# **Pre-Coding Essentials (Component: schemas/ballots.schema.json, Version/FormulaID: VM-ENGINE v0) — 14/89**

## **1) Goal & Success**

Goal: JSON Schema for **raw ballots** (not tallies) that the engine tabulates into UnitScores.

Success: Validates canonical top-level metadata and enforces **exactly one** ballot payload (plurality | approval | score | ranked\_irv | ranked\_condorcet); rejects malformed ballots early.

## **2) Scope**

In scope: Top-level IDs/links, one-of payload selection, per-ballot shapes by type, basic bounds (IDs, arrays, integer ranges).

Out of scope: Cross-file referential checks (unit/option existence), denominator policy, duplicates across files (handled in pipeline validation).

## **3) Inputs → Outputs**

Inputs: ballots.json (raw ballots).

Outputs: Pass/fail against schema; on pass, loader builds typed in-memory ballots for tabulation.

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

**Root object**

id **(required, string)** — TLY:<name>:v<digit+>

label **(required, string)** — human-readable name (appears in report)

reg\_id **(required, string)** — REG:<name>:<version>

ballot\_type **(required, enum)** — plurality | approval | score | ranked\_irv | ranked\_condorcet

payload **(required, object)** — **exactly one** of the following keys must be present:

plurality

approval

score

ranked\_irv

ranked\_condorcet

notes *(optional, string)*

**Payloads (mutually exclusive)**

**plurality**

ballots **(required, array)** of:

{ unit\_id: string /\* U:… \*/, vote: string /\* OPT:… \*/ }

*Blank ballots:* allow { unit\_id, vote: null } if needed (schema nullable).

**approval**

ballots **(required, array)** of:

{ unit\_id: string /\* U:… \*/, approvals: array<string /\* OPT:… \*/> }

approvals may be empty to represent a **blank** (valid) ballot.

**score**

scale\_min **(required, int)** — typically 0

scale\_max **(required, int)** — > scale\_min, typically 5

ballots **(required, array)** of:

{ unit\_id: string /\* U:… \*/, scores: object{ OPT: int } }

Each int must be in [scale\_min .. scale\_max].

Omitted options imply score = 0 unless a stricter rule is chosen in pipeline.

**ranked\_irv**

ballots **(required, array)** of:

{ unit\_id: string /\* U:… \*/, ranking: array<string /\* OPT:… \*/> }

ranking elements must be unique (schema can enforce via uniqueItems: true).

**ranked\_condorcet**

ballots **(required, array)** of the same shape as ranked\_irv.

All payloads: ballots may be empty (edge tests). Size limits enforced in pipeline (DoS guard).

## **5) Variables (validators & enums used in schema)**

## **6) Functions**

(Schema only.)

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

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

$defs: TlyId, RegId, UnitId, OptId, Score, PluralityBallot, ApprovalBallot, ScoreBallot, RankedBallot.

Root: type: object, required: ["id","label","reg\_id","ballot\_type","payload"], additionalProperties: false.

**One-of** selection:

oneOf: exactly one of payload.plurality, payload.approval, payload.score, payload.ranked\_irv, payload.ranked\_condorcet must be present.

Couple with const checks: e.g., if payload.approval exists then ballot\_type **must** equal "approval", etc.

For arrays:

minItems: 0, optionally uniqueItems: false (duplicates allowed as separate ballots).

Per-item unit\_id/OPT: fields use regex; deep referential checks deferred to pipeline.

score: add allOf ensuring scale\_max > scale\_min and each scores.\* within bounds.

Allow **nullable** vote in plurality to represent blank ballots (optional); alternatively, omit and treat as invalid in pipeline.

Keep all objects additionalProperties: false for strictness.

## **8) State Flow**

Loader validates against this schema → builds typed Ballots by mode → pipeline TABULATE computes UnitScores and turnout (valid vs blank/invalid) per unit.

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

Integers only for counts/scores; no floats.

Canonicalization (UTF-8, LF, sorted keys) enforced outside schema; stable IDs ensure reproducible hashing downstream.

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

**Multiple payloads present** → schema **fail**.

**Mismatch** (ballot\_type ≠ payload) → schema **fail**.

**Out-of-range scores** or non-unique ranking when uniqueItems: true → schema **fail**.

**Unknown fields** anywhere → schema **fail** (strict mode).

**Cross-file** issues (unknown unit\_id/OPT:) → accepted by schema, rejected in pipeline cross-validation.

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

Valid examples for each payload type (tiny 1–2 ballots) → **pass**.

File with both approval and plurality payloads → **fail**.

ballot\_type="score" with ranked\_irv payload → **fail**.

Score with scale\_min=3, scale\_max=3 or score value outside bounds → **fail**.

Ranked ballots with duplicate options in ranking (when enforced) → **fail**.

Plurality with vote: null accepted **only** if we choose to model blanks via null; otherwise schema should reject and pipeline handles blanks via tallies.