# **Pre-Coding Essentials (Component: crates/vm\_core/Cargo.toml, Version/FormulaID: VM-ENGINE v0) — 21/89**

## **1) Goal & Success**

Goal: Define the **core library crate** manifest for IDs, entities, variables, rounding, and RNG—no I/O.

Success: Builds as an rlib on all targets; optional serde feature compiles; no accidental JSON/IO deps; other crates (vm\_io, vm\_algo, vm\_pipeline) link cleanly.

## **2) Scope**

In scope: package metadata, edition/rust-version, [lib], features, minimal deps, crate-level lints.

Out of scope: binaries, CLI flags, JSON/FS handling (lives in vm\_io), web/UI deps.

## **3) Inputs → Outputs**

Inputs: Workspace toolchain, root Cargo profiles.

Outputs: vm\_core rlib exposing types/traits used across the engine.

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

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

## **6) Functions**

(Manifest only; no code signatures here.)

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

[package] — name vm\_core, version 0.1.0, edition 2021, rust-version = pinned toolchain major/minor; license = "Apache-2.0 OR MIT".

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

[features]

default = ["std"]

std = []

serde = ["dep:serde"]

[dependencies]

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

rand\_chacha = { version = "0.3", default-features = false }

rand\_core = { version = "0.6", default-features = false }

*(No serde\_json, no anyhow, no thiserror here; keep core lean.)*

[dev-dependencies] (minimal; only what unit tests in vm\_core require).

(Optional) [lints] or #![deny(...)] configured in code; keep Cargo clean.

No build.rs.

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

Other crates depend on vm\_core; vm\_io enables serde when it needs serialization; pipeline/algo link the RNG and rounding helpers from here.

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

Determinism aided by pinning RNG implementation (rand\_chacha) and exposing a seedable API from vm\_core::rng.

No float-based deps here; numeric/rounding code is in this crate’s source, not in dependencies.

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

If serde is disabled, vm\_core must still compile (types with #[cfg(feature="serde")] derives only).

Do not introduce std::fs/serde\_json here—keeps layering clean (vm\_io handles I/O).

Any added dependency must be default-features = false to avoid pulling in unexpected platform features.

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

cargo check -p vm\_core (default features) OK.

cargo check -p vm\_core --no-default-features --features serde OK.

cargo check -p vm\_core --no-default-features OK (compiles without std if/when code supports it, otherwise keep std required for now).

Downstream compile: cargo test -p vm\_io with features = ["serde"] succeeds, proving feature wiring.