **schema fail**.

Digests map with non-hex value ⇒ **schema fail**.

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

**Happy path (tally):** reg\_id, parameter\_set\_id, ballot\_tally\_id, result\_id, all digests hex64, UTC timestamp → **pass**.

**Random ties:** tie\_policy="random" with valid rng\_seed → **pass**; omit seed → **fail**.

**Deterministic ties:** tie\_policy="deterministic\_order" with deterministic\_order\_key="option\_order\_index" → **pass**.

**Frontier present:** includes frontier\_map\_id **and** matching sha → **pass**; omit sha → **fail**.

**ID shapes:** malformed RUN: or RES: rejected by regex.