# **Pre-Coding Essentials (Component: schemas/division\_registry.schema.json, Version/FormulaID: VM-ENGINE v0) — 13/89**

## **1) Goal & Success**

Goal: JSON Schema that locks the **DivisionRegistry** structure used by all runs/tests.

Success: Validates canonical IDs, required provenance fields, unit fields/constraints, and optional adjacency; rejects malformed/ambiguous registries.

## **2) Scope**

In scope: Top-level registry object, Units[], optional Adjacency[], canonical ID formats, basic numeric bounds.

Out of scope: Cross-document rules that require parameters (e.g., population weighting on/off), graph cycle detection (done in pipeline validation).

## **3) Inputs → Outputs**

Inputs: Registry JSON (division\_registry.json).

Outputs: Pass/fail against this schema; error paths precise; downstream loader gets strongly-typed data.

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

**Root object**

id **(required, string)** — REG:<name>:<version>

name **(required, string)** — human label

version **(required, string)** — version tag included in id

provenance **(required, object)**:

source **(required, string)**

published\_date **(required, YYYY or ISO date string)**

notes *(optional, string)*

units **(required, array, minItems ≥ 1)** — list of **Unit**

adjacency *(optional, array)* — list of **Adjacency**

**Unit**

id **(required, string)** — U:<REG\_ID>:<path>

name **(required, string)**

level **(required, string)** — e.g., Country, Region, District (free text; constrained by docs, not enum here)

parent *(nullable, string)* — null for root; else a Unit.id

magnitude **(required, integer ≥ 1)** — seats/power slots

eligible\_roll **(required, integer ≥ 0)**

population\_baseline *(integer ≥ 0; required? see note)*

population\_baseline\_year *(string “YYYY”; required? see note)*

protected\_area *(optional, boolean)*

**Note (baseline fields):** keep them **present but optional** at schema level; pipeline cross-validation will **require** them when weighting method = population\_baseline.

**Adjacency**

unit\_id\_a **(required, string)** — must reference a Unit.id

unit\_id\_b **(required, string)**

type **(required, enum)** — land | bridge | water

notes *(optional, string)*

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

## **6) Functions**

(Schema only.)

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

$schema draft: use JSON Schema Draft 2020-12.

Define $defs:

RegId, UnitId, DateYyyy, AdjType.

Root: type: object, required: ["id","name","version","provenance","units"], additionalProperties: false.

units: array of $defs.Unit with minItems: 1 and uniqueItems: true (by deep equal; ID uniqueness rechecked in pipeline).

adjacency: array of $defs.Adj (optional); allow empty.

Unit object: required fields as above; numeric bounds (minimum), parent nullable: true.

Add **format/regex** checks for IDs & date; keep **cross-references** (parent/adjacency to existing IDs) for pipeline validation, not schema (JSON Schema can’t enforce forward refs easily).

For canonicalization: add a **non-normative** $comment stating LF/UTF-8/sorted keys policy (enforced elsewhere).

## **8) State Flow**

Loader: schema-validate → on success, construct in-memory model → pipeline does cross-validation (root count=1, no cycles, parent existence, adjacency references).

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

Determinism supported by: stable IDs, LF-only JSON (outside schema), integer types for counts.

No rounding/floats here.

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

**Root count:** exactly one unit with parent = null (checked in pipeline).

**Parent loops:** detect cycles in pipeline; schema only shapes data.

**WTA constraint:** if later allocation\_method = winner\_take\_all, pipeline ensures **all involved units have magnitude = 1**.

**Baseline missing:** if weighting by population is selected, fail in validation when population\_baseline(\_year) absent.

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

**Happy path:** minimal registry: 1 root unit (magnitude=1, roll provided), valid REG/U IDs, provenance present → passes.

**Bad IDs:** lowercase reg: or malformed U: → schema **fails** on regex.

**Bad numeric bounds:** magnitude=0 or negative rolls → schema **fails**.

**Adjacency type:** any value outside land|bridge|water → schema **fails**.

**Cross-ref checks (pipeline tests):**

Multiple roots or no root → **fail**.

parent points to non-existent ID → **fail**.

Adjacency references unknown units → **fail**.

Cycle in parents → **fail**.

**Authoring note (implementation hints):**

Keep the schema **strict** (additionalProperties: false) in all objects.

Prefer **regex** for ID surface shape; deep validation (e.g., that the Unit.id embeds the same REG: as root) happens in code.

Include $id: "https://…/schemas/division\_registry.schema.json" for stable tooling references.