# **Pre-Coding Essentials (Component: crates/vm\_core/src/entities.rs, Version/FormulaID: VM-ENGINE v0) — 24/89**

## **1) Goal & Success**

Goal: Define the **domain types** used across the engine (registry, units, options, adjacency, common blocks like turnout/labels), with **no I/O** and stable semantics aligned to Docs 1–7 & Annex.

Success: Types compile on all targets; invariants are encoded (e.g., magnitude ≥ 1); sorting helpers exist (units by UnitId, options by order\_index then OptionId); no JSON/FS dependencies.

## **2) Scope**

In scope: Structs/enums for **DivisionRegistry**, **Unit**, **OptionItem**, **Adjacency**, **Provenance**, **Turnout**, **DecisivenessLabel**; thin constructors/validators.

Out of scope: Parameter variables (live in variables.rs), ID parsing (in ids.rs), serialization (in vm\_io), pipeline ephemera (lives in vm\_pipeline), report rendering.

## **3) Inputs → Outputs**

Inputs: none at runtime (library definitions).

Outputs: Strongly-typed values used by vm\_io (decode/encode), vm\_algo (compute), vm\_pipeline (state machine), vm\_report (mapping).

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

*(IDs come from ids.rs; variables from variables.rs.)*

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

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

**DivisionRegistry**

pub fn new(id:RegId, name:String, version:String, provenance:Provenance, units:Vec<Unit>, adjacency:Vec<Adjacency>) -> Result<Self, EntityError>

pub fn root\_units(&self) -> impl Iterator<Item=&Unit>

pub fn unit(&self, id:&UnitId) -> Option<&Unit>

**Unit**

pub fn new(...) -> Result<Self, EntityError> *(checks magnitude≥1, non-negative rolls/baselines, parent≠self)*

pub fn is\_root(&self) -> bool

**OptionItem**

pub fn new(id:OptionId, display\_name:String, order\_index:u16, is\_status\_quo:bool) -> Result<Self, EntityError>

**Adjacency**

pub fn new(a:UnitId, b:UnitId, edge:EdgeType) -> Result<Self, EntityError> *(reject a==b)*

**Sorting helpers (deterministic)**

pub fn sort\_units\_by\_id(units:&mut [Unit])

pub fn sort\_options\_canonical(opts:&mut [OptionItem]) // by order\_index then id

## **7) Algorithm Outline (implementation plan)**

Define **data enums**:

EdgeType = { Land, Bridge, Water }

DecisivenessLabel = { Decisive, Marginal, Invalid }

YyyyOrIsoDate as a tiny tagged enum or validated String newtype.

Define **structs** as above; derive Clone, Debug, Eq, PartialEq, Ord, PartialOrd, Hash where meaningful.

Implement **constructors** that enforce local invariants:

magnitude≥1; eligible\_roll≥0.

If population\_baseline.is\_some() then population\_baseline\_year.is\_some() (pairing rule).

Adjacency: a != b.

Provide **deterministic sort** helpers:

Units by UnitId (lexicographic).

Options by order\_index then OptionId.

Keep **no serialization** code here. serde derives gated behind feature="serde" with #[serde(transparent)] only for simple newtypes.

## **8) State Flow (very short)**

vm\_io constructs these from validated JSON; vm\_algo consumes them; vm\_pipeline aggregates and labels results; vm\_report reads finalized results (not defined here).

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

**Stable total orders** exposed via helpers (Units by ID; Options by order\_index then ID).

**No floats**; counts and baselines are integers.

Presentation rounding happens in report layer; nothing here rounds.

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

Multiple roots or zero roots are **not** validated here; leave to pipeline **VALIDATE** step.

Missing baseline fields are allowed here (optional) but become **pipeline errors** when population weighting is enabled.

valid\_ballots should equal ballots\_cast - invalid\_or\_blank; constructor for Turnout enforces that or computes it.

Adjacency duplicates or cross-registry edges are checked later (pipeline).

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

Unit::new rejects magnitude=0, accepts magnitude≥1.

Turnout::new(100, 7) yields valid\_ballots=93; negative-like underflows (u64) impossible by API.

Option sort: (order\_index,id) total and stable; equal order\_index breaks ties by OptionId.

Adjacency::new(a,b,…) rejects a==b.

Sorting helpers produce the same order on all OS/arch.

Optional baseline pair invariant enforced (value ↔ year).

**Notes for coding**

Keep this file purely **domain** (no path logic, no JSON).

Document each invariant with a doc comment and a unit test.

Public API should be minimal; most mutation via constructors to keep invariants true.