AXILOG

DVEC-001

# Deterministic Verifiable Execution Contract

Production Gold | v1.3 | Foundation

| | |
|---|---|
| Last updated | 2 April 2026 |
| Supersedes | DVEC-001 v1.2 |
| Depends on | DVM-SPEC-001 |
| Compliance | DO-178C · IEC 62304 · ISO 26262 |
| Changelog | 1 entry |

## Contents

**Scope:** Axioma Framework · Axilog Substrate · certifiable-* Ecosystem **Alignment:** DVM-SPEC-001 v1.0-rc1 · Axioma v0.4 · MDCP · Patent GB2521625.0

---

# Revision History

| Version | Status | Notes |
|---|---|---|
| v1.0 | Superseded | Initial draft |
| v1.1 | Superseded | Fault propagation, memory totality, SRS traceability, CI gates, pre-commit guardrail, closure artifacts |
| v1.2 | Superseded | Domain separation registry clarified — evidence type tags and chain tags formally separated |
| v1.3 | Final | Terminology standardised to “non-conformant system”; arithmetic guardrail AST note; §2.5 terminal reset clarified |

---

# 0. Governing Principle

> **If behaviour cannot be proven, it is non-conformant.**

## 0.1 Contract Versioning

This contract is versioned. All code MUST declare its contract version in the module header:

```
// DVEC: v1.3
```

A contract upgrade requires:

- Explicit migration assessment
- Re-verification of affected modules
- Changelog entry in axiom registry
- No silent adoption of new constraints

---

# 1. Determinism Mandate (Class M)

## 1.1 Bit-Identity Requirement

```
F(S, I) → O

∀ platforms P₁, P₂:
  F_P1(S, I) = F_P2(S, I)
```

Violation = catastrophic integrity failure.

## 1.2 Forbidden Constructs

```
❌ float, double
❌ undefined behaviour (C UB of any kind)
❌ non-deterministic iteration (unordered collections)
❌ system clock access in domain logic
❌ hidden global state
❌ uncontrolled randomness
```

## 1.3 Mandatory Constructs

```
✅ fixed-point arithmetic (Q16.16 minimum)
✅ explicit ordering in all collections
✅ canonical encoding before hashing
✅ deterministic reduction topology
```

## 1.4 Determinism Classification (Required)

Every module MUST declare its determinism class at the top of its primary header:

```
// AXILOG DETERMINISM: D1 — Strict Deterministic
// AXILOG DETERMINISM: D2 — Constrained Deterministic (deps declared below)
// AXILOG DETERMINISM: D3 — Observed Non-Deterministic (oracle boundary)
```

- D1 → strict deterministic — bit-identical across all platforms
- D2 → constrained deterministic — deterministic given declared dependency set (§6)
- D3 → observed non-deterministic — routed through Oracle Boundary Contract (§3)

---

# 2. Totality Contract (Class M)

## 2.1 Total Function Requirement

```
∀ input ∈ Domain:
  ∃ output ∈ Codomain
```

No partial functions. No undefined branches. No implicit fallthrough.

## 2.2 Boundedness

All computation MUST be statically or provably bounded.

```
❌ unbounded recursion
❌ uncontrolled loops
❌ dynamic growth without constraint
```

## 2.3 State Machine Closure

- All transitions declared exhaustively
- All transitions enforced mechanically
- No undeclared state reachable

## 2.4 Fault Propagation Contract (Critical)

All functions MUST accept and propagate fault context:

```
int32_t dvm_add_q16(int32_t a, int32_t b, ct_fault_flags_t *faults);
```

**Required:**

```
✅ fault context mutation at every arithmetic operation
✅ explicit propagation at every call site
✅ fault checked at execution boundary
```

**Forbidden:**

```
❌ errno-based error handling
❌ silent failure (returning 0 without fault flag)
❌ unchecked return values
❌ ignorable error states
```

## 2.5 Fail-Closed Terminality (Closure Invariant)

Any fault flag set MUST:

- Trigger transition to `FAILED` state
- Halt further state mutation
- Require explicit system reset to resume

No fault may be cleared implicitly. No fault may be silently ignored.

> **Reset scope note:** At the SRS layer, the definition of "explicit system reset" MUST be declared per deployment context — whether process restart, subsystem reinitialisation, or full system-level reset. The DVEC mandates that reset is explicit and declared; the specific reset boundary is a deployment-level SRS requirement. Undefined reset scope constitutes a non-conformant system.

---

# 3. Oracle Boundary Contract (Class M)

## 3.1 Entropy Admission Rule

No external input may influence system state unless it has been:

1. Captured
2. Canonicalised
3. Committed (AX:OBS:v1)
4. Referenced

## 3.2 Oracle Types

All of the following are oracle interactions governed by this section:

- LLM inference
- Time injection
- Vector retrieval
- Sensors / physical inputs
- External APIs

## 3.3 Time Rule

```
❌ System time forbidden in domain logic
✅ Time injected as immutable canonical input
✅ Committed as AX:OBS:v1 (oracle.type: TIME_INJECTION)
```

Within domain execution, time is immutable state data. There is no concept of “current time” inside domain execution.

## 3.4 Time Monotonicity (Closure Invariant)

Time inputs MUST be strictly non-decreasing.

- Any rollback = protocol violation = non-conformant system
- Non-monotonic time permitted only if declared at system initialisation and committed to the ledger
- A system that accepts a timestamp ≤ the prior committed timestamp without declaration constitutes a non-conformant system

# 4. Evidence Mandate (Class M)

## 4.1 Evidence Completeness

```
cause → evidence → state transition
```

No state transition may occur without a prior committed cause. Retroactive evidence generation is non-conformant.

## 4.2 Canonicalisation

All evidence payloads MUST:

- Use RFC 8785 (JCS) canonicalisation
- Produce byte-identical output for identical inputs
- Be canonicalised BEFORE hashing

```
❌ hashing raw structs
❌ non-canonical JSON serialisation
❌ serialiser-dependent output
```

## 4.3 Commitment Function

```
commit(e) = SHA-256(tag || LE64(|payload|) || payload)
```

No variation permitted. Domain separation via tag is mandatory.

## 4.4 Domain Separation Registry (Closure Invariant — v1.2)

*Clarified in v1.2: evidence type tags and chain tags are formally separated namespaces. No cross-namespace reuse permitted.*

**Evidence type tags** — typed evidence records committed to the ledger:

| Tag | Type | Description |
|---|---|---|
| `AX:STATE:v1` | State Commitment | Canonical DVM state hash |
| `AX:TRANS:v1` | Transition Commitment | State change witness |
| `AX:OBS:v1` | Observation Record | Oracle interaction record |
| `AX:POLICY:v1` | Policy Assertion | Governance rule evaluation |
| `AX:PROOF:v1` | Verification Proof | Replay or conformance proof |

**Chain tags** — ledger protocol prefixes, not evidence types:

| Tag | Protocol | Description |
|---|---|---|
| `AX:LEDGER:v1` | Axioma hash chain | Per §5.2 ledger chaining |
| `DVM:LEDGER:v1` | DVM computation chain | Per DVM-SPEC-001 §7 |
| `DVM:STATE:v1` | DVM state commitment | Per DVM-SPEC-001 §7.1 |
| `DVM:INGRESS:v1` | DVM ingress oracle | Per DVM-SPEC-001 §7.1 |

**Rules:**

- Evidence type tags MAY appear as payloads in the ledger chain
- Chain tags MAY NOT be used as evidence type identifiers
- No tag from either namespace may be reused in the other
- No ad hoc tags permitted outside this registry

## 4.5 Evidence Type Versioning

The evidence type set is closed within a major version.

New evidence types require:

- A new major version tag (e.g. `AX:OBS:v2`)
- A formal migration specification
- Backward compatibility proof for existing chains
- Registry update with explicit changelog entry

---

# 5. Ledger Contract (Class M)

## 5.1 Total Order Guarantee

```
∀ events A, B:
  A < B  XOR  B < A
```

No two events share the same chain position. No concurrency ambiguity permitted. Enforced via per-organisation chain head lock (SELECT FOR UPDATE).

## 5.2 Hash Chain Integrity

$$L_0 = \text{SHA-256}(\text{"AX:LEDGER:v1"} \parallel \text{commit}(e_0))$$
$$L_n = \text{SHA-256}(\text{"AX:LEDGER:v1"} \parallel L_{n-1} \parallel \text{commit}(e_n))$$

Any break invalidates the entire downstream chain.

## 5.3 Immutability

```
✅ Append-only
❌ No UPDATE on committed records
❌ No DELETE on committed records
```

# 6. Dependency Closure (Class M — D2 Components)

All D2 components MUST declare their complete dependency set in the module header:

| Dependency | Requirement |
|---|---|
| Model version | Fully qualified identifier + version hash |
| Parameters | All runtime parameters — no defaults |
| Tool versions | External tool version identifiers |
| Configuration | All parameters influencing behaviour |
| Data snapshot | Index state hash, corpus hash |

A missing dependency constitutes a non-conformant system — not a runtime surprise.

---

# 7. Cross-Platform Identity (Class M)

## 7.1 DVM Contract

All D1 execution MUST be:

- Integer-only arithmetic
- Saturating on overflow
- Defined overflow behaviour
- Identical reduction topology

## 7.2 Widen-Then-Operate (Closure Invariant)

All fixed-point arithmetic MUST follow:

```
1. Widen operands (e.g. int32_t → int64_t)
2. Operate
3. Saturate to target width
4. Record fault flag if saturated
```

No raw operator arithmetic on fixed-point types.

## 7.3 Golden Reference Requirement

All D1 systems MUST:

- Produce a certifiable-harness compatible 368-byte golden reference
- Match bit-identically across x86_64, ARM64, and (future) RISC-V
- Fail CI if golden reference diverges between architectures

Failure = non-conformant system.

---

# 8. No Boilerplate Rule

## 8.1 The Three-Part Test

Every line of code must satisfy at least one of:

1. Enforces an invariant
2. Implements a contract
3. Contributes to a proof

If none apply — delete it.

## 8.2 Prohibition

```
❌ wrapper abstractions without invariants
❌ generic helpers without proof value
❌ duplicated patterns without formal reason
❌ defensive code masking invariant failure
```

## 8.3 Deferred Correctness Prohibition (Critical)

Forbidden in all production code:

```
❌ TODO
❌ FIXME
❌ OPTIMIZE
❌ HACK
❌ commented-out code blocks
❌ placeholder returns (0, NULL without fault)
❌ incomplete switch arms with silent fallthrough
❌ stub implementations
```

> **A TODO is an undeclared invariant failure.**

If a proof obligation cannot be satisfied now, it MUST be registered as an explicit open axiom in the registry — not hidden in a comment.

---

# 9. Proof & Traceability

## 9.1 CI Gate (Mandatory)

CI MUST execute and pass ALL of the following:

```
✅ verify-axiom-coverage (all Class M axioms dual-layer anchored)
✅ cross-platform bit-identity test (D1 components)
✅ hash stability test (commitment reproducibility)
✅ property test suite (all registered tests)
✅ static analysis — zero warnings (clang-tidy, cppcheck)
✅ forbidden pattern scan (§16)
✅ TODO/FIXME/HACK scan
✅ missing SRS anchor detection
✅ raw arithmetic on fixed-point types detection
```

A merge that fails any gate constitutes a non-conformant system regardless of review approval.

## 9.2 SRS Traceability (Critical)

Every public function MUST include a requirements anchor:

```
// SRS-042-SHALL: Deterministic addition must saturate on overflow
int32_t dvm_add_q16(int32_t a, int32_t b, ct_fault_flags_t *faults);
```

- 1:1 mapping: function ↔ requirement
- No orphan code permitted
- No orphan requirements permitted

---

# 10. Testing = Proof

## 10.1 Required Test Classes

```
✅ Property-based tests (all inputs, not happy path)
✅ Cross-platform identity tests
✅ Replay equivalence tests
✅ Hash stability tests
✅ Fault injection tests (all fault paths exercised)
```

## 10.2 Forbidden Test Patterns

```
❌ Example-based happy path tests only
❌ Non-deterministic mocks (random output, time-dependent)
❌ Mocks bypassing oracle contract
❌ Unverified snapshot tests without canonicalisation
❌ Tests that pass on one platform but are not verified on others
```

---

# 11. Performance Contract

Performance optimisations MUST NOT:

- Break determinism
- Introduce non-deterministic ordering
- Alter reduction topology
- Change observable fault behaviour

**Correctness > Performance. Always.**

A performance optimisation that alters any of the above is a correctness regression, not an improvement.

---

# 12. Code Standards (C99)

## 12.1 Memory Totality (Critical)

```
❌ malloc / free / realloc in runtime paths
✅ Static allocation at initialisation
✅ Caller-provided buffers
```

All buffer ownership MUST be explicit:

```
/* Correct pattern — caller owns all memory */
void dvm_matmul(
    int32_t       *out,       /* caller-allocated output */
    const int32_t *a,         /* caller-owned input */
    const int32_t *b,         /* caller-owned input */
    size_t         n,
    ct_fault_flags_t *faults
);
```

No hidden allocation. No implicit ownership transfer.

## 12.2 Arithmetic Enforcement (Critical)

```
❌ Raw operators on fixed-point types (+, -, *, /)
✅ dvm_add_q16()
✅ dvm_mul_q16()
✅ dvm_div_q16()
✅ Saturating operations with fault propagation
```

## 12.3 Compiler Contract

```
✅ Strict C99 only
✅ -Wall -Wextra -Werror
✅ No implicit casts
✅ Explicit integer sizes (uint32_t, int64_t, etc.)
✅ Must compile identically: GCC, Clang, MSVC
❌ No compiler-specific extensions
❌ No UB (validated by static analysis)
❌ No uninitialised memory
❌ No pointer aliasing ambiguity
```

## 13. Concurrency Contract

Where parallel execution is used:

```
✅ Fixed tree reduction topology
✅ Identical reduction order across runs
✅ Deterministic scheduling
❌ Race conditions
❌ Unordered reduction
❌ Shared mutable state without explicit control
```

A parallel implementation is conformant if and only if it produces bit-identical canonical state to sequential execution.

## 14. Verification Checklist (Mandatory Before Merge)

```
Determinism
  □ Determinism class declared in module header
  □ No forbidden constructs present
  □ Bit-identical across platforms (D1)

Totality
  □ All inputs mapped to outputs
  □ No undefined states reachable
  □ All fault paths exercised

Evidence
  □ All state transitions have prior evidence
  □ RFC 8785 canonicalisation applied
  □ Correct tag from domain separation registry (§4.4)

Ledger
  □ Total order preserved
  □ No UPDATE or DELETE

Oracle
  □ All external inputs committed before use
  □ Time injected via Time Oracle

Memory
  □ No heap allocation in runtime paths
  □ All buffer ownership declared

Proof
  □ SRS anchor present on every public function
  □ No TODO / FIXME / HACK
  □ Golden reference match (D1 systems)

CI
  □ All CI gates will pass
```

## 15. Code Review Contract

**A review is verification, not approval.**

A reviewer MUST explicitly confirm:

- Determinism class declared and correct
- No forbidden constructs present
- Fault propagation complete at every call site
- SRS traceability complete
- Memory ownership declared
- Domain separation tags correct (§4.4)
- CI gates will pass

Approval without verification constitutes a non-conformant system. A review that does not confirm the above is invalid and non-conformant.

---

## 16. Pre-Commit Guardrail

Automatic rejection if any of the following are detected:

**Memory violations:**

```
malloc  free  realloc
```

**Floating-point violations:**

```
float  double
```

**Deferred correctness:**

```
TODO  FIXME  HACK  ???  OPTIMIZE
```

**Time violations:**

```
time(  clock(  gettimeofday(  clock_gettime(
```

**Traceability violations:**

```
public function without SRS anchor
```

**Arithmetic violations:**

```
raw +  -  *  /  on fixed-point types without dvm_* wrapper
```

> **Implementation note:** Raw arithmetic detection MUST use AST-level analysis (clang-tidy plugin or equivalent) — not grep. Token-level scanning cannot distinguish operator context. A grep-based scan is insufficient and constitutes a non-conformant system.

**Tag violations:**

```
evidence tag not in §4.4 registry
chain tag used as evidence type
```

---

## 17. Closure Artifacts (Required)

The system MUST include and maintain:

| Artifact | Purpose |
|---|---|
| Mechanised RTM | SRS ↔ code 1:1 traceability map |
| Golden verifier | Cross-platform bit-identity validation |
| JCS linter | RFC 8785 canonicalisation verification |
| Static WCET gate | Worst-case execution time bound proof |
| Domain tag linter | §4.4 registry conformance check |

---

## 18. Closing Principle

> A system is not correct because it passes tests. It is correct because it cannot be incorrect without detection.

## Final Statement

> **This contract enforces not just correctness — but the inevitability of correctness.**

## Ecosystem Alignment

| Layer | Document | Status |
|---|---|---|
| Computation | DVM-SPEC-001 v1.0-rc1 | Complete |
| Framework | Axioma v0.4 | Complete |
| Development | DVEC-001 v1.3 | Final |
| Substrate | axilog.io | Registered |
| Proven patterns | SpeyBooks Kernel (M1–M7) | Production Gold |
| Existing assets | certifiable-* (9 repos) | Production Gold |

*DVEC-001 v1.3 — Final William Murray · SpeyTech · March 2026 Patent GB2521625.0*