AXILOG

SRS-003

Policy Evaluation & Operational Envelope Specification

Locked v0.3 L4 [D2]

Last updated 26 March 2026
Depends on SRS-001 · SRS-002 · DVEC-001
Changelog 1 entry

Contents

Depends on: SRS-001 v0.3 · SRS-002 v0.3 · DVEC-001 v1.3

---

1. PURPOSE

This document defines the **Policy Evaluation Contract** for the Axioma framework.

It specifies:

- deterministic policy evaluation
- operational safety envelope enforcement
- policy evidence generation
- integration with L5 agent behaviour

Objective: Every policy decision SHALL be deterministic, replayable, and anchored to cryptographic evidence.

Transition: L5 defines "how to behave" → L4 defines "what is allowed".

---

2. DEFINITIONS

2.1 Policy

A **policy** is a deterministic constraint evaluated against admitted system inputs.

2.2 Policy Evaluation

A **policy evaluation** is a pure function:

```
P : Input → (Result × Evidence)
```

2.3 AX:POLICY:v1

A canonical record representing a single policy evaluation.

2.4 Certified Operational Envelope (COE)

The **COE** is the set of all states permitted by all active policies.

2.5 Policy Set

The **policy set** is the finite, closed set of all policies evaluated by the system.

2.6 Policy Result

Value	Meaning
`PERMITTED`	Input is within operational envelope
`BREACH`	Input violates policy constraint

---

3. DETERMINISM MODEL

3.1 Determinism Class

The policy layer SHALL operate under:

D2 — Constrained Deterministic

3.2 Determinism Definition

SRS-003-SHALL-001

For identical:

- admitted inputs (`AX:OBS:v1`)
- policy set

the system SHALL produce:

- identical `AX:POLICY:v1` records
- identical evaluation results

3.3 Oracle Boundary

SRS-003-SHALL-002

Policy evaluation MUST operate only on:

- admitted `AX:OBS:v1` inputs
- static configuration

Direct system access is **FORBIDDEN**.

---

4. POLICY INPUT MODEL

4.1 Input Binding

SRS-003-SHALL-011

All policy evaluations SHALL operate on data derived from admitted `AX:OBS:v1`.

Extraction MUST:

- be deterministic
- be schema-defined
- produce identical values across platforms

4.2 Input Integrity

Policy evaluation MUST NOT:

- mutate input payload
- access non-admitted state

---

5. POLICY SET CLOSURE

5.1 Closed Set Requirement

SRS-003-SHALL-013

The policy set SHALL be:

- finite

- explicitly defined at compile time
- identical across all builds

5.2 No Dynamic Policies

Runtime policy creation is **FORBIDDEN**.

5A. POLICY IDENTITY

5A.1 Policy Identity Stability

SRS-003-SHALL-019

`policy_id` SHALL:

- be compile-time constant
- be globally unique
- be identical across all builds
- use ASCII encoding only (no locale dependence)

---

6. POLICY EVALUATION ORDERING

6.1 Global Ordering

SRS-003-SHALL-012

Policy evaluations SHALL occur in deterministic order:

```
ORDER BY (ledger_seq ASC, policy_id ASC)
```

6.2 Ordering Source

- `ledger_seq` is the primary ordering axis
- `policy_id` resolves intra-input ordering

7. POLICY EVALUATION CONTRACT

7.1 Evaluation Function

SRS-003-SHALL-003

Each policy SHALL be a pure function:

```
P(input) → (result, evidence)
```

7.2 Fixed-Point Arithmetic

SRS-003-SHALL-002 (extended)

All numeric evaluation MUST use:

Q16.16 fixed-point arithmetic

Floating-point operations are **FORBIDDEN**.

Example:

```
if (velocity > MAX_SAFE_Q16) {
    /* BREACH */
}
```

A policy breach detected on an ARM64 edge device will be **exactly reproducible** on an x86_64 auditor workstation.

7.3 Policy Aggregation

SRS-003-SHALL-014

If **ANY** policy returns `BREACH`:

- overall result SHALL be `BREACH`

`PERMITTED` is valid only if **ALL** policies return `PERMITTED`.

7A. POLICY TOTALITY

7A.1 Totality Requirement

SRS-003-SHALL-020

Policy evaluation SHALL be total.

For any admitted input:

- evaluation MUST produce a result
- no undefined evaluation paths are permitted

If evaluation cannot be completed:

- result SHALL be `BREACH`
- agent SHALL transition to `STOPPED`

8. AX:POLICY:v1 RECORD

8.1 Canonical Requirement

SRS-003-SHALL-004

All records MUST be:

- RFC 8785 (JCS) canonicalized
- bit-identical for identical inputs

8.2 Required Fields

SRS-003-SHALL-005

Each record SHALL contain:

Field	Type	Description
`actual`	Q16.16	Observed value
`ledger_seq`	uint64	Input ordering source
`policy_id`	string	Policy identifier
`result`	enum	`PERMITTED` | `BREACH`
`threshold`	Q16.16	Policy constraint

8.3 Field Order

SRS-003-SHALL-018

Fields MUST appear in lexicographic order:

```
actual, ledger_seq, policy_id, result, threshold
```

8.4 Canonical Format

```
{"actual":<q16.16>,"ledger_seq":<uint64>,"policy_id":"<string>","result":"
<enum>","threshold":<q16.16>}
```

8.5 Forbidden Fields

The following SHALL NOT appear:

- timestamps (unless admitted)
- floating-point values
- random values
- memory addresses
- process/thread IDs

8A. EVIDENCE MULTIPLICITY

8A.1 One Record Per Policy

SRS-003-SHALL-021

Each policy SHALL produce exactly one `AX:POLICY:v1` record.

Aggregation SHALL NOT collapse multiple policy evaluations into a single record.

8A.2 Evidence Count

For N policies evaluated against one input:

- exactly N `AX:POLICY:v1` records SHALL be produced
- each record SHALL be independently committed

---

9. PRE-COMMIT INVARIANT

9.1 Policy Commitment Requirement

SRS-003-SHALL-015

Each policy evaluation SHALL:

```
evaluate → construct AX:POLICY:v1 → commit → influence L5
```

9.2 No Silent Decisions

No policy result may influence behaviour without a committed record.

9A. CROSS-LAYER ORDERING

9A.1 Cross-Layer Ordering Invariant

SRS-003-SHALL-022

For each input, the commit ordering SHALL be:

```
AX:OBS:v1
→ AX:POLICY:v1 (0..N)
→ AX:TRANS:v1 (1)
```

All `AX:POLICY:v1` records MUST be committed **BEFORE** the corresponding `AX:TRANS:v1` record.

9A.2 Ordering Invariance

This ordering SHALL be invariant across all platforms.

Interleaving of policy and transition commits is **FORBIDDEN**.

---

10. FAILURE SEMANTICS

10.1 Evaluation Failure

SRS-003-SHALL-016

If evaluation fails:

- result SHALL be treated as `BREACH`
- agent SHALL transition to `STOPPED`

10.2 Commit Failure

If `AX:POLICY:v1` cannot be committed:

- agent SHALL transition to `STOPPED`
- failure SHALL be treated as critical

---

11. PURITY CONSTRAINT

11.1 No Hidden State

SRS-003-SHALL-017

Policy evaluation SHALL be a pure function of:

- admitted inputs
- static configuration

11.2 Forbidden Dependencies

Policies MUST NOT depend on:

- global mutable state
- system clock
- randomness
- IO

---

12. BOUNDEDNESS

12.1 Execution Bound

SRS-003-SHALL-010

Policy evaluation SHALL execute in:

```
O(N), where N = number of policies
```

12.2 Constraint

N MUST be:

- fixed
- bounded
- compile-time constant

12.3 Memory

SRS-003-SHALL-009

Policy evaluation SHALL perform **zero dynamic memory allocation**.

---

13. L5 INTEGRATION CONTRACT

13.1 Enforcement Rule

SRS-003-SHALL-006

No L5 transition SHALL complete if:

```
policy_result == BREACH
```

13.2 Violation Propagation

SRS-003-SHALL-007

A `BREACH` SHALL:

- force L5 transition to `ALARM` or `STOPPED`
- be recorded in L5 transition evidence

---

14. REPLAY EQUIVALENCE

14.1 Replay Requirement

SRS-003-SHALL-003 (extended)

Given identical:

- inputs
- policy set

the system SHALL reproduce identical:

- `AX:POLICY:v1` records
- policy outcomes

---

15. TRACEABILITY

15.1 Requirement Mapping

SRS-003-SHALL-008

Each policy evaluation SHALL be traceable to:

- an SRS requirement
- an `AX:POLICY:v1` record

---

16. PHASE 3 CLOSURE CRITERIA

Phase 3 is complete when:

16.1 Determinism

- identical inputs → identical outputs

16.2 Ordering

- evaluation order deterministic

16.3 Closure

- policy set is finite and fixed

16.4 Evidence

- all decisions produce canonical records

16.5 Enforcement

- L5 transitions blocked on `BREACH`

16.6 Replay

- policy outcomes reproducible

---

17. REQUIREMENT SUMMARY

ID	Requirement	Section
SRS-003-SHALL-001	Determinism definition	3.2
SRS-003-SHALL-002	Oracle boundary / fixed-point	3.3, 7.2
SRS-003-SHALL-003	Deterministic evaluation	7.1, 14.1
SRS-003-SHALL-004	Canonical record	8.1
SRS-003-SHALL-005	Required fields	8.2
SRS-003-SHALL-006	L5 enforcement	13.1
SRS-003-SHALL-007	Violation propagation	13.2
SRS-003-SHALL-008	Traceability	15.1
SRS-003-SHALL-009	Zero allocation	12.3
SRS-003-SHALL-010	Bounded execution	12.1
SRS-003-SHALL-011	Input binding	4.1
SRS-003-SHALL-012	Ordering	6.1
SRS-003-SHALL-013	Policy set closure	5.1
SRS-003-SHALL-014	Aggregation rule	7.3
SRS-003-SHALL-015	Pre-commit invariant	9.1
SRS-003-SHALL-016	Failure semantics	10.1
SRS-003-SHALL-017	Purity constraint	11.1
SRS-003-SHALL-018	Canonical structure	8.3
SRS-003-SHALL-019	Policy identity stability	5A.1
SRS-003-SHALL-020	Policy totality	7A.1
SRS-003-SHALL-021	One record per policy	8A.1
SRS-003-SHALL-022	Cross-layer ordering	9A.1

Total: 22 SHALL requirements

18. FINAL STATEMENT

The Axioma policy layer SHALL:

> Define a deterministic, bounded, and fully observable operational envelope whose decisions are cryptographically anchored and replayable across platforms.

System Property:

> No behaviour can occur without both evidence (L6) and permission (L4).

Layer Integration:

Layer	Role
L6	Truth — what happened
L5	Behaviour — what to do
L4	Admissibility — what is allowed

19. REVISION HISTORY

Version	Date	Author	Changes
0.1-scaffold	2026-03-26	William Murray	Initial scaffold
0.2	2026-03-26	William Murray	Full specification, 18 SHALL requirements
0.3	2026-03-26	William Murray	Added SHALL-019 (policy identity), SHALL-020 (totality), SHALL-021 (evidence multiplicity), SHALL-022 (cross-layer ordering). Audit-frozen FINAL.

20. DOCUMENT APPROVAL

Role	Name	Date	Signature
Author	William Murray	2026-03-26	
Reviewer			
Approver			