# **Pre-Coding Essentials (Component: schemas/frontier\_map.schema.json, Version/FormulaID: VM-ENGINE v0) — 20/89**

## **1) Goal & Success**

Goal: JSON Schema for **FrontierMap**—per-Unit frontier status and contiguity outcomes derived from the run.

Success: Validates FR: ID; echoes frontier-related parameters; lists every Unit with its **status**, **observed support (ratio)**, and **flags** (contiguity/mediation/protection/enclave). Strict (additionalProperties:false).

## **2) Scope**

In scope: Top-level IDs/links, chosen frontier mode & knobs, per-Unit status block, required flags, optional audit crumbs.

Out of scope: Computing statuses/contiguity (done in pipeline), geometry/topology beyond identifiers.

## **3) Inputs → Outputs**

Inputs (by reference): DivisionRegistry (REG:), ParameterSet (PS:) frontier variables; Aggregates/UnitScores used during mapping.

Output: One frontier\_map.json object (optionally referenced by RunRecord and report).

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

**Root**

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

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

parameter\_set\_id **(required, string)** — PS:<...> *(trace which knobs were active)*

mode **(required, enum)** — none | sliding\_scale | autonomy\_ladder

contiguity\_edge\_types **(required, array enum)** — items in { "land","bridge","water" }, uniqueItems:true, minItems:1

corridor\_policy **(required, enum)** — none | ferry\_allowed | corridor\_required

bands *(required iff mode != "none", array)* — each { min\_pct:int 0..100, max\_pct:int 0..100, status:string } with min\_pct ≤ max\_pct

units **(required, array)** — list of **UnitFrontier** (see below)

notes *(optional, string)*

**UnitFrontier (array items)**

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

support **(required, object)** — { num:int ≥0, den:int ≥1 } *(observed support used for mapping; exact meaning follows Doc 4 rules—e.g., approval* ***rate*** *for approval ballots)*

status **(required, string)** — one of the bands[].status values (or "none" if mode="none")

flags **(required, object)**:

contiguity\_ok **(bool)**

mediation\_flagged **(bool)**

protected\_override\_used **(bool)**

enclave **(bool)**

adjacency\_summary *(optional, object)*:

used\_edges *(array enum)* — subset of {land,bridge,water} actually linking this unit’s status cluster

corridor\_used *(bool, optional)* — true if corridor logic was required

reasons *(optional, array<string>)* — short machine-readable reason codes for failed checks or mediations

Arrays should be **sorted** (Units by unit\_id); schema can’t enforce order—loader will.

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

## **6) Functions**

(Schema only.)

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

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

$defs: FrId, RegId, PsId, Ratio, Edge, Band, UnitFrontier.

Root: type:"object", required = ["id","reg\_id","parameter\_set\_id","mode","contiguity\_edge\_types","corridor\_policy","units"], additionalProperties:false.

Conditional: if mode != "none" ⇒ require bands with at least one item; each band validates % bounds; uniqueness/non-overlap checked in pipeline.

units: array of strict UnitFrontier objects; all integer minima set; status is string (pipeline ensures it matches a bands[].status or "none").

Keep all objects additionalProperties:false.

## **8) State Flow**

Pipeline **MAP\_FRONTIER** constructs this object after gates; **RunRecord** may reference FR:; report reads it to render maps/status tables.

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

Support stored as **exact ratio {num,den}**; no floats.

Canonical JSON rules (UTF-8, LF, sorted keys) apply at I/O; stable Unit ordering for reproducible hashing.

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

mode="none" ⇒ bands **must be absent**; all status values should be "none".

Overlapping/out-of-order bands ⇒ **pipeline** fails validation (schema only checks shape/bounds).

contiguity\_edge\_types empty or includes unknown strings ⇒ **schema fail**.

Missing support or den=0 ⇒ **schema fail**.

Units present in registry but missing here: allowed? **No**—pipeline should ensure **one entry per Unit**.

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

**Happy path:** mode="sliding\_scale", valid bands, three units with sorted unit\_ids, ratios {num,den}, flags booleans → **pass**.

**None mode:** mode="none" with no bands, statuses "none" → **pass**.

**Bad band:** min\_pct>max\_pct → **schema fail**.

**Unknown edge:** used\_edges:["air"] → **schema fail**.

**Zero denominator in support** → **schema fail**.

**Pipeline cross-checks:** duplicate/missing units; band overlap; status not in bands; contiguity inconsistencies → **pipeline fail**.