# **Pre-Coding Essentials (Component: crates/vm\_cli/src/args.rs, Version/FormulaID: VM-ENGINE v0) — 67/89**

## **1) Goal & Success**

Goal: Define a deterministic **CLI argument surface** that maps cleanly to the fixed pipeline and offline policy; validate inputs early (including manifest rules) before running.

Success: Args enforce **exact one** of ballots | ballot\_tally, **exact one** of {manifest} | {explicit files}, accept only local files, and (if provided) a valid 32-byte hex RNG seed; parsing is OS-agnostic and side-effect free.

## **2) Scope**

In scope: clap/structopt style parsing; cross-field validation (mutual exclusivity, required-together); normalization of paths; basic content checks for seed format and manifest invariants (without I/O beyond existence).

Out of scope: Running the pipeline, reading files, hashing, or network (forbidden at runtime).

## **3) Inputs → Outputs (with schemas/IDs)**

Inputs (user flags):

**Manifest mode:** --manifest <path> (points to run manifest JSON). Enforce manifest rules: one REG, one PS, and **exactly one** of ballots or ballot\_tally; canonicalization tag must match constant; each sha256 is 64-hex.

**Explicit mode:** --registry <path> --params <path> **and** exactly one of --ballots <path> | --tally <path>.

Optional inputs: --adjacency <path> (Frontier), --autonomy <path> (optional package). Shapes are defined in Annex B Part 0.

Output controls: --out <dir>, --render json|html (reporting reads only Result/RunRecord/FrontierMap).

Determinism: --seed <64-hex> (optional). If provided, must decode to **32 bytes**; else leave to ParameterSet/manifest.

Output: Args struct consumed by main.rs; contains normalized paths, selected renderers, and validated switches mapping to **LOAD→…→BUILD\_RUN\_RECORD** stages.

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

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

## **6) Functions (signatures only)**

rust

CopyEdit

#[derive(Parser)]

pub struct Args {

// Mode selection

#[arg(long, conflicts\_with\_all=["registry","ballots","tally","params"])]

pub manifest: Option<PathBuf>,

// Explicit mode

#[arg(long)] pub registry: Option<PathBuf>,

#[arg(long)] pub params: Option<PathBuf>,

#[arg(long, conflicts\_with="tally")] pub ballots: Option<PathBuf>,

#[arg(long, conflicts\_with="ballots")] pub tally: Option<PathBuf>,

// Optional inputs

#[arg(long)] pub adjacency: Option<PathBuf>,

#[arg(long)] pub autonomy: Option<PathBuf>,

// Output & rendering

#[arg(long, default\_value = ".")] pub out: PathBuf,

#[arg(long, value\_parser=["json","html"], num\_args=0..=2)]

pub render: Vec<String>,

// Determinism

#[arg(long)] pub seed: Option<String>, // 64 lowercase hex

#[arg(long)] pub validate\_only: bool, // parse/validate, no run

#[arg(long)] pub quiet: bool,

}

pub fn parse\_and\_validate() -> Result<Args, CliError>;

fn validate\_manifest\_mode(a:&Args) -> Result<(),CliError>;

fn validate\_explicit\_mode(a:&Args) -> Result<(),CliError>;

fn validate\_seed\_format(hex:&str) -> Result<(),CliError>; // 64 hex → 32 bytes

(Conflicts/requirements enforce the manifest rules and ballots vs tally choice.)

## **7) Algorithm Outline (bullet steps)**

Parse with clap.

If --manifest: ensure no explicit inputs present; for fast fail, check JSON **exists**; defer schema validation to loader, but precheck: canonicalization tag present string, one REG, one PS, and **exactly one** of ballots|tally in the manifest.

If explicit mode: require --registry and --params and exactly one of --ballots | --tally.

If --seed present: must be **64 lowercase hex** decoding to **32 bytes**; else error.

Normalize paths; return Args. Main will drive the fixed pipeline order.

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

args.rs → main.rs orchestrates **LOAD → … → BUILD\_RUN\_RECORD**. No network; all inputs are local files.

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

No time/RNG used here; seed is **input**, not generated.

Enforce canonicalization expectations early (lowercase hex, presence of canonicalization tag); downstream serialization uses UTF-8, LF, sorted keys, UTC.

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

Both --ballots **and** --tally (or neither) ⇒ error.

Missing --registry or --params in explicit mode ⇒ error.

Bad --seed (odd length, non-hex, not 32B) ⇒ error.

Manifest with wrong canonicalization tag or malformed sha256 ⇒ error.

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

vm --help prints flags; parsing works on all OS targets.

**Manifest mode**: minimal Annex B Part 0 manifest passes; ballots↔tally swap passes; both present fails.

**Explicit mode**: require --registry --params + exactly one of ballots|tally.

Seed validation rejects non-64-hex / non-32B; accepts valid.