ENGINEERING STANDARDS LIBRARY / DATA, STATE, AND EVIDENCE

Event Sourcing, CQRS, & Durable State Machine Standard

The architecture of state management has failed.

PERMANENT DOCUMENT ID

MYTHOS-DAT-0004

PUBLICATION

Candidate Standard
Version 1.0.0-candidate

ASSURANCE PROFILE

Standard / Advanced
High Assurance

PERMANENT PUBLICATION IDENTITY

Event Sourcing, CQRS, & Durable State Machine Standard

DOCUMENT ID
MYTHOS-DAT-0004

VERSION
1.0.0-candidate

STATUS
Candidate Standard

PUBLISHER
Mythos Systems

PUBLICATION DATE
2026-07-24

SCHEDULED REVIEW
2027-07-24

12	6	4	1
ATOMIC CRITERIA	ASSURANCE VIEWS	IMPLEMENTATION PROFILES	PERMANENT IDENTITY

Publication doctrine. This standard is a permanent library identity. Interpretation must preserve its recorded evidence, controlled exceptions, formal revision history, and explicit relationship to companion standards.

DOMAIN CONTROL SYSTEM

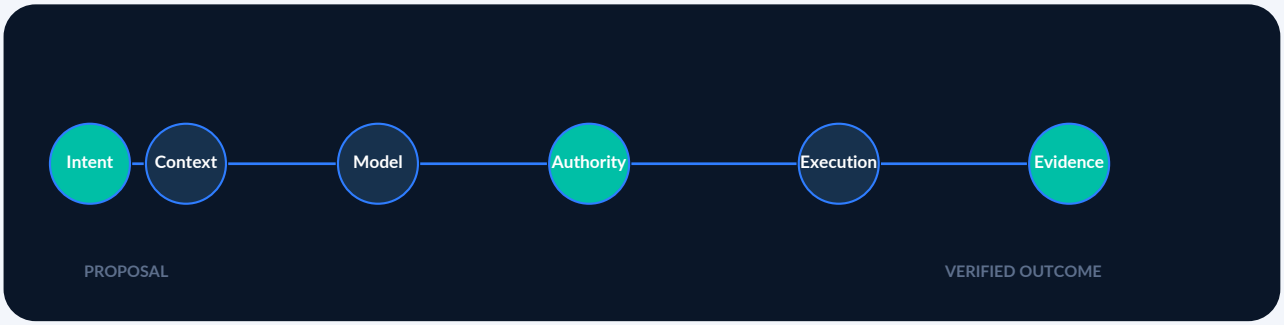
MYTHOS-DAT-0004 inside the data, state, and evidence constellation

Connected assurance	Architecture, implementation, operations, evidence, and lifecycle are evaluated as one controlled system.
Specialization rule	Companion standards may add precision, but cannot silently weaken this publication.

Executive orientation

Establishes event-sourcing, CQRS, durable state-machine, replay, idempotency, consistency, migration, and evidence requirements for authoritative state systems.

WHY THIS STANDARD EXISTS	The architecture of state management has failed.
OPERATIONAL SCOPE	This standard covers the evaluation of: - Event Sourcing architectures (append-only logs, event stores) - Command Query Responsibility Segregation (CQRS) patterns and projections - Durable execution runtimes (Temporal, Restate, DBOS) - Workflow engines and distributed state machines (Argo, Airflow, Cadence) - Formal verification of state machines (TLA+, Apalache, Promela/SPIN) - Event schema evolution and versioning - Local-first event streams (SQLite-based WALs)
PRIMARY AUDIENCE	- Principal engineers and software architects designing distributed systems - Platform engineers building durable workflow infrastructure - AI engineers building resilient, crash-proof agent execution loops - SREs managing long-running processes - Standards bodies seeking a reference state management methodology



Assurance principle. Model capability, data quality, identity, authority, operational evidence, and human accountability are treated as a connected system. A high aggregate score cannot compensate for a failed mandatory safety or authority boundary.

Contents

Evaluation criterion index

This standard contains 12 atomic criteria. Each criterion is independently testable and requires retained evidence.

ID	EVALUATION CRITERION
4.1.1	State Derivation
4.2.1	Crash Survival
4.3.1	Read/Write Segregation
4.4.1	Transition Correctness
4.5.1	Replay Safety
4.6.1	Event Versioning
5.1.1	MSDS-Durability
5.1.2	MSDS-Replay
6.1.1	Non-Deterministic Replay (Double-Spend)
6.1.2	Poisoned Event Stream
6.1.3	Unbounded Saga Rollback
6.1.4	State Machine Deadlock

MYTHOS-DAT-0004-4.1.1

4.1.1 State Derivation

Normative source requirement

Is the current state derived from an immutable history of events?

Score	Criteria
0	State mutated in-place via CRUD operations
1	Audit log exists, but system reads/writes from the mutable state
2	Events stored, but state is cached and mutated directly
3	Pure Event Sourcing: state is strictly a projection of the event log
4	Event log backed by consensus (Raft/Paxos) with cryptographic linking
5	Perfect sourcing: formally verified event log, zero mutation capability, BLAKE3 hash-chained events

---

CONTROL INTENT	REQUIRED EVIDENCE
Create a measurable, reviewable control that can be verified independently of model or vendor claims.	Reproducible benchmark results, workload definitions, raw outputs, and variance records.
ASSESSMENT METHOD	MATURITY SIGNAL
Run a version-pinned workload with representative inputs, record distributions rather than a single score, repeat under failure and load, and compare reproduced results with the stated acceptance threshold.	0 absent | 1 ad hoc | 2 repeatable | 3 controlled | 4 measured | 5 continuously verified

Decision rule. A criterion is not satisfied by documentation alone when the claim concerns runtime behavior. Reproduced evidence, responsible ownership, known limitations, and controlled exceptions are required.

MYTHOS-DAT-0004-4.2.1

4.2.1 Crash Survival

Normative source requirement

Can an executing workflow/agent survive a process crash and resume exactly where it left off?

Score	Criteria
0	In-memory execution; state lost on crash
1	Checkpoints saved periodically to a database
2	Application-level retry logic with idempotency keys
3	Durable execution runtime (Temporal/Restate) persisting state after every step
4	Durable execution with automatic deadlock detection and timeout recovery
5	Perfect durability: formally verified execution replay, zero loss of in-flight state, sub-second recovery time

---

CONTROL INTENT	REQUIRED EVIDENCE
Create a measurable, reviewable control that can be verified independently of model or vendor claims.	Reproducible benchmark results, workload definitions, raw outputs, and variance records. Context manifests, retrieval traces, source citations, freshness records, conflict decisions, and memory provenance.
ASSESSMENT METHOD	MATURITY SIGNAL
Run a version-pinned workload with representative inputs, record distributions rather than a single score, repeat under failure and load, and compare reproduced results with the stated acceptance threshold.	0 absent | 1 ad hoc | 2 repeatable | 3 controlled | 4 measured | 5 continuously verified

Decision rule. A criterion is not satisfied by documentation alone when the claim concerns runtime behavior. Reproduced evidence, responsible ownership, known limitations, and controlled exceptions are required.

MYTHOS-DAT-0004-4.3.1

4.3.1 Read/Write Segregation

Normative source requirement

Are read models decoupled from the write model (event log)?

Score	Criteria
0	Single model used for both reads and writes
1	Database read replicas used for queries
2	Separate logical views, but same physical database
3	True CQRS: dedicated projection builders consuming events to build materialized views
4	Multiple specialized read models (e.g., Graph DB for relationships, Search DB for text) built from one event stream
5	Perfect CQRS: zero read/write contention, eventually consistent projections with lag monitoring, automated projection rebuilding

---

CONTROL INTENT	REQUIRED EVIDENCE
Create a measurable, reviewable control that can be verified independently of model or vendor claims.	Reproducible benchmark results, workload definitions, raw outputs, and variance records. Model and dataset cards, lineage, licenses, evaluation reports, release approvals, and rollback artifacts.
ASSESSMENT METHOD	MATURITY SIGNAL
Run a version-pinned workload with representative inputs, record distributions rather than a single score, repeat under failure and load, and compare reproduced results with the stated acceptance threshold.	0 absent | 1 ad hoc | 2 repeatable | 3 controlled | 4 measured | 5 continuously verified

Decision rule. A criterion is not satisfied by documentation alone when the claim concerns runtime behavior. Reproduced evidence, responsible ownership, known limitations, and controlled exceptions are required.

MYTHOS-DAT-0004-4.4.1

4.4.1 Transition Correctness

Normative source requirement

Is the state machine mathematically proven to be correct?

Score	Criteria
0	State transitions handled by ad-hoc `if/else` statements
1	Enum-based state machine in code
2	Framework-based state machine (XState) with defined transitions
3	State machine modeled in a formal language (TLA+/Apalache)
4	Formal model proves absence of deadlocks, livelocks, and illegal transitions
5	Perfect verification: formally verified state machine, CI/CD gates blocking merges that violate the model, mathematical proof of termination

---

CONTROL INTENT	REQUIRED EVIDENCE
Create a measurable, reviewable control that can be verified independently of model or vendor claims.	Reproducible benchmark results, workload definitions, raw outputs, and variance records. Model and dataset cards, lineage, licenses, evaluation reports, release approvals, and rollback artifacts.
ASSESSMENT METHOD	**MATURITY SIGNAL**
Run a version-pinned workload with representative inputs, record distributions rather than a single score, repeat under failure and load, and compare reproduced results with the stated acceptance threshold.	0 absent | 1 ad hoc | 2 repeatable | 3 controlled | 4 measured | 5 continuously verified

Decision rule. A criterion is not satisfied by documentation alone when the claim concerns runtime behavior. Reproduced evidence, responsible ownership, known limitations, and controlled exceptions are required.

MYTHOS-DAT-0004-4.5.1

4.5.1 Replay Safety

Normative source requirement

Can the event log be replayed without causing side-effects (double-charges, duplicate messages)?

Score	Criteria
0	Replay causes side-effects to re-execute
1	Manual idempotency checks in application code
2	External side-effects guarded by idempotency keys
3	Durable runtime intercepts side-effects (deterministic execution context)
4	Replay engine executes entirely offline, replaying only external responses
5	Perfect determinism: formally verified deterministic replay, zero non-deterministic functions allowed, mathematical proof of idempotency

---

CONTROL INTENT	REQUIRED EVIDENCE
Create a measurable, reviewable control that can be verified independently of model or vendor claims.	Reproducible benchmark results, workload definitions, raw outputs, and variance records. Context manifests, retrieval traces, source citations, freshness records, conflict decisions, and memory provenance.
ASSESSMENT METHOD	MATURITY SIGNAL
Run a version-pinned workload with representative inputs, record distributions rather than a single score, repeat under failure and load, and compare reproduced results with the stated acceptance threshold.	0 absent | 1 ad hoc | 2 repeatable | 3 controlled | 4 measured | 5 continuously verified

Decision rule. A criterion is not satisfied by documentation alone when the claim concerns runtime behavior. Reproduced evidence, responsible ownership, known limitations, and controlled exceptions are required.

MYTHOS-DAT-0004-4.6.1

4.6.1 Event Versioning

Normative source requirement

Can event schemas evolve without breaking historical replay?

Score	Criteria
0	No versioning; schema changes break replay
1	Versioned events but manual migration scripts required
2	Backward-compatible schema evolution (Protobuf/Avro)
3	Upcasters/migrators automatically transform old events to new schemas during replay
4	Schema registry enforcing compatibility rules (e.g., no field deletion without default)
5	Perfect evolution: formally verified schema compatibility, zero replay breakage, automated upcaster testing

---

CONTROL INTENT	REQUIRED EVIDENCE
Create a measurable, reviewable control that can be verified independently of model or vendor claims.	Reproducible benchmark results, workload definitions, raw outputs, and variance records.
ASSESSMENT METHOD	MATURITY SIGNAL
Run a version-pinned workload with representative inputs, record distributions rather than a single score, repeat under failure and load, and compare reproduced results with the stated acceptance threshold.	0 absent | 1 ad hoc | 2 repeatable | 3 controlled | 4 measured | 5 continuously verified

Decision rule. A criterion is not satisfied by documentation alone when the claim concerns runtime behavior. Reproduced evidence, responsible ownership, known limitations, and controlled exceptions are required.

MYTHOS-DAT-0004-5.1.1

5.1.1 MSDS-Durability

Normative source requirement

- Kill Test: 100+ tests killing the executing process at every possible step in the workflow, verifying it resumes correctly.
- Zombie Test: 50+ tests simulating network partitions to verify the system does not spawn duplicate workflows.

CONTROL INTENT	REQUIRED EVIDENCE
Create a measurable, reviewable control that can be verified independently of model or vendor claims.	Architecture records, implemented configuration, test evidence, operational telemetry, exception decisions, and accountable approval.
ASSESSMENT METHOD	MATURITY SIGNAL
Inspect the architecture and configured control, execute normal and adverse scenarios, verify measurable outcomes, sample retained evidence, and confirm that exceptions are time-bound and approved.	0 absent | 1 ad hoc | 2 repeatable | 3 controlled | 4 measured | 5 continuously verified

Decision rule. A criterion is not satisfied by documentation alone when the claim concerns runtime behavior. Reproduced evidence, responsible ownership, known limitations, and controlled exceptions are required.

MYTHOS-DAT-0004-5.1.2

5.1.2 MSDS-Replay

Normative source requirement

- Side-Effect Isolation: 100+ tests replaying the event log to verify no external API is called twice.
- Schema Migration: 50+ tests applying a new schema version to a historical event log to verify upcasters function correctly.

CONTROL INTENT	REQUIRED EVIDENCE
Create a measurable, reviewable control that can be verified independently of model or vendor claims.	Architecture records, implemented configuration, test evidence, operational telemetry, exception decisions, and accountable approval.
ASSESSMENT METHOD	MATURITY SIGNAL
Exercise crash, retry, duplicate, reorder, partition, and restore scenarios; compare authoritative state before and after replay; confirm idempotency and recovery evidence.	0 absent | 1 ad hoc | 2 repeatable | 3 controlled | 4 measured | 5 continuously verified

Decision rule. A criterion is not satisfied by documentation alone when the claim concerns runtime behavior. Reproduced evidence, responsible ownership, known limitations, and controlled exceptions are required.

MYTHOS-DAT-0004-6.1.1

6.1.1 Non-Deterministic Replay (Double-Spend)

Normative source requirement

Severity: Critical Description: A workflow is replayed, but a non-deterministic function (e.g., `Date.now()` or `Math.random()`) causes a different execution path, resulting in a duplicate side-effect (e.g., charging a user twice).

Required Controls: 1. Durable execution runtimes that intercept and mock non-deterministic APIs. 2. Strict prohibition of non-deterministic functions in workflow code. 3. Automated replay testing in CI/CD.

CONTROL INTENT	REQUIRED EVIDENCE
Create a measurable, reviewable control that can be verified independently of model or vendor claims.	Architecture records, implemented configuration, test evidence, operational telemetry, exception decisions, and accountable approval.
ASSESSMENT METHOD	MATURITY SIGNAL
Exercise crash, retry, duplicate, reorder, partition, and restore scenarios; compare authoritative state before and after replay; confirm idempotency and recovery evidence.	0 absent | 1 ad hoc | 2 repeatable | 3 controlled | 4 measured | 5 continuously verified

Decision rule. A criterion is not satisfied by documentation alone when the claim concerns runtime behavior. Reproduced evidence, responsible ownership, known limitations, and controlled exceptions are required.

MYTHOS-DAT-0004-6.1.2

6.1.2 Poisoned Event Stream

Normative source requirement

Severity: High Description: A malicious or buggy service writes an invalid event to the log, corrupting the state projection and causing all replays to fail.

Required Controls: 1. Strict schema validation at the event broker/store boundary. 2. Dead-letter queues for un-parseable events. 3. Versioned schemas to ensure backward compatibility.

CONTROL INTENT	REQUIRED EVIDENCE
Create a measurable, reviewable control that can be verified independently of model or vendor claims.	Architecture records, implemented configuration, test evidence, operational telemetry, exception decisions, and accountable approval.
ASSESSMENT METHOD	MATURITY SIGNAL
Trace the decision path from identity and policy through execution, attempt bypass and privilege-escalation scenarios, verify revocation behavior, and inspect the resulting evidence record.	0 absent | 1 ad hoc | 2 repeatable | 3 controlled | 4 measured | 5 continuously verified

Decision rule. A criterion is not satisfied by documentation alone when the claim concerns runtime behavior. Reproduced evidence, responsible ownership, known limitations, and controlled exceptions are required.

MYTHOS-DAT-0004-6.1.3

6.1.3 Unbounded Saga Rollback

Normative source requirement

Severity: High Description: A distributed transaction (Saga) fails, but the compensating action also fails, leaving the system in an inconsistent state.

Required Controls: 1. Durable execution to retry compensating actions until success. 2. Formal verification of the Saga state machine to prove it eventually reaches a consistent state. 3. Alerting on stuck "RollingBack" states.

CONTROL INTENT	REQUIRED EVIDENCE
Create a measurable, reviewable control that can be verified independently of model or vendor claims.	Architecture records, implemented configuration, test evidence, operational telemetry, exception decisions, and accountable approval.
ASSESSMENT METHOD	MATURITY SIGNAL
Exercise crash, retry, duplicate, reorder, partition, and restore scenarios; compare authoritative state before and after replay; confirm idempotency and recovery evidence.	0 absent | 1 ad hoc | 2 repeatable | 3 controlled | 4 measured | 5 continuously verified

Decision rule. A criterion is not satisfied by documentation alone when the claim concerns runtime behavior. Reproduced evidence, responsible ownership, known limitations, and controlled exceptions are required.

MYTHOS-DAT-0004-6.1.4

6.1.4 State Machine Deadlock

Normative source requirement

Severity: High Description: Two workflows wait on resources held by each other, permanently halting progress.

Required Controls: 1. TLA+/Apalache formal verification to prove absence of deadlock. 2. Timeouts on all state transitions.

CONTROL INTENT	REQUIRED EVIDENCE
Create a measurable, reviewable control that can be verified independently of model or vendor claims.	Context manifests, retrieval traces, source citations, freshness records, conflict decisions, and memory provenance.
ASSESSMENT METHOD	MATURITY SIGNAL
Inject stale, conflicting, low-authority, and malicious information; inspect selection and citation behavior; verify scope isolation, correction, deletion, and provenance retention.	0 absent | 1 ad hoc | 2 repeatable | 3 controlled | 4 measured | 5 continuously verified

Decision rule. A criterion is not satisfied by documentation alone when the claim concerns runtime behavior. Reproduced evidence, responsible ownership, known limitations, and controlled exceptions are required.

Current authority and freshness register

These sources inform interpretation but do not convert this candidate publication into certification, legal compliance, or implementation evidence. Rolling sources must be version-pinned at adoption time.

AUTHORITY	VERSION / FRESHNESS	APPLICABILITY LIMIT
NIST - Artificial Intelligence Risk Management Framework (AI RMF 1.0)	NIST AI 100-1; current framework, revision underway Expires 2026-10-24	Voluntary risk-management framework; does not establish product certification or implementation effectiveness.
OpenTelemetry - Semantic Conventions	1.43.0 rolling specification; GenAI conventions maintained separately Expires 2026-08-24	Several GenAI and agent conventions remain under active development and require version pinning.
W3C - Verifiable Credentials Data Model v2.0	W3C Recommendation, May 15 2025 Expires 2027-01-24	Data model conformance does not establish issuer trust, credential truth, or ecosystem governance.
C2PA - Content Credentials Technical Specification	2.4 rolling publication Expires 2026-10-24	Provenance establishes signed assertions and tamper evidence, not the truth or quality of the underlying content.

Source lineage appendix

The following text preserves the substantive source draft in a normalized public form. Where it conflicts with the permanent publication identity, candidate status, current authority register, or evidence requirements above, the controlled publication layer governs.

0. Executive Mandate

The architecture of state management has failed.

Organizations build complex workflows and autonomous agents using CRUD (Create, Read, Update, Delete) databases. They overwrite the current state of an entity, erasing its history. When an agent crashes mid-execution, it leaves the system in a zombie state-half-complete and unrecoverable. When a bug occurs, engineers stare at a single database row, utterly incapable of answering the fundamental question: "How did the system arrive in this state?"

This standard exists because:

1. CRUD Destroys Truth: Overwriting state destroys history. The current state of a system is merely a projection of its past events. The events themselves are the absolute truth. Systems MUST be modeled as append-only event logs.

2. In-Memory Execution is Fragile: An agent or workflow that relies on in-memory variables to track its progress will lose that progress the moment its container is OOM-killed or the pod migrates. Execution MUST be durable, persisted after every step, and resumable from exactly the point of failure.

3. Read/Write Coupling Kills Scale: Forcing the same data model to serve high-volume transactional writes and complex analytical queries results in lock contention, slow APIs, and rigid schemas. Command Query Responsibility Segregation (CQRS) MUST separate the write model (events) from the read models (projections).

4. State Machines Must Be Proven, Not Hoped: An agent's lifecycle (e.g., `Planning -> AwaitingApproval -> Executing -> Verifying`) is a state machine. Deploying a state machine without mathematical proof of its correctness guarantees deadlocks, livelocks, and illegal transitions. State machines MUST be formally verified.

This document establishes the Mythos Event Sourcing & Durable State Standard (MESDSS), a comprehensive, evidence-based methodology for evaluating state architectures against event immutability, durable execution, and formal verification.

Part I. Scope and Applicability

1.1 In Scope

This standard covers the evaluation of:

- Event Sourcing architectures (append-only logs, event stores)
- Command Query Responsibility Segregation (CQRS) patterns and projections
- Durable execution runtimes (Temporal, Restate, DBOS)
- Workflow engines and distributed state machines (Argo, Airflow, Cadence)
- Formal verification of state machines (TLA+, Apalache, Promela/SPIN)
- Event schema evolution and versioning
- Local-first event streams (SQLite-based WALs)

1.2 Out of Scope

- General database design and indexing (covered in future MYTHOS-DBS standards)
- General domain modeling (covered in MYTHOS-SWE-0001)
- Tamper-evident ledgers and cryptographic audit trails (covered in MYTHOS-DAT-0005)

1.3 Audience

- Principal engineers and software architects designing distributed systems
- Platform engineers building durable workflow infrastructure
- AI engineers building resilient, crash-proof agent execution loops
- SREs managing long-running processes
- Standards bodies seeking a reference state management methodology

Part II. Definitions and Taxonomy

2.1 State Dimensions

Dimension	Definition
Event Sourcing	A pattern where state changes are stored as a sequence of immutable, append-only events. The current state is derived by replaying the events.
Durable Execution	A runtime paradigm where the execution state (variables, stack, program counter) is persisted after every step, allowing the process to survive crashes and resume seamlessly.
CQRS	Command Query Responsibility Segregation. Separating the data mutation path (Commands -> Events) from the data retrieval path (Queries -> Materialized Views).
Deterministic Replay	The ability to replay an event log or execution history and arrive at the exact same state, guaranteeing no side-effects are re-executed.
Saga	A sequence of local transactions where each transaction updates the database and publishes a message/event to trigger the next step, with compensating actions for failures.

2.2 The State Architecture Doctrine

> A database that overwrites its past has no memory. A workflow that cannot survive a crash is a liability. State is a projection of events; events are the absolute truth. A state machine that is not formally verified is a deadlock waiting to happen.

Part III. Evaluation Methodology

3.1 Scoring Model

Each capability is scored from 0 to 5.

Score	Meaning
0	Fails basic task; CRUD, in-memory execution, unverified state machines
1	Basic audit logging but still CRUD-based
2	Functional event streaming but lacks durable execution or CQRS
3	Solid production performance; Event Sourcing, CQRS, durable runtime (Temporal)
4	Strong performance; deterministic replay, schema evolution, formal state models
5	Best-in-class; sets the standard for mathematically proven, zero-loss durable state

Weighting

Category	Weight	Rationale
Event Sourcing & Immutability	20%	CRUD destroys history; events are the source of truth
Durable Execution & Recovery	20%	In-memory workflows die on crash; execution must be durable
CQRS & Projection Management	15%	Coupling reads and writes limits scale and agility
State Machine Formal Verification	15%	Unverified state machines deadlock
Determinism & Replay	15%	Non-deterministic replay causes double-execution bugs
Schema Evolution & Versioning	15%	Events are immortal; schemas must evolve without breaking

Total: 100%

A system MUST score at least 3.0 in Event Sourcing and Durable Execution to be considered for production use.

Part IV. Evaluation Categories

4.1 Event Sourcing and Immutability (Weight: 20%)

4.1.1 State Derivation

Is the current state derived from an immutable history of events?

Score	Criteria
0	State mutated in-place via CRUD operations
1	Audit log exists, but system reads/writes from the mutable state
2	Events stored, but state is cached and mutated directly
3	Pure Event Sourcing: state is strictly a projection of the event log
4	Event log backed by consensus (Raft/Paxos) with cryptographic linking
5	Perfect sourcing: formally verified event log, zero mutation capability, BLAKE3 hash-chained events

4.2 Durable Execution and Recovery (Weight: 20%)

4.2.1 Crash Survival

Can an executing workflow/agent survive a process crash and resume exactly where it left off?

Score	Criteria
0	In-memory execution; state lost on crash
1	Checkpoints saved periodically to a database
2	Application-level retry logic with idempotency keys
3	Durable execution runtime (Temporal/Restate) persisting state after every step
4	Durable execution with automatic deadlock detection and timeout recovery
5	Perfect durability: formally verified execution replay, zero loss of in-flight state, sub-second recovery time

4.3 CQRS and Projection Management (Weight: 15%)

4.3.1 Read/Write Segregation

Are read models decoupled from the write model (event log)?

Score	Criteria
0	Single model used for both reads and writes
1	Database read replicas used for queries
2	Separate logical views, but same physical database
3	True CQRS: dedicated projection builders consuming events to build materialized views
4	Multiple specialized read models (e.g., Graph DB for relationships, Search DB for text) built from one event stream
5	Perfect CQRS: zero read/write contention, eventually consistent projections with lag monitoring, automated projection rebuilding

4.4 State Machine Formal Verification (Weight: 15%)

4.4.1 Transition Correctness

Is the state machine mathematically proven to be correct?

Score	Criteria
0	State transitions handled by ad-hoc `if/else` statements
1	Enum-based state machine in code
2	Framework-based state machine (XState) with defined transitions
3	State machine modeled in a formal language (TLA+/Apalache)
4	Formal model proves absence of deadlocks, livelocks, and illegal transitions
5	Perfect verification: formally verified state machine, CI/CD gates blocking merges that violate the model, mathematical proof of termination

4.5 Determinism and Replay (Weight: 15%)

4.5.1 Replay Safety

Can the event log be replayed without causing side-effects (double-charges, duplicate messages)?

Score	Criteria
0	Replay causes side-effects to re-execute
1	Manual idempotency checks in application code
2	External side-effects guarded by idempotency keys
3	Durable runtime intercepts side-effects (deterministic execution context)
4	Replay engine executes entirely offline, replaying only external responses
5	Perfect determinism: formally verified deterministic replay, zero non-deterministic functions allowed, mathematical proof of idempotency

4.6 Schema Evolution and Versioning (Weight: 15%)

4.6.1 Event Versioning

Can event schemas evolve without breaking historical replay?

Score	Criteria
0	No versioning; schema changes break replay
1	Versioned events but manual migration scripts required
2	Backward-compatible schema evolution (Protobuf/Avro)
3	Upcasters/migrators automatically transform old events to new schemas during replay
4	Schema registry enforcing compatibility rules (e.g., no field deletion without default)
5	Perfect evolution: formally verified schema compatibility, zero replay breakage, automated upcaster testing

Part V. The Mythos State & Durability Suite (MSDS-State)

Traditional database benchmarks measure transaction throughput (TPS). The MSDS-State suite evaluates crash recovery, replay safety, and state machine correctness.

5.1 Suite Composition

5.1.1 MSDS-Durability

- Kill Test: 100+ tests killing the executing process at every possible step in the workflow, verifying it resumes correctly.
- Zombie Test: 50+ tests simulating network partitions to verify the system does not spawn duplicate workflows.

5.1.2 MSDS-Replay

- Side-Effect Isolation: 100+ tests replaying the event log to verify no external API is called twice.
- Schema Migration: 50+ tests applying a new schema version to a historical event log to verify upcasters function correctly.

5.2 Execution Protocol

```
1. Deploy state architecture with durable execution runtime
2. Execute complex, multi-step workflows with external side-effects
3. Run MSDS-State suite (automated crash injection + replay)
4. Score each category 0-5
5. Check for in-place state mutations for core logic (instant disqualification)
6. If all thresholds met: approve for production
```

Part VI. Security Threat Model for State Management

6.1 Threat Catalog

6.1.1 Non-Deterministic Replay (Double-Spend)

Severity: Critical Description: A workflow is replayed, but a non-deterministic function (e.g., `Date.now()` or `Math.random()`) causes a different execution path, resulting in a duplicate side-effect (e.g., charging a user twice).

Required Controls:

1. Durable execution runtimes that intercept and mock non-deterministic APIs.

2. Strict prohibition of non-deterministic functions in workflow code.

3. Automated replay testing in CI/CD.

6.1.2 Poisoned Event Stream

Severity: High Description: A malicious or buggy service writes an invalid event to the log, corrupting the state projection and causing all replays to fail.

Required Controls:

1. Strict schema validation at the event broker/store boundary.

2. Dead-letter queues for un-parseable events.

3. Versioned schemas to ensure backward compatibility.

6.1.3 Unbounded Saga Rollback

Severity: High Description: A distributed transaction (Saga) fails, but the compensating action also fails, leaving the system in an inconsistent state.

Required Controls:

1. Durable execution to retry compensating actions until success.

2. Formal verification of the Saga state machine to prove it eventually reaches a consistent state.

3. Alerting on stuck "RollingBack" states.

6.1.4 State Machine Deadlock

Severity: High Description: Two workflows wait on resources held by each other, permanently halting progress.

Required Controls:

1. TLA+/Apalache formal verification to prove absence of deadlock.

2. Timeouts on all state transitions.

Part VII. The Mythos State & Durability Standard

7.1 The Mythos State Doctrine

```
A database that overwrites its past has no memory.
A workflow that cannot survive a crash is a liability.

State is a projection of events; events are the absolute truth.
A state machine that is not formally verified is a deadlock waiting to happen.
```

```
State may be eventual.
Durability must be absolute.
```

7.2 Selection Rules

1. No state architecture enters production without passing MSDS-State.
2. No architecture is approved if it mutates state in-place for core domain logic.
3. No architecture is approved if its workflows cannot survive a process crash.
4. No architecture is approved without deterministic replay for side-effects.
5. No state machine is approved without formal verification of transitions.

7.3 The Prime Directive for State Architecture

> Append the event, do not mutate the row. > Persist the step, do not trust the memory. > Segregate the read, do not lock the write. > Prove the transition, do not hope the logic.

```
Academic benchmarks measure transactions per second.
Applied benchmarks measure crash recovery.
Security benchmarks measure replay safety.
Operational benchmarks measure state machine correctness.

> Processes may die.
> State must be absolute.
```

End of Standard MYTHOS-DAT-0004

End of controlled publication

MYTHOS-DAT-0004 remains a Candidate Standard until technical review, security and privacy review, accessibility review, current-source validation, and formal approval are recorded.