# **Pre-Coding Essentials (Component: schemas/result.schema.json, Version/FormulaID: VM-ENGINE v0) — 18/89**

## **1) Goal & Success**

Goal: JSON Schema for **Result**—the computed outcome bundle for a run.

Success: Validates RES: ID; carries input IDs (REG, TLY, PS); includes **per-unit blocks**, **aggregates**, **legitimacy gates**, and the **final label** (Decisive|Marginal|Invalid). Shapes/fields align with Docs 1/4/5/7; integers/ratios only (percentages are presentation-only).

## **2) Scope**

In scope: Top-level identifiers, per-unit summaries (scores/turnout/allocation/flags), aggregates by level, gate outcomes as exact ratios, final label (+reason), optional frontier\_map\_id, optional tie\_log.

Out of scope: Frontier geometry/content (that’s FrontierMap), provenance timestamps (that’s RunRecord), rendering/rounding (Doc 7 handles presentation).

## **3) Inputs → Outputs**

Inputs (by reference): reg\_id (REG:...), ballot\_tally\_id (TLY:...), parameter\_set\_id (PS:...).

Output: A single strict Result JSON object; report consumes it (plus optional FrontierMap and RunRecord).

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

**Root**

id **(required, string)** — RES:<short-hash>

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

ballot\_tally\_id **(required, string)** — TLY:<...>

parameter\_set\_id **(required, string)** — PS:<...>

label **(required, enum)** — Decisive | Marginal | Invalid

label\_reason *(optional, string)* — short rationale used in report

aggregates **(required, object)** — by level

units **(required, array)** — list of **UnitBlock**

gates **(required, object)** — quorum / majority / double-majority / symmetry outcomes

tie\_log *(optional, array)* — entries from tie resolution (if any)

frontier\_map\_id *(optional, string)* — FR:<...> when mapping run produced one.

**UnitBlock (array items)**

unit\_id **(required, string)** — U:REG:...

turnout **(required, object)** — { ballots\_cast:int≥0, invalid\_or\_blank:int≥0, valid\_ballots:int≥0 }

scores *(required for non-ranked inputs)* — map OPT:... → int≥0 (plurality=votes, approval=approvals, score=score\_sum)

allocation **(required, object)** — map OPT:... → int (seats) **or** power\_pct:int (WTA 100)

flags **(required, object)** — { unit\_data\_ok:bool, unit\_quorum\_met:bool, unit\_pr\_threshold\_met:bool, protected\_override\_used:bool, mediation\_flagged:bool }.

**Aggregates (by level)**

Object keyed by level (country, region, district used), each with:

totals — map OPT:... → int (seats or votes as applicable)

shares — map OPT:... → ratio{num:int, den:int}

turnout — { ballots\_cast:int, invalid\_or\_blank:int, valid\_ballots:int, eligible\_roll:int }

weighting\_method — echo of VM-VAR-030 for clarity.

**Gates (legitimacy outcomes)**

quorum — { observed:ratio, threshold\_pct:int, pass:bool }

majority — { observed:ratio, threshold\_pct:int, pass:bool }

double\_majority — { national: {observed:ratio, threshold\_pct:int, pass:bool}, regional: {observed:ratio, threshold\_pct:int, pass:bool}, pass:bool }

symmetry — { pass:bool }

**Ratios are integers only**; reporting does the 1-decimal rendering. Approval gate’s observed value is the **approval rate** (approvals\_for\_change / valid\_ballots).

## **5) Variables (validators & enums to embed in the schema)**

## **6) Functions**

(Schema only.)

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

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

$defs: ResId, RegId, TlyId, PsId, UnitId, OptId, Ratio, UnitBlock, GateOutcome.

Root object: required = ["id","reg\_id","ballot\_tally\_id","parameter\_set\_id","label","aggregates","units","gates"], additionalProperties:false.

**UnitBlock**: strict object; integers ≥0; allocation either seats map **or** WTA power (choose one via oneOf).

**Aggregates**: require turnout and either totals or shares (allow both); ratios encoded as {num,den} ints.

**Gates**: encode shapes above; require integers for thresholds; ratios only.

Optional tie\_log array item schema: { context:string, candidates:array<OPT>, policy:enum, order\_or\_seed:string, winner:OPT }. (Produced only when ties block decisions.)

Non-normative $comment: arrays **should** be sorted (Units by unit\_id; Options by order\_index then id)—enforced in code for determinism.

## **8) State Flow**

Populated by **BUILD\_RESULT** after LABEL step; then **RunRecord** is built pointing to it. Reports read Result (+ optional FrontierMap, RunRecord).

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

**Integers & ratios only**; **no floats** inside Result.

Percentages are derived at report time; **round half to even** only at defined comparison points; report shows **one decimal**.

Ordering is stable: Units by **Unit ID**, Options by **order\_index then ID**; canonical JSON (UTF-8, LF, sorted keys).

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

**Validation failed** earlier ⇒ label="Invalid", gates panel contains N/A/Fail as per report rules; frontier omitted.

**Gates failed** ⇒ label="Invalid"; frontier omitted.

**IRV/Condorcet**: carry round logs/pairwise *only* via audit/TieLog if used; continuing-denominator policy is fixed.

**WTA**: allocation uses 100% power for the winner; schema must allow either seat map or power%.

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

Minimal **Decisive** result with one unit, Sainte-Laguë seats in canonical order; label present.

aggregates.turnout.valid\_ballots = ballots\_cast - invalid\_or\_blank