# **Pre-Coding Essentials (Component: crates/vm\_io/Cargo.toml, Version/FormulaID: VM-ENGINE v0) — 29/89**

## **1) Goal & Success**

Goal: Manifest for **vm\_io** (I/O & canonical JSON + schema validation + hashing + loaders).

Success: Builds as rlib; depends on vm\_core; JSON Schema validation and SHA-256 hashing available; **no UI/CLI deps**; all deps declared with default-features = false where sensible.

## **2) Scope**

In scope: package metadata, features, deps for JSON parse/validate (serde, serde\_json, jsonschema), hashing (sha2/digest), path handling, error derives.

Out of scope: algorithms/pipeline/report (other crates), RNG, UI.

## **3) Inputs → Outputs**

Inputs: Workspace toolchain; schemas & fixtures at runtime.

Outputs: Library used by vm\_pipeline/vm\_cli to read/validate/canonicalize/hashes.

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

## **5) Variables (build/features)**

## **6) Functions**

(Manifest only.)

## **7) Algorithm Outline (manifest structure)**

[package] name vm\_io, version 0.1.0, edition 2021, license = "Apache-2.0 OR MIT".

[lib] name="vm\_io", path="src/lib.rs", crate-type=["rlib"].

[features]

default = ["std","serde","schemaval","hash","path\_utf8"]

std = []

serde = ["dep:serde", "dep:serde\_json"]

schemaval = ["dep:jsonschema"]

hash = ["dep:sha2", "dep:digest"]

path\_utf8 = ["dep:camino"]

[dependencies] (pin major versions; disable unnecessary defaults)

vm\_core = { path = "../vm\_core" }

serde = { version = "1", features = ["derive"], optional = true, default-features = false }

serde\_json = { version = "1", optional = true } *(std only; used for parse/write)*

jsonschema = { version = "0.17", optional = true, default-features = false, features = ["draft2020-12"] }

sha2 = { version = "0.10", optional = true, default-features = false }

digest = { version = "0.10", optional = true, default-features = false }

hex = { version = "0.4", default-features = false }

camino = { version = "1", optional = true, default-features = false }

thiserror = { version = "1", default-features = false } *(error enums for loader/validator)*

[dev-dependencies]

assert\_json\_diff = "2" *(optional, for tests)*

tempfile = "3"

No build.rs.

## **8) State Flow**

vm\_pipeline/vm\_cli link vm\_io to load & validate: manifest → schemas → inputs; vm\_io also exposes canonical JSON writer and hashing.

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

Use **BTreeMap** or explicit canonical writer for sorted keys; LF line endings ensured in writer code (not by crate config).

Hashing via sha2 on **canonical bytes** only. No float parsing beyond JSON numbers (counts are integers by schema).

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

Do **not** pull network features; all crates compiled offline.

Keep serde\_json features minimal (no arbitrary precision toggles needed).

Ensure jsonschema feature gate allows building without validator (for tiny builds).

Avoid platform-specific deps; path handling via camino only when enabled.

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

cargo check -p vm\_io with defaults.

cargo check -p vm\_io --no-default-features --features "std,serde,hash" (validator off) builds.

cargo check -p vm\_io --no-default-features **fails intentionally** (I/O crate requires std)—documented.

Link test: cargo test -p vm\_pipeline compiles with vm\_io providing loaders/validator/hash.

No unwanted transitive default features (cargo tree -e features clean).