**Annex A — Variable Canonical Reference Table (Updated)**

**Part 1/3: Scope, Definitions, and FID Coverage**

**Status.** Normative. Required for all releases and for tests that verify the Formula ID (FID) and canonical artifact hashing.  
**Purpose.** Specify exactly **what is hashed** to produce the **Formula ID** and which rules are treated as **fixed constants**, so byte-identical Results/RunRecords are possible across platforms.

**0) Definitions**

* **Formula ID (FID):** Cryptographic fingerprint of the **normative rule set** (not of any dataset or run instance).
* **Canonical Serialization:** Deterministic byte representation used for the FID **and** for hashing Result/RunRecord artifacts.
* **Normative Manifest (NM):** Machine-readable bundle that enumerates all outcome-affecting variables (IDs, domains, defaults) **plus** the fixed algorithmic constants.

**1) What the FID covers**

**1.1 Outcome-affecting variables (ID ranges)**

Include the **existence**, **domain/semantics**, and **default value** for:

* **Ballot:** VM-VAR-001 … 007
* **Allocation & MMP:** VM-VAR-010 … 017
* **Gates & Families:** VM-VAR-020 … 029 **(+** VM-VAR-021\_scope **)**
* **Aggregation:** VM-VAR-030 … 031
* **Frontier & Contiguity:** VM-VAR-040 … 048
* **Ties & RNG:** VM-VAR-032 … 033 *(renumbered; supersedes legacy 050–052)*
* **Executive toggle:** VM-VAR-073

**Explicit exclusion:** Presentation/reporting controls (e.g., label display thresholds/policies) are **not** part of the FID.

**1.2 Fixed algorithmic rules (constants)**

These are **constants** included in the FID (not user-tunable):

* **Approval gate denominator:** legitimacy support % = approvals\_for\_change / valid\_ballots.
* **IRV exhaustion policy:** reduce\_continuing\_denominator.
* **Rounding for internal comparisons:** *round half to even* at defined decision points.
* **Allowed allocation families:**  
  winner\_take\_all, proportional\_favor\_big (D’Hondt), proportional\_favor\_small (Sainte-Laguë), largest\_remainder, mixed\_local\_correction (MMP).
* **MMP sequencing (order is fixed):** local seats → target shares → deficits → top-ups (per mlc\_correction\_level) → overhang handling per overhang\_policy/total\_seats\_model.
* **Contiguity edge types:** the set {land, bridge, water} and their semantics.

**1.3 What the FID does not cover**

* Data schemas / entity lists (Doc 1), pipeline function names (Doc 5), report templates/precision text (Doc 7), performance profiles, UI text/translations.
* Any **run-time** parameter values chosen for a specific run (those live in the RunRecord, not in the FID).
* **Presentation-only** or **operational** toggles that do not change outcomes (e.g., report labeling policies).

**Annex A — Variable Canonical Reference Table (Updated)**

**Part 2/3: Normative Manifest & Canonical Serialization**

**2) Building the Normative Manifest (NM)**

Construct a single JSON object that captures **all outcome-affecting variables** and **fixed constants** in a canonical layout. This NM is the **sole input** for computing the **Formula ID (FID)**.

**2.1 Canonical field order (top level keys, in this exact order)**

1. "schema\_version" — string (e.g., "NM-1.0").
2. "variables" — array (sorted by **VM-VAR ID** ascending). Each item has:
   * "id" (e.g., "VM-VAR-022"),
   * "name" (stable snake\_case),
   * "domain" (closed set or numeric range),
   * "default",
   * "notes" (brief semantics; no markdown).
3. "constants" — object with keys:
   * "approval\_gate\_denominator",
   * "irv\_exhaustion\_policy",
   * "rounding\_rule",
   * "allocation\_families" (array; fixed order),
   * "mmp\_sequence" (array; fixed order),
   * "contiguity\_edge\_types" (array; fixed set).
4. "compat" — object with keys:
   * "reserved\_ids" (arrays by ID range),
   * "fid\_policy\_version" (string selecting the hash policy).
5. "origin" — object (informative, **excluded from FID**):
   * "docs\_commit\_refs" (map of doc → VCS ref),
   * "generated\_at\_utc" (UTC ISO-8601).

**FID scope:** Only fields **1–4** are hashed. "origin" is carried for traceability but **never** included in the FID computation.

**2.2 FID computation (high level)**

1. Assemble the NM with fields **1–4** only.
2. **Canonicalize** to JSON per §3.
3. Hash the bytes using the algorithm implied by "fid\_policy\_version" (e.g., the policy may select SHA-256).
4. Encode the digest as the **FID** and embed it where required (e.g., RunRecord).

**3) Canonical Serialization Rules**

*(apply to the NM, and to any canonicalized Results/RunRecords used for hashing)*

1. **Encoding:** UTF-8, **no BOM**, Unix line endings (\n).
2. **Whitespace:** Minimal JSON; single space after : and , where applicable; no trailing spaces; exactly one final newline.
3. **Object key order:** All JSON object keys sorted **lexicographically (byte order)**.
4. **Array ordering (deterministic):**
   * **Variables**: sorted by **VM-VAR ID** ascending.
   * **Units**: sorted by **Unit ID** (lexicographic).
   * **Options**: sorted by **Option.order\_index**, then by **Option ID**.
   * Any list without intrinsic order **must be sorted** deterministically before hashing.
5. **Timestamps:** UTC ISO-8601 YYYY-MM-DDThh:mm:ssZ.
6. **Numbers:** Counts as integers; internal comparisons use integer/rational math. No presentation rounding in canonical bytes.
7. **Stable artifacts:** RES:… and RUN:… identifiers derive from **canonical bytes** (inputs + engine + FID), not from filesystem paths or environment.

**4) What goes into Result/RunRecord hashing inputs**

When producing canonical bytes for **Result** and **RunRecord** (and any bundle that is hashed or signed), **include**:

* The **ParameterSet snapshot** (all **VM-VAR** values actually used in the run).
* The **Formula ID (FID)** and **EngineVersion**.
* The **Dataset IDs** used: REG:… (DivisionRegistry), TLY:… (BallotTally), PS:… (ParameterSet), and any ADJMAP:…, AP:… if referenced.
* **Frontier controls** when applicable: frontier\_mode (040), frontier\_bands (042), and contiguity/island settings (047/048).
* **Ties** when randomized: tie\_policy (032) and tie\_seed (033).

**Explicit exclusions (never hashed into FID-scoped bytes):**

* Presentation/reporting toggles (e.g., label policies/thresholds, language).
* Non-normative metadata (author, free-text notes, UI preferences).
* Any display rounding—presentation happens later in the report layer.

**5) Compatibility & Reserved IDs**

* **Reserved (legacy, do not reuse):**
  + VM-VAR-050 … 052 (old tie variables; replaced by **032–033**).
  + VM-VAR-060 … 062 (old label placeholders; superseded by **044–046** in **non-FID** space).
* **Frontier & Ties naming:** Ensure **040/042** (frontier) and **032/033** (ties) are used consistently across docs, code, and fixtures.
* **Threshold domains & levels:** Keep 022/023 at **50..75** and 031 = country (v1)\*\* consistent everywhere.

**Annex A — Variable Canonical Reference Table (Updated)**

**Part 3/3: Validation & Conformance**

**6) Conformance requirements (must)**

An implementation **conforms** if all items below are true:

1. **Variables & domains**

* Supports exactly the variable set and IDs listed in Parts **1–2**.
* Enforces domains/defaults (e.g., 022/023 ∈ 50..75, 031 = country in v1).
* Treats presentation toggles as **non-FID** (ignored for FID).

1. **Fixed constants**

* Uses the fixed approval denominator (**approvals\_for\_change / valid\_ballots**).
* Applies *round half to even* only at the specified comparison points.
* Uses reduce\_continuing\_denominator for IRV.
* Honors the fixed allocation family set and MMP sequence.

1. **Ordering & determinism**

* Sorts **Units by Unit ID**; **Options by (order\_index, id)** everywhere.
* Produces identical canonical bytes (NM, Result, RunRecord) across OS/arch for the same inputs.

1. **RNG for ties**

* Only used when tie\_policy = random.
* Seeded **solely** by VM-VAR-033 tie\_seed; no OS entropy/time; no parallel RNG.
* Tie decisions and the seed are recorded in TieLog/RunRecord.

1. **Canonical serialization**

* UTF-8, no BOM, LF line endings.
* JSON object keys in lexicographic order; arrays sorted as specified.
* One trailing newline; no trailing spaces.

1. **Hashing scope**

* FID is computed **only** from the NM fields 1–4 (Part 2).
* Result/RunRecord hashes include ParameterSet snapshot, FID, EngineVersion, dataset IDs, and frontier/ties when applicable; exclude presentation toggles.

**7) Required validations (engine must reject on failure)**

* **Variable validation:** unknown IDs; values out of domain; missing required defaults.
* **Frontier bands:** unordered/overlapping bands; missing AP id when required for ladder.
* **Double-majority without frontier:** mode ∉ {by\_list, by\_tag} **or** empty affected\_region\_family\_ref.
* **Adjacency integrity:** edges referencing unknown Units; illegal edge types.
* **Tie config:** tie\_policy = random with missing/invalid tie\_seed.
* **Serialization guard:** attempts to hash bytes that violate canonical JSON rules (ordering/whitespace/EOL).

Each rejection must surface a **typed error** and an **operator-readable reason**.

**8) Reference conformance tests (minimum)**

A. **FID stability (NM)**

* Build NM from two different hosts; FID digests **match** byte-for-byte.

B. **Artifact stability**

* Given identical inputs and seed, serialized **Result** and **RunRecord** bytes are identical across:
  + Windows/macOS/Linux
  + x86-64/arm64

C. **Ordering checks**

* Shuffle input Option order; outputs remain identical (thanks to order\_index).

D. **Approval denominator invariance**

* Toggle report precision/formatting; FID and artifact hashes **unchanged**.

E. **Tie reproducibility**

* With tie\_policy = random, fixed tie\_seed: same winners and TieLog on all hosts.
* With deterministic: same winners via (order\_index, id); no RNG used.

F. **Frontier mapping**

* Single-band (“binary”) and multi-band cases produce exactly one status per Unit; contiguity and island rules respected; canonical bytes stable.

G. **Gate scoping**

* quorum\_per\_unit\_scope = frontier\_only vs frontier\_and\_family yields expected inclusion/exclusion behavior and hashes.

**9) Parallelism & reduction rules**

* Per-Unit **TABULATE/ALLOCATE** may run in parallel.
* **All reductions/aggregations** must iterate in the **stable orders** defined above.
* **No parallel RNG**; tie contexts are resolved serially in stable order.

**10) Hashing workflow (reference)**

1. Build **NM** (Parts 1–2), canonicalize JSON, compute **FID**.
2. Run engine → produce **Result** and **RunRecord** objects.
3. Canonicalize **Result** (sorted keys, LF) and compute RES:… hash/ID.
4. Canonicalize **RunRecord** and compute RUN:… hash/ID.
5. Verify that re-serialization immediately reproduces identical bytes and IDs.

**11) Provenance & audit fields (minimum)**

* **RunRecord** must include: FID, EngineVersion, dataset IDs (REG/TLY/PS/…),  
  ParameterSet snapshot, timestamps (UTC), and — if randomized — tie\_policy & tie\_seed.
* **Result** must include: label & reasons, gate outcomes, allocation/aggregates,  
  optional FrontierMap summary, and embedded TieLog when applicable.

**12) Compatibility & reserved IDs**

* Mark 050–052 (old ties) and 060–062 (old labels) **reserved/legacy**; do not reuse.
* Keep 022/023 (50..75) and 031 = country (v1) aligned across all docs/code/fixtures.
* Use 032/033 for ties and 040/042 for frontier consistently.

**13) Failure policy (artifact production)**

* **VALIDATE fail:** build **Invalid** Result + RunRecord (with reasons); **do not** emit FrontierMap; still canonicalize/hash.
* **Gates fail:** mark **Invalid**; skip frontier; log blocking ties only if they must be recorded.

**14) Implementation notes (non-normative but recommended)**

* Add a CI job that diff-checks canonical bytes across OS/arch.
* Enforce LF and key sorting with a pre-commit or serialization layer tests.
* Keep a tiny corpus of cross-platform “golden” artifacts to catch drift.