Annex A — Variable Canonical Reference Table.md

File

Annex B — Canonical Test Pack.md

File

Annex C — Glossary & Definitions.md

File

Doc 1 — Database Specification (Entities, Fields, Relationships).md

File

Doc 2 — Common Variables Specification (Core, Operational Defaults, Advanced Controls).md

File

Doc 3 — Technical Platform & Release Policy.md

File

Doc 4 — Algorithm Specification (Steps, Allocation, Gates & Edge Cases).md

File

Doc 5 — Processing Pipeline Specification (State Machine & Functions).md

File

Doc 6 — Test Specifications (Allocation, Gates, Frontier & Determinism).md

File

Doc 7 — Reporting Specification (Structure, Templates & Visual Rules).md

File

Here is my doc dor the project. I will give you skeletton files you will validate alignment. Fixed reference is the 10 docs i give you now

Thought for 1m 49s

**ChatGPT said:**

Confirmed. I’ve loaded the 10 files you just uploaded and will treat them as the fixed reference set for alignment checks.

When you send a skeleton file, I’ll validate it against these rules and report PASS/FAIL plus precise deltas:

1. VM-VAR membership & FID scope  
   • Included vs Excluded ranges, and manifest build rules.  
   • Tie controls: 050 in FID; 052 excluded; 051 reserved.  
   • Doc 2A pointer reiterating 050 (Included) vs 052 (Excluded) and full Included ranges.
2. Deterministic ordering & step order  
   • Units by ascending unit\_id; options by order\_index (then option\_id); allocations mirror registry order.  
   • Pipeline S1–S7, including where ties happen and labels computed (presentation-only).
3. Canonicalization & IDs / FID linkage  
   • Canonical JSON (UTF-8, LF, sorted keys) and manifest hashing expectations.  
   • Artifacts must carry formula\_id from the Normative Manifest.
4. Ties & RNG usage  
   • Policies and constraints: deterministic\_order uses order\_index; random uses seed 052; consume exactly k draws for k-way ties; only draw if a tie actually occurs.  
   • Seed echo rules and RNG profile pinned by Annex B.  
   • Conformance (seed doesn’t change FID; policy 050 does).
5. Frontier & FrontierMap  
   • Outcome-affecting controls 040–042 (+047–049) and deterministic evaluation; band\_met/band\_value semantics.  
   • Emission of FrontierMap only when 034=true and frontier active (presentation-only effect on presence).
6. Presentation toggles are non-FID  
   • 032–035, 060–062 never alter allocations nor FID; labels are presentation.  
   • Excluded-set conformance (toggling these cannot change canonical artifacts, except optional FrontierMap presence).
7. Test harness invariants (for any fixtures embedded in a skeleton)  
   • Canonical form & IDs; FID recomputation equals artifacts; tie expectations and draw counts; reproducibility.

What I’ll return on each skeleton:

* Verdict: PASS/FAIL
* Violations: numbered list with the exact rule and citation (as above)
* Minimal fix suggestions (only to resolve the violation; no new features)

Send the first skeleton when ready.

**You said:**

05 - .gitignore.md

File

01 - Workspace Manifest Cargo.toml.md

File

02 - rust-toolchain.toml.md

File

03 - .cargo config.toml.md

File

04 - Makefile.md

File

is this well aligned on reference?

Thought for 1m 54s

**ChatGPT said:**

Short answer: yes — these five skeletons are aligned with the fixed reference set. I’d mark all **PASS**, with a couple of small, non-blocking notes.

**01 — Workspace Manifest (Cargo.toml)** — **PASS**

* Matches determinism/portability aims: byte-identical outputs across OS/arch and separation of engine vs. renderer concerns. 【】
* No conflict with FID rules or presentation toggles (those live outside FID). 【】
* Note (optional): keeping Tauri/UI out of default-members is consistent with the engine-only determinism contract. No change required.

**02 — rust-toolchain.toml** — **PASS**

* Exact toolchain pin supports cross-OS byte-identity and is expected at release (“pinned toolchain”). 【】【】

**03 — .cargo/config.toml** — **PASS**

* Offline builds + vendoring align with the “no network during official runs / deterministic platform” requirements. 【】【】
* No normative conflict with profiles being in Cargo.toml (correct separation).

**04 — Makefile** — **PASS (with 2 tiny checks to keep)**

* Targets mirror the canonical pipeline and artifacts (Result, RunRecord, optional FrontierMap). 【】【】
* Determinism and RNG handling in fixture/verify steps match test-pack expectations (seed echoed only if a random tie actually occurs; re-runs are byte-identical). 【】【】
* Keep this check explicit in CI: emit **FrontierMap** only when 034=true and frontier executes; FID must remain unchanged by 034. 【】
* Packaging as reproducible archives is fine (non-normative, but consistent with the determinism section).

**05 — .gitignore** — **PASS**

* Ignoring build/cache/IDE noise while **tracking** normative inputs (schemas, fixtures, vendor) is aligned with platform & publication rules (spec bundle + canonical test pack are part of the release and must be versioned). 【】
* Only canonical artifacts are result.json, run\_record.json, and optional frontier\_map.json; ignoring other transient outputs is consistent. 【】
* Presentation/report toggles (032–035) and 052 (tie seed) are **non-FID**; your ignores/comments don’t interfere with that boundary. 【】【】

If you want, I can run the same pass on the next skeletons and flag any deltas the moment they appear.

Sources

**You said:**

15 - schemas ballot\_tally.schema.json.md

File

06 - .gitattributes.md

File

07 - .editorconfig.md

File

08 - .pre-commit-config.yaml.md

File

09 - README.md.md

File

10 - LICENSE.md

File

11 - CONTRIBUTING.md.md

File

12 - SECURITY.md.md

File

13 - schemas division\_registry.schema.json.md

File

14 - schemas ballots.schema.json.md

File

check those

Thought for 2m 47s

**ChatGPT said:**

Short answer: most repo scaffolding aligns; three schema items need changes to match the fixed specs.

**Verdicts (per file)**

* **06 – .gitattributes**: **PASS.** Enforces LF-only and treats binaries as binary, which supports canonical JSON and cross-OS determinism.
* **07 – .editorconfig**: **PASS.** UTF-8 + LF + final newline mirrors canonicalization rules.
* **08 – .pre-commit-config.yaml**: **PASS.** Fast LF/whitespace, JSON sanity, cargo fmt, plus a deterministic smoke test is consistent with the platform rules and the RNG policy (seed is only relevant when ties are randomized).
* **09 – README.md**: **ADJUST.** Overall structure is fine (offline/determinism, quickstart, annex-B example). One sentence is off: “Manifest must provide exactly one of **ballots** or precomputed **tally**.” The engine’s contract is **registry + tally + params**; Annex-B’s file contract fixes this to those three inputs only. Please drop the “ballots” alternative.
* **10 – LICENSE**: **PASS (non-normative).** Licensing isn’t governed by Docs 1–7; nothing here conflicts with the technical spec.
* **11 – CONTRIBUTING.md**: **PASS.** “Spec-first,” FID rules, determinism, ordering and reporting notes line up with Doc 1 & Doc 7.
* **12 – SECURITY.md**: **PASS.** Offline/no-network, schema-first validation, seeded RNG guidance all match the platform policy.
* **13 – schemas/division\_registry.schema.json**: **FAIL (shape mismatch).** Doc 1B defines a minimal, **strict** registry shape:
  + Top level: { "schema\_version": "1.x", "units": [...] } (no id/name/version/provenance/adjacency at this layer).
  + units[] must sort by unit\_id and each unit carries options[] sorted by order\_index.  
    Your outline adds fields and omits options[] at the unit level. Please align to Doc 1B’s envelope and keep extra provenance/adjacency out of the normative schema (those can be validated later in the pipeline if you need them).
* **14 – schemas/ballots.schema.json**: **FAIL (out of scope + ID prefix).** The engine/test-pack contract uses **aggregated tallies**, not raw ballot files; test case inputs are exactly registry.json, tally.json, params.json. If you keep a ballots schema for upstream ingestion, mark it clearly **non-normative** and don’t imply the engine accepts it directly. Also, the id prefix TLY: is reserved for tallies—change to something like BAL: to avoid confusion.
* **15 – schemas/ballot\_tally.schema.json**: **ADJUST (naming & structure).** Conceptually aligned (single file per mode; non-negativity; basic sanity), but Doc 1B’s catalog expects per-unit **totals** and an **array** of options ordered by order\_index, not an object map. Also, Doc 1B’s terminology uses valid/invalid counts rather than ballots\_cast/invalid\_or\_blank. Please:
  + Represent options as an **array** (options[]) ordered by registry order\_index;
  + Use the spec’s naming (totals.valid\_ballots, totals.invalid\_ballots) so Annex-B cases validate cleanly.  
    That keeps ordering and hashing behavior consistent with the normative examples/tests.

**Why these changes matter**

* Canonicalization & ordering are determinism primitives; deviating shapes or maps vs arrays can change byte-level outputs even if math is identical. The platform mandates stable ordering + canonical JSON for hashing.
* The official test pack (Annex B) is wired to registry.json + tally.json + params.json. Schemas that differ won’t validate against those fixtures or produce the expected hashes/FID.

If you want, I can rewrite 13/14/15 to the Doc 1B envelopes and keep your extra metadata as pipeline-level validation (non-normative).

**You said:**

16 - schemas parameter\_set.schema.json.md

File

17 - schemas manifest.schema.json.md

File

18 - schemas result.schema.json.md

File

19 - schemas run\_record.schema.json.md

File

20 - schemas frontier\_map.schema.json.md

File

21 - crates vm\_core Cargo.toml.md

File

22 - crates vm\_core src lib.rs.md

File

23 - crates vm\_core src ids.rs.md

File

24 - crates vm\_core src entities.rs.md

File

25 - crates vm\_core src variables.rs.md

File

check those

Thought for 3m 15s

**ChatGPT said:**

Here’s an alignment check of the new batch against the 10 fixed references. Verdict per file with the minimum deltas to bring each one into spec:

**16 – schemas/parameter\_set.schema.json → FAIL (rework required)**

* Seed/type: spec’s tie seed **VM-VAR-052 is an integer ≥ 0**, not a 64-hex string. Make 052 integer and keep it **Excluded** from FID; **050 tie\_policy** remains outcome-affecting and in FID.
* “on/off” strings: booleans should be real JSON booleans where variables are typed as boolean in Annex A / Doc 1B; don’t serialize booleans as "on"|"off". (Follow Annex A domains as authoritative.)
* Variable identities: 073 is **algorithm\_variant** (enum), not an “executive toggle”. Remove any executive\_enabled notion and model 073 per Doc 2C/Annex A.
* Families & conditionals: tie and frontier gates/conditionals are controlled by the **registered VM-VAR IDs**; avoid inventing names like “weighting\_method” for 030/031 (they are eligibility\_override\_list and ballot\_integrity\_floor).
* Canonicalization/FID note is good—keep the reminder but ensure **Included set** matches Annex A (001–007, 010–017, 020–031 incl. 021/029–031, 040–049, 050, 073; Excluded: 032–035, 052, 060–062).  
  (Your current outline explicitly requires a 64-hex seed, uses string booleans, and mislabels 073. )

**17 – schemas/manifest.schema.json → ADJUST**

* Inputs for runs/tests are **registry + tally + params only**. The loader function takes {registry, tally, params}; there’s **no “raw ballots”** path in the normative pipeline, and Annex B test cases list exactly those three files. Drop ballots\_path and require ballot\_tally\_path.
* Optional digests{path→sha256} is fine; enforce 64-hex.  
  (Your schema currently allows either ballots\_path or ballot\_tally\_path; that conflicts with the pipeline contract for canonical runs.)

**18 – schemas/result.schema.json → ADJUST**

* Must include **formula\_id** on the Result (computed from the Normative Manifest); this is recorded on Result and RunRecord.
* **Input references belong in RunRecord**, not Result. Remove reg\_id, ballot\_tally\_id, parameter\_set\_id from Result root.
* **Shares/ratios**: Doc 1B stores **shares as JSON numbers** (engine precision), not {num,den} objects. Keep integer math in code, but emit numbers here.
* Tie events live in **RunRecord** (ties[]); drop tie\_log from Result.  
  (Your outline currently omits formula\_id, carries input IDs, models shares as {num,den}, and puts a tie log in Result. )

**19 – schemas/run\_record.schema.json → ADJUST**

* Add the **normative manifest digest** field (nm digest) to support FID verification; record formula\_id here.
* Engine metadata must include vendor/name/version/build per Doc 1B; reflect that in engine{}.
* Determinism block: Keep **tie\_policy** and only include **rng\_seed** if policy was random (no extra “deterministic\_order\_key” knob exists—deterministic order is **always** by option.order\_index).
* Inputs: keep canonical **64-hex** digests for all inputs.  
  (Your current outline has a spare “deterministic\_order\_key” and lacks nm digest/vendor/name/build.)

**20 – schemas/frontier\_map.schema.json → FAIL (rework required)**

* Per Doc 1B, the per-unit diagnostics are **band\_met: boolean** and **band\_value: number**; use those normalized names and types (not a free-form status or a {num,den} support object).
* Emit FrontierMap **only if VM-VAR-034 = true** and frontier executed; that’s a pipeline output gated by the toggle.  
  (Your outline uses status strings and ratio objects for “support”; rename/reshape to band\_met/band\_value and align emission conditions. )

**21 – crates/vm\_core/Cargo.toml → PASS**

* Lean core, optional serde, pinned rand\_chacha/rand\_core, no I/O deps—matches spec layering and determinism guidance.

**22 – crates/vm\_core/src/lib.rs → PASS (note)**

* Public surface (IDs, entities, variables, ratio compare, seeded RNG) matches the layering; ensure downstream uses numbers for shares when serializing Result, per Doc 1B.

**23 – crates/vm\_core/src/ids.rs → ADJUST (tighten & de-prefix)**

* **RES/RUN/FR formats** must enforce **64-hex** in regexes: ^RES:[0-9a-f]{64}$, ^RUN:\d{4}-\d{2}-\d{2}T\d{2}-\d{2}-\d{2}Z-[0-9a-f]{64}$, ^FR:[0-9a-f]{64}$.
* **unit\_id / option\_id** are not prefixed (U:/OPT:) in the spec; they are free strings within the allowed character class and length. Don’t require U:REG:… or OPT:… in parsers.  
  (Your current plan allows arbitrary [A-Za-z0-9.\_-]+ after RES:/FR: and introduces U:/OPT: forms not present in Doc 1B; tighten to the canonical shapes and relax unit/option to spec’s allowed charset.)

**24 – crates/vm\_core/src/entities.rs → PASS (minor notes)**

* Deterministic sort helpers and invariants align. Keep order\_index uniqueness and canonical ordering as hard rules.

**25 – crates/vm\_core/src/variables.rs → ADJUST**

* Model **050 tie\_policy (enum)** and **052 tie\_seed (integer)** exactly; validation: seed required iff policy = random; but 052 remains **Excluded** from FID.
* Frontier tuning 047–049 domains should follow Annex A (window in [0,1], backoff/strictness enums).
* Ensure the **Included set** used for FID matches Annex A; presentation vars (032–035, 060–062) must never affect FID.

If you want, I can apply these deltas directly to the skeletons (schemas + ids.rs + variables.rs + result schema) so the next test run lines up with the Canonical Test Pack.

Sources

**You said:**

.

2/2