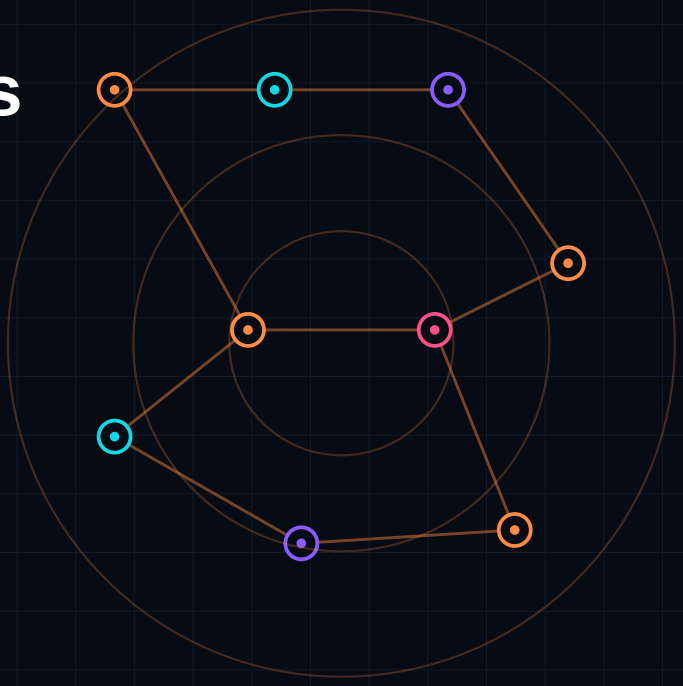
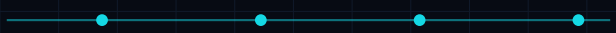
MYTHOS SYSTEMS

ENGINEERING STANDARDS LIBRARY / PORTFOLIO EDITION

Distributed Systems, Database, and Reliability Standards Portfolio

Durable workflows, service contracts, databases, storage, isolation, SLOs, capacity, recovery, failure testing, and engineering economics.





PERMANENT DOCUMENT ID
MYTHOS-DSTDBSRE-PORT-0001

PUBLICATION
Candidate Portfolio Edition

VERSION
1.0.0-candidate

PERMANENT PUBLICATION IDENTITY

Distributed Systems, Database, and Reliability Standards Portfolio

DOCUMENT ID
MYTHOS-DSTDBSRE-PORT-0001

VERSION
1.0.0-candidate

STATUS
Candidate Portfolio Edition

PUBLISHER
Mythos Systems

PUBLICATION DATE
2026-09-17

CLASSIFICATION
Public

8	94	8	1
STANDARDS	CRITERIA	ATOMIC SOURCES	PERMANENT IDENTITY

Portfolio doctrine. This generated reader view does not replace its atomic source publications. Permanent IDs, evidence boundaries, assessment scope, freshness, and revision history remain authoritative at the atomic publication level.

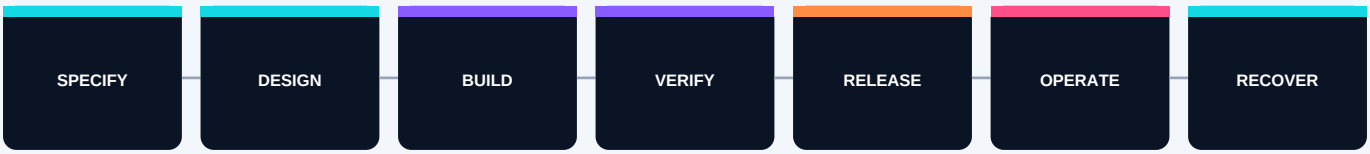
Publication domains

- Durable workflows and event-driven systems
- API, gRPC, and service-mesh contracts
- Database and storage architecture
- Tenant and data-access isolation
- SLOs, capacity, disaster recovery, and FinOps

PORTFOLIO READING MAP

From code to verified recovery

A visual navigation model for the software, platform, state, and reliability lifecycle.



STATE & DATA

contracts -> events -> durable state -> replicas -> storage -> evidence

SERVICE TOPOLOGY

API -> service mesh -> queues -> workers -> dependencies -> users

RELIABILITY

SLI -> SLO -> error budget -> capacity -> failure test -> recovery proof

ENGINEERING STANDARDS LIBRARY / DISTRIBUTED SYSTEMS

Durable Workflows, Queues, and Event-Driven Sagas Standard

Durable execution, message delivery semantics, idempotency, compensation, event ordering, and recoverable distributed workflows.

PERMANENT DOCUMENT ID

MYTHOS-DST-0001

PUBLICATION

Candidate Standard
Version 1.0.0-candidate

ASSURANCE PROFILE

Standard / Advanced
High Assurance

PERMANENT PUBLICATION IDENTITY

Durable Workflows, Queues, and Event-Driven Sagas Standard

DOCUMENT ID
MYTHOS-DST-0001

VERSION
1.0.0-candidate

STATUS
Candidate Standard

PUBLISHER
Mythos Systems

PUBLICATION DATE
2026-07-24

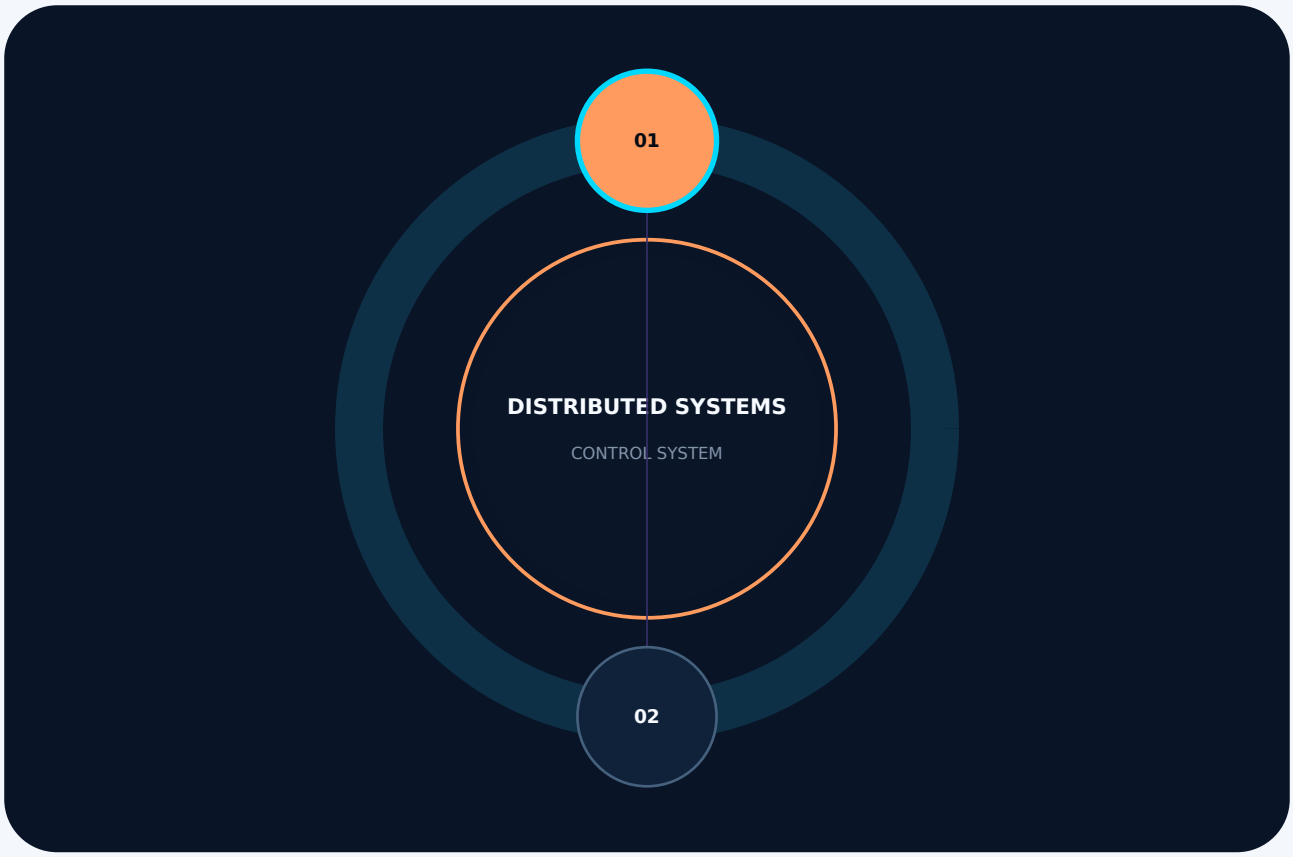
SCHEDULED REVIEW
2027-01-24

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

Publication doctrine. This publication establishes durable guidance or requirements for its defined scope. Interpretation must preserve the permanent document identity, recorded evidence, controlled exceptions, and formal revision history.

DOMAIN CONTROL SYSTEM

MYTHOS-DST-0001 inside the distributed systems standard constellation

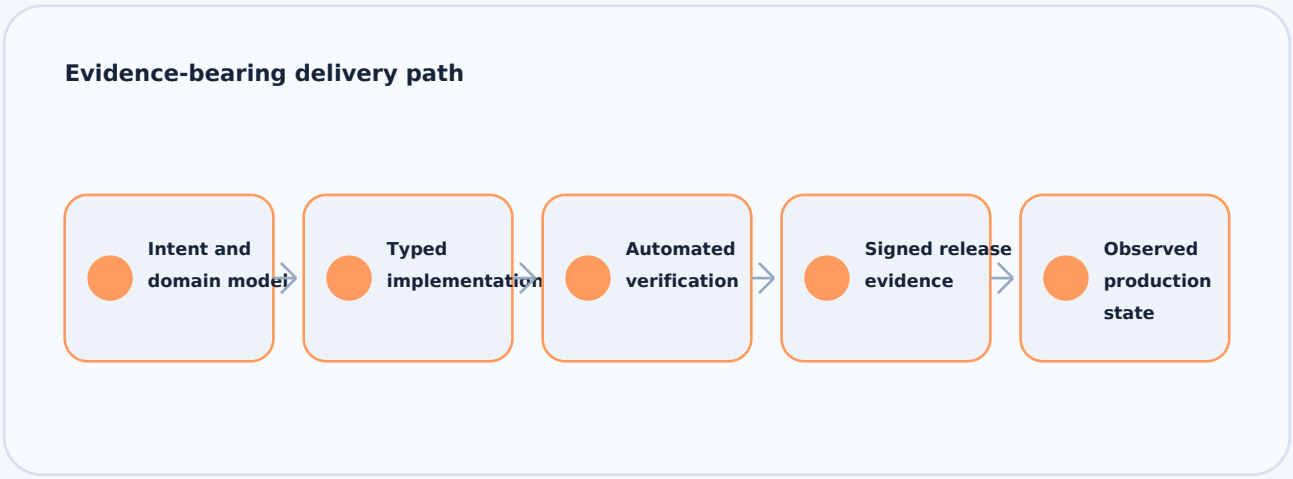


Connected assurance	Architecture, implementation, operations, evidence, and lifecycle are evaluated as one controlled system.
Specialization rule	Companion standards may add precision, but they cannot silently weaken this publication's requirements.

EXECUTIVE POSITION

Durable Workflows, Queues, and Event-Driven Sagas Standard

The architecture of distributed systems has failed.



CONTENTS

Publication navigation

MYTHOS-DST-0001 inside the distributed systems standard constellation . . . 3

Durable Workflows, Queues, and Event-Driven Sagas Standard . . . 4

Assurance model and evaluation surface . . . 7

Criteria and evidence map . . . 8

Requirements, evaluation methodology, and implementation guidance . . . 9

Executive Mandate . . . 9

Scope and Applicability . . . 9

In Scope . . . 9

Out of Scope . . . 10

Audience . . . 10

Definitions and Taxonomy . . . 10

Workflow Dimensions . . . 10

The Distributed Systems Doctrine . . . 11

Evaluation Methodology . . . 11

Scoring Model . . . 11

Weighting . . . 11

Evaluation Categories . . . 12

Durable Execution and Crash Recovery (Weight: 20%) . . . 12

Workflow Resilience . . . 12

Event-Driven Delivery (Transactional Outbox) (Weight: 20%) . . . 12

Dual-Write Elimination . . . 12

Saga and Distributed Transactions (Weight: 20%) . . . 12

Cross-Service Consistency . . . 12

Idempotency and Exactly-Once (Weight: 15%) . . . 13

Message Processing Guarantees . . . 13

Queue Management and Backpressure (Weight: 15%) . . . 13

Load Shedding . . . 13

Poison Message and DLQ Handling (Weight: 10%) . . . 13

Fault Isolation . . . 13

The Mythos Distributed Systems Suite (MDSS) 14

Suite Composition 14

MDSS-Durability 14

MDSS-Consistency 14

Execution Protocol 14

Implementation evidence and review gates 15

Current authority register 16

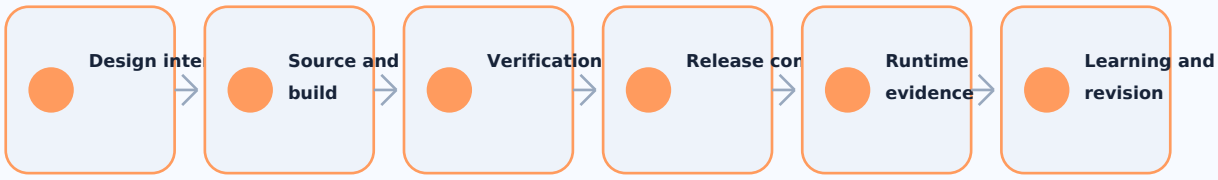
CONTROL TOPOLOGY

Assurance model and evaluation surface

The standard is assessed as an evidence-bearing system. Capability scores do not replace mandatory gates, implementation proof, or operational validation.

Capability family	Evaluation nodes
Durable Execution and Crash Recovery (Weight: 20%)	1
Event-Driven Delivery (Transactional Outbox) (Weight: 20%)	1
Saga and Distributed Transactions (Weight: 20%)	1
Idempotency and Exactly-Once (Weight: 15%)	1
Queue Management and Backpressure (Weight: 15%)	1
Poison Message and DLQ Handling (Weight: 10%)	1
Suite Composition	2
Additional Evaluation Dimension	4

Lifecycle control loop



EVALUATION INDEX

Criteria and evidence map

#	Capability family	Criterion
1	Durable Execution and Crash Recovery (Weight: 20%)	Workflow Resilience
2	Event-Driven Delivery (Transactional Outbox) (Weight: 20%)	Dual-Write Elimination
3	Saga and Distributed Transactions (Weight: 20%)	Cross-Service Consistency
4	Idempotency and Exactly-Once (Weight: 15%)	Message Processing Guarantees
5	Queue Management and Backpressure (Weight: 15%)	Load Shedding
6	Poison Message and DLQ Handling (Weight: 10%)	Fault Isolation
7	Suite Composition	MDSS-Durability
8	Suite Composition	MDSS-Consistency
9	Additional Evaluation Dimension	Workflow Dimensions
10	Additional Evaluation Dimension	The Distributed Systems Doctrine
11	Additional Evaluation Dimension	Scoring Model
12	Additional Evaluation Dimension	Suite Composition

NORMATIVE PUBLICATION

Requirements, evaluation methodology, and implementation guidance

The following body preserves the substantive source standard while normalizing publication status, identity, navigation, and visual presentation.

Executive Mandate

The architecture of distributed systems has failed.

Organizations attempt to coordinate complex, multi-step business logic and autonomous AI agent tasks using synchronous HTTP REST calls and in-memory state. When a downstream service times out, the upstream service is left in a zombie state. When a process crashes mid-execution, the workflow is lost forever. To solve data consistency across microservices, they attempt to implement Two-Phase Commits (2PC), which lock databases and guarantee cascading failures under network partitions.

This standard exists because:

1. Synchronous Coupling is a Distributed Monolith: If Service A cannot complete its task because Service B is offline, you do not have microservices; you have a distributed monolith. Inter-service communication MUST be decoupled via asynchronous message queues or durable orchestration runtimes.
2. In-Memory Workflows are Fragile: An agent or business workflow that relies on in-memory variables to track its progress through a 10-step process will lose all progress the moment its container is OOM-killed or the pod migrates. Execution MUST be durable, persisting state after every step, and resumable from exactly the point of failure.
3. Two-Phase Commits (2PC) are Prohibited: Distributed transactions that lock resources across multiple microservices do not scale and deadlock under network partitions. Systems MUST implement the Saga pattern, where a distributed transaction is broken into a sequence of local transactions, each with a deterministic compensating action to roll back on failure.
4. The Dual-Write Problem is Inevitable: Updating a database and then publishing an event to a message broker is a guaranteed data loss vector. If the broker is offline, the database update is committed, but the event is lost. Systems MUST implement the Transactional Outbox pattern, writing the state change and the event in the *same* local database transaction, with a separate process (CDC) reliably publishing the event.

This document establishes the Mythos Distributed Systems Standard (MDSS), a comprehensive, evidence-based methodology for evaluating distributed workflows against crash resilience, guaranteed delivery, and distributed transactional integrity.

Scope and Applicability

In Scope

This standard covers the evaluation of:

- Durable execution runtimes (Temporal, Restate, DBOS, Cadence)
- Message brokers and queues (Apache Kafka, NATS JetStream, RabbitMQ, SQS)
- Event-driven architecture and the Transactional Outbox pattern (CDC via Debezium)
- Saga patterns (Orchestration vs. Choreography) and compensating transactions
- Idempotency, exactly-once processing, and retry strategies (exponential backoff)
- Dead-letter queues (DLQs) and poison message isolation

Out of Scope

- General API contract design and gRPC (covered in MYTHOS-DST-0002)
- Event sourcing for state derivation (covered in MYTHOS-DAT-0004)
- General software design and DDD (covered in MYTHOS-SWE-0001)

Audience

- Distributed systems engineers and backend architects
- Platform engineers building event-driven infrastructure
- AI engineers building durable, multi-step agent loops
- SREs managing message broker fleets
- Standards bodies seeking a reference distributed systems methodology

Definitions and Taxonomy

Workflow Dimensions

Dimension	Definition
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 without re-executing side effects.
Saga	A sequence of local transactions where each transaction updates the database and publishes a message/event to trigger the next step. If a step fails, the saga executes a series of compensating transactions to rollback the preceding steps.
Transactional Outbox	A pattern that solves the dual-write problem by saving domain state changes and outgoing messages in the *same* database transaction, guaranteeing they are committed together, with a separate process reading the outbox to publish to a message broker.
Idempotency	The guarantee that processing the same message multiple times yields the same result as processing it once, achieved via unique message IDs and deduplication logic.
Compensating Transaction	An operation that semantically undoes the effects of a previously committed local transaction (e.g., issuing a refund to undo a payment), since a distributed transaction cannot be physically rolled back.

The Distributed Systems Doctrine

> A synchronous call is a chained failure. An in-memory workflow is a liability. A dual-write is guaranteed data loss. A distributed transaction without a compensating action is data corruption.

Evaluation Methodology

Scoring Model

Each capability is scored from 0 to 5.

Score	Meaning
0	Fails basic task; synchronous chains, in-memory state, 2PC, dual-writes
1	Basic message queues used, but lacks durability or idempotency
2	Functional async architecture, but lacks durable runtime or outbox pattern
3	Solid production performance; durable runtime, outbox pattern, orchestrated sagas, idempotency
4	Strong performance; exactly-once semantics, poisoned queue isolation, deterministic replay
5	Best-in-class; sets the standard for mathematically verified, zero-loss distributed systems

Weighting

Category	Weight	Rationale
Durable Execution & Crash Recovery	20%	In-memory workflows die on crash; state must be durable
Event-Driven Delivery (Outbox)	20%	Dual-writes guarantee eventual data loss
Saga & Distributed Transactions	20%	2PC deadlocks; Sagas with compensations are mandatory
Idempotency & Exactly-Once	15%	At-least-once delivery without idempotency causes double-charges
Queue Management & Backpressure	15%	Unbounded queues cause memory exhaustion and cascading failures
Poison Message & DLQ Handling	10%	A single bad message blocks the entire queue forever

Total: 100%

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

Evaluation Categories

Durable Execution and Crash Recovery (Weight: 20%)

Workflow Resilience

Can a multi-step workflow survive a process crash and resume exactly where it left off?

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

Event-Driven Delivery (Transactional Outbox) (Weight: 20%)

Dual-Write Elimination

Is the system mathematically prevented from losing events during a database commit?

Score	Criteria
0	Database updated, then event published via HTTP/broker (Dual-write risk)
1	Best-effort publishing with a background retry queue
2	Transactional Outbox implemented in application code (polling the DB)
3	CDC (Change Data Capture) via Debezium reading DB transaction logs to publish events
4	Exactly-once delivery semantics enforced end-to-end (Kafka transactions)
5	Perfect delivery: formally verified atomic state-and-event commit, zero lost or duplicate events

Saga and Distributed Transactions (Weight: 20%)

Cross-Service Consistency

How are transactions spanning multiple microservices rolled back on failure?

Score	Criteria
0	Two-Phase Commits (2PC) locking databases across services
1	Manual rollback scripts or "hope it works" methodology
2	Choreographed Sagas (event-driven, but hard to track flow)
3	Orchestrated Sagas: Central durable runtime (Temporal) commands services and executes deterministic compensating actions on failure

Score	Criteria
4	Formal verification of the Saga state machine to prove it eventually reaches a consistent state
5	Perfect consistency: formally verified compensating logic, zero stuck states, mathematical proof of eventual consistency

Idempotency and Exactly-Once (Weight: 15%)

Message Processing Guarantees

Can the system process the same message multiple times without side effects?

Score	Criteria
0	At-most-once (fire and forget) or no deduplication
1	At-least-once delivery, but consumers are not idempotent (double-charges)
2	Basic deduplication using a database unique constraint
3	Explicit idempotency keys (e.g., Stripe-style) tracked in a deduplication table
4	Exactly-once processing semantics (consumer transactions tied to message ACKs)
5	Perfect idempotency: formally verified deduplication logic, zero side effects on replay, mathematical proof of exactly-once execution

Queue Management and Backpressure (Weight: 15%)

Load Shedding

How does the system handle a surge in messages that exceeds processing capacity?

Score	Criteria
0	Unbounded queues; memory exhaustion crashes the node
1	Fixed-size queues; messages dropped when full (silent data loss)
2	Basic auto-scaling of consumers, but lag occurs
3	Explicit backpressure mechanisms; upstream producers are throttled or rejected (HTTP 429)
4	Priority-based queueing (critical messages jump the line)
5	Perfect management: formally verified load shedding, zero OOM crashes, mathematical optimization of throughput vs. latency

Poison Message and DLQ Handling (Weight: 10%)

Fault Isolation

What happens when a message cannot be processed due to malformed data or a logic bug?

Score	Criteria
0	Infinite retry loop; the poison message blocks the entire queue forever
1	Manual operator intervention required to delete the message
2	Max-retry limit, but message is dropped/deleted after limit reached

Score	Criteria
3	Automated Dead-Letter Queue (DLQ): after N retries, message moved to DLQ for analysis, queue continues processing
4	Automated alerting on DLQ depth; automated replay mechanism once the logic bug is fixed
5	Perfect isolation: formally verified poison detection, zero queue blockages, cryptographic provenance of failed messages

The Mythos Distributed Systems Suite (MDSS)

Traditional backend benchmarks measure API throughput (RPS). The MDSS evaluates crash recovery, dual-write elimination, and saga consistency.

Suite Composition

MDSS-Durability

1. Kill Test: 100+ tests killing the executing process at every possible step in a 10-step saga, verifying the durable runtime resumes it without dropping state.
1. Dual-Write Injection: 50+ tests simulating a network partition between the database and the message broker, verifying the Outbox/CDC pattern prevents data loss.

MDSS-Consistency

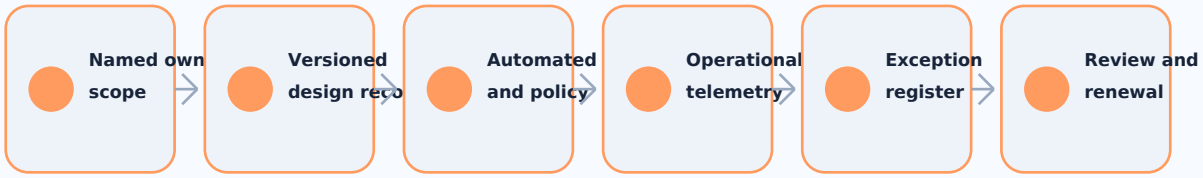
1. Saga Failure Test: 100+ tests forcing a failure at the final step of a distributed transaction, verifying all preceding services execute their compensating transactions correctly.
1. Poison Message Test: 50+ tests injecting a malformed JSON payload into the queue, verifying it is routed to the DLQ after 3 retries without blocking valid messages.

Execution Protocol

ASSURANCE AND ADOPTION

Implementation evidence and review gates

Required assurance chain



A passing score is not sufficient without current evidence. High-assurance adoption requires traceable design decisions, reproducible builds, independently reviewable tests, controlled secrets, runtime observability, failure exercises, and a dated exception register.

Assurance maturity spectrum

Current authority register

Authority	Version / snapshot	Applicability	Review by
TLA+ Foundation: TLA+ Language and Tools Documentation	Rolling documentation snapshot	High	2027-01-24
OpenAPI Initiative: OpenAPI Specification	3.2.0	High	2027-01-24
AsyncAPI Initiative: AsyncAPI Specification	3.1.0	High	2027-01-24
Kubernetes Project: Kubernetes Documentation	v1.36 current documentation	High	2026-10-22
OpenTelemetry Project: OpenTelemetry Documentation and Specification	Rolling specification snapshot; profiles public alpha and declarative configuration stable in 2026	High	2026-10-22
Cloud Native Computing Foundation: Cloud Native Landscape and Project Documentation	Rolling catalog snapshot	Moderate	2026-08-23

Research limitation. Official documentation establishes published interfaces and declared behavior. It does not prove the performance, security, interoperability, cost, or operational suitability of a specific implementation. Those claims require controlled testing and retained evidence.

ENGINEERING STANDARDS LIBRARY / DISTRIBUTED SYSTEMS

API Contracts, gRPC, and Service Mesh Standard

Versioned API contracts, RPC semantics, service-mesh boundaries, compatibility, resilience, and observable service communication.

PERMANENT DOCUMENT ID

MYTHOS-DST-0002

PUBLICATION

Candidate Standard
Version 1.0.0-candidate

ASSURANCE PROFILE

Standard / Advanced
High Assurance

PERMANENT PUBLICATION IDENTITY

API Contracts, gRPC, and Service Mesh Standard

DOCUMENT ID
MYTHOS-DST-0002

VERSION
1.0.0-candidate

STATUS
Candidate Standard

PUBLISHER
Mythos Systems

PUBLICATION DATE
2026-07-24

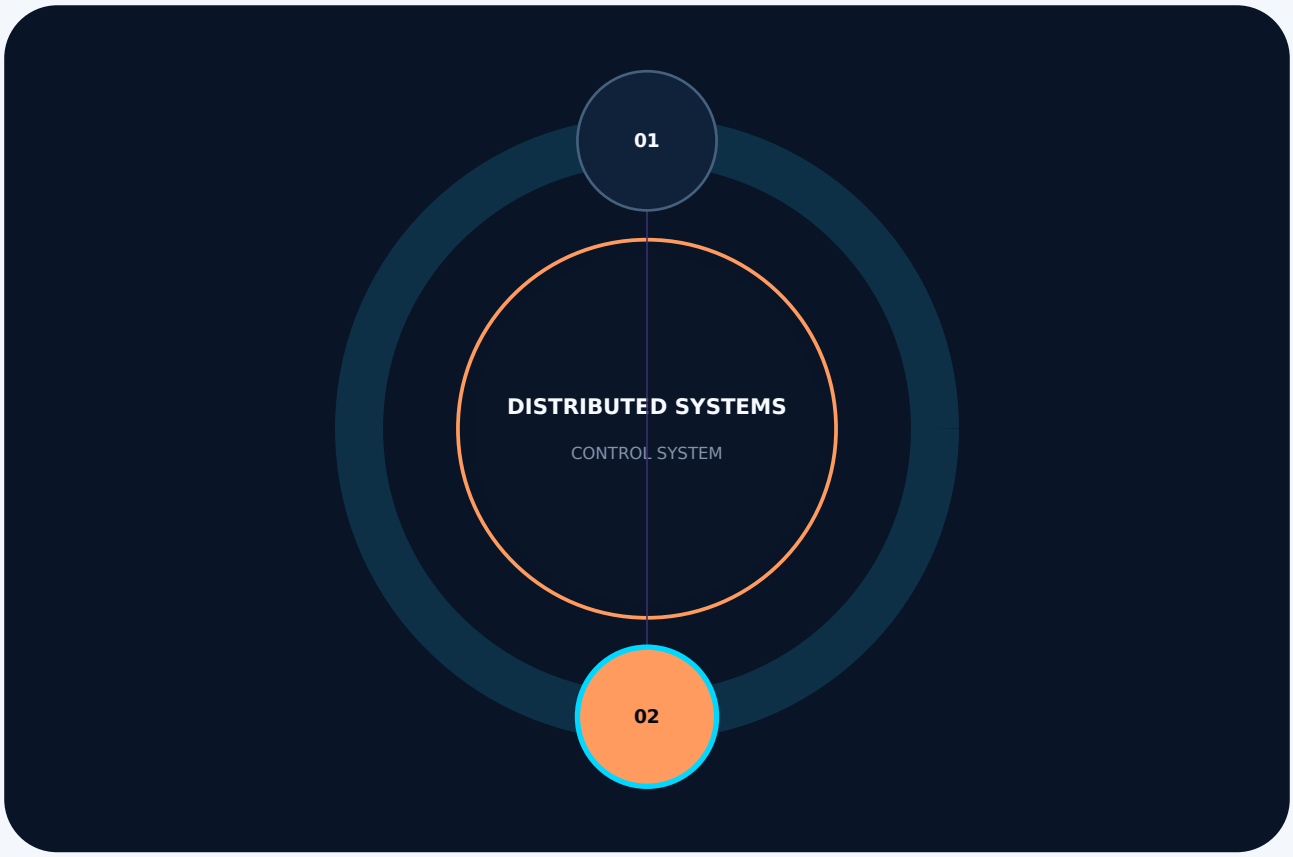
SCHEDULED REVIEW
2027-01-24

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

Publication doctrine. This publication establishes durable guidance or requirements for its defined scope. Interpretation must preserve the permanent document identity, recorded evidence, controlled exceptions, and formal revision history.

DOMAIN CONTROL SYSTEM

MYTHOS-DST-0002 inside the distributed systems standard constellation



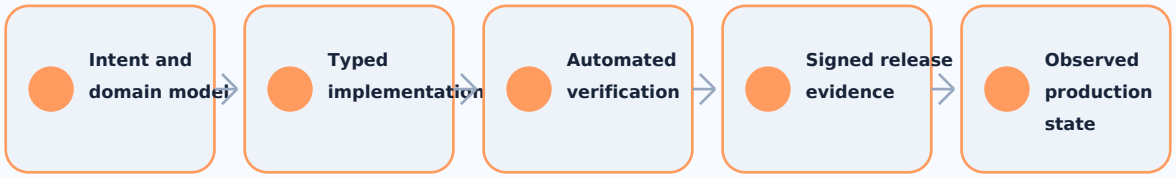
Connected assurance	Architecture, implementation, operations, evidence, and lifecycle are evaluated as one controlled system.
Specialization rule	Companion standards may add precision, but they cannot silently weaken this publication's requirements.

EXECUTIVE POSITION

API Contracts, gRPC, and Service Mesh Standard

The architecture of internal service-to-service communication has failed.

Evidence-bearing delivery path



Assurance maturity spectrum

CONTENTS

Publication navigation

MYTHOS-DST-0002 inside the distributed systems standard constellation . . . 3

API Contracts, gRPC, and Service Mesh Standard . . . 4

Assurance model and evaluation surface . . . 7

Criteria and evidence map . . . 8

Requirements, evaluation methodology, and implementation guidance . . . 9

Executive Mandate . . . 9

Scope and Applicability . . . 10

In Scope . . . 10

Out of Scope . . . 10

Audience . . . 10

Definitions and Taxonomy . . . 10

Mesh & Contract Dimensions . . . 10

The API & Mesh Doctrine . . . 11

Evaluation Methodology . . . 11

Scoring Model . . . 11

Weighting . . . 11

Evaluation Categories . . . 12

Schema-First and Protocol Buffers (Weight: 20%) . . . 12

Contract Integrity . . . 12

Service Mesh Decoupling (Weight: 20%) . . . 12

Network Abstraction . . . 12

Zero-Trust mTLS and Identity (Weight: 20%) . . . 12

East-West Encryption . . . 12

L7 Authorization and Policy (Weight: 15%) . . . 13

Access Control . . . 13

Declarative Traffic Management (Weight: 15%) . . . 13

Resiliency Control . . . 13

Observability and Tracing (Weight: 10%) . . . 13

Telemetry Pipeline . . . 13

The Mythos API & Mesh Suite (MAMS) 14

Suite Composition 14

MAMS-Contract 14

MAMS-Mesh 14

Execution Protocol 14

Implementation evidence and review gates 15

Current authority register 16

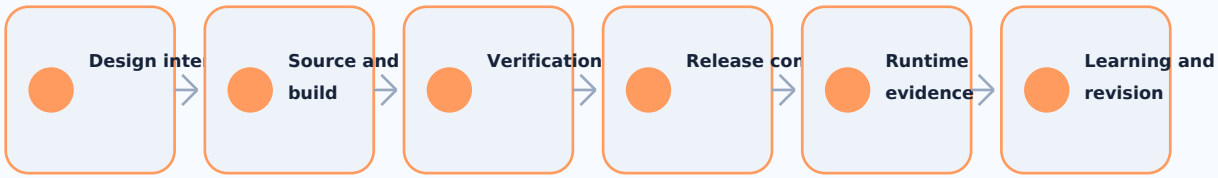
CONTROL TOPOLOGY

Assurance model and evaluation surface

The standard is assessed as an evidence-bearing system. Capability scores do not replace mandatory gates, implementation proof, or operational validation.

Capability family	Evaluation nodes
Schema-First and Protocol Buffers (Weight: 20%)	1
Service Mesh Decoupling (Weight: 20%)	1
Zero-Trust mTLS and Identity (Weight: 20%)	1
L7 Authorization and Policy (Weight: 15%)	1
Declarative Traffic Management (Weight: 15%)	1
Observability and Tracing (Weight: 10%)	1
Suite Composition	2
Additional Evaluation Dimension	4

Lifecycle control loop



EVALUATION INDEX

Criteria and evidence map

#	Capability family	Criterion
1	Schema-First and Protocol Buffers (Weight: 20%)	Contract Integrity
2	Service Mesh Decoupling (Weight: 20%)	Network Abstraction
3	Zero-Trust mTLS and Identity (Weight: 20%)	East-West Encryption
4	L7 Authorization and Policy (Weight: 15%)	Access Control
5	Declarative Traffic Management (Weight: 15%)	Resiliency Control
6	Observability and Tracing (Weight: 10%)	Telemetry Pipeline
7	Suite Composition	MAMS-Contract
8	Suite Composition	MAMS-Mesh
9	Additional Evaluation Dimension	Mesh & Contract Dimensions
10	Additional Evaluation Dimension	The API & Mesh Doctrine
11	Additional Evaluation Dimension	Scoring Model
12	Additional Evaluation Dimension	Suite Composition

NORMATIVE PUBLICATION

Requirements, evaluation methodology, and implementation guidance

The following body preserves the substantive source standard while normalizing publication status, identity, navigation, and visual presentation.

Executive Mandate

The architecture of internal service-to-service communication has failed.

Organizations build distributed microservices and allow developers to communicate between them using REST APIs and JSON. Because JSON is stringly-typed and lacks a strict schema, contracts break at runtime when a developer adds a required field or changes a data type. To fix the cascading failures, developers embed resiliency logic—retries, timeouts, and circuit breakers—directly into the application code. When the network changes, the code must be rewritten and redeployed. Furthermore, they assume the internal network is trusted, passing plaintext traffic and unauthenticated requests between services.

This standard exists because:

1. JSON/REST is for the Edge, Not the Mesh: JSON requires runtime parsing, bloated payloads, and manual validation. It guarantees contract drift. Internal communication MUST utilize schema-first, binary protocols like gRPC and Protocol Buffers (Protobuf), providing compile-time type safety and strict backward/forward compatibility.
2. Application Code is Not Network Code: Embedding retry logic, circuit breakers, and mTLS handshakes into business logic couples the domain to the network. Systems MUST utilize a Service Mesh (Istio, Linkerd, Cilium) to offload all traffic management, observability, and security to an out-of-process sidecar or node-level proxy.
3. The Internal Network is Hostile: A breach of a single frontend service instantly grants an attacker plaintext access to all downstream databases and internal APIs. Systems MUST enforce Zero-Trust east-west traffic, requiring mTLS with cryptographically verifiable workload identities (SPIFFE) for every request.
4. Resiliency Must be Declarative: Circuit breakers and retry policies hardcoded in application libraries are unmanageable at scale. Systems MUST declaratively configure traffic policies (e.g., "retry 3 times with 50ms backoff") via the Service Mesh control plane, allowing operations to tune resiliency without code deployments.

This document establishes the Mythos API & Mesh Standard (MAMS), a comprehensive, evidence-based methodology for evaluating internal communication architectures against contract integrity, zero-trust isolation, and network decoupling.

Scope and Applicability

In Scope

This standard covers the evaluation of:

1. API Contract frameworks (gRPC, Protocol Buffers, OpenAPI)
1. Schema evolution, versioning, and backward/forward compatibility
1. Service Mesh control planes and data planes (Istio, Linkerd, Envoy, Cilium)
1. Zero-trust east-west traffic (mTLS, SPIFFE workload identity)
1. L7 traffic management (circuit breakers, retries, timeouts, traffic shifting)
1. Declarative policy enforcement (OPA/Envoy integration)

Out of Scope

1. External API Gateway security (covered in MYTHOS-SEC-0011)
1. Asynchronous event-driven messaging (covered in MYTHOS-DST-0001)
1. General identity and PKI infrastructure (covered in MYTHOS-ID-0001/0002)

Audience

1. Distributed systems engineers and microservice architects
1. Platform engineers managing Kubernetes and Service Mesh infrastructures
1. DevSecOps teams enforcing zero-trust internal networking
1. SREs managing traffic shifting and incident response
1. Standards bodies seeking a reference API and mesh methodology

Definitions and Taxonomy

Mesh & Contract Dimensions

Dimension	Definition
Schema-First Design	An architectural paradigm where the API contract (e.g., `.proto` file) is defined, version-controlled, and peer-reviewed *before* any application code is written.
Protocol Buffers (Protobuf)	A language-neutral, platform-neutral binary serialization format developed by Google, offering strict typing and compact payloads compared to JSON.
Service Mesh	An infrastructure layer that facilitates service-to-service communication between microservices, typically via a sidecar proxy (e.g., Envoy), handling routing, security, and observability.
mTLS (Mutual TLS)	A mutual authentication process where both the client and the server present and verify cryptographic certificates, ensuring both parties are who they claim to be.
L7 Authorization	Access control enforced at the application layer (Layer 7), allowing policies based on HTTP methods, gRPC methods, or headers (e.g., "Service A can call `CreateUser` but not `DeleteUser`").

The API & Mesh Doctrine

> A JSON payload is a runtime crash waiting to happen. A microservice managing its own mTLS is a coupled monolith. An unauthenticated internal call is a lateral movement vector. If the circuit breaker is in the code, the network cannot adapt.

Evaluation Methodology

Scoring Model

Each capability is scored from 0 to 5.

Score	Meaning
0	Fails basic task; JSON/REST, plaintext internal, hardcoded resiliency
1	Basic gRPC used, but lacks a mesh or mTLS enforcement
2	Functional service mesh, but lacks L7 authorization or strict schema rules
3	Solid production performance; Protobuf/gRPC, Istio/Linkerd mesh, mTLS everywhere, OPA L7 authz
4	Strong performance; SPIFFE workload identity, formally verified schema evolution, zero-trust east-west
5	Best-in-class; sets the standard for mathematically proven, autonomous internal networking

Weighting

Category	Weight	Rationale
Schema-First & Protocol Buffers	20%	JSON causes contract drift and runtime crashes
Service Mesh Decoupling	20%	Application code must not handle network routing or mTLS
Zero-Trust mTLS & Identity	20%	Plaintext internal traffic guarantees lateral movement
L7 Authorization & Policy	15%	Network-level ACLs are too broad for microservices
Declarative Traffic Management	15%	Hardcoded retries/circuit breakers cannot adapt to outages
Observability & Tracing	10%	A mesh without distributed tracing is a black box

Total: 100%

A system MUST score at least 3.0 in Schema-First and Zero-Trust mTLS to be considered for production use.

Evaluation Categories

Schema-First and Protocol Buffers (Weight: 20%)

Contract Integrity

Are internal APIs strongly typed and resilient to breaking changes?

Score	Criteria
0	Developers define JSON payloads in code; no central schema
1	OpenAPI/Swagger documentation exists, but is generated after the fact
2	Protobuf used, but schema reviews are not enforced
3	Strict schema-first design: `.proto` files are the source of truth; CI/CD blocks breaking changes (e.g., changing field types, deleting required fields)
4	Automated backward/forward compatibility testing; deprecated fields managed via `reserved` keywords
5	Perfect integrity: formally verified schema evolution, zero runtime contract violations, mathematical proof of backward compatibility

Service Mesh Decoupling (Weight: 20%)

Network Abstraction

Is traffic management, routing, and security handled out-of-process?

Score	Criteria
0	Application code handles HTTP routing, retries, and TLS
1	Internal API Gateway used, but services still manage their own TLS
2	Sidecar proxies (Envoy) injected, but require manual configuration
3	Full Service Mesh (Istio/Linkerd/Cilium) declaratively managing all east-west traffic, mTLS, and retries
4	Heterogeneous workloads (VMs, Kubernetes, Edge) unified under a single mesh control plane
5	Perfect decoupling: formally verified network abstraction, zero application-level network code, mathematical proof of sidecar isolation

Zero-Trust mTLS and Identity (Weight: 20%)

East-West Encryption

Is all internal traffic encrypted and mutually authenticated?

Score	Criteria
0	Plaintext HTTP inside the VPC; IP-based trust
1	TLS used, but with long-lived, manually rotated certificates
2	mTLS enforced, but certificates are tied to static service accounts
3	SPIFFE/SPIRE workload identity; short-lived SVIDs automatically rotated by the mesh

Score	Criteria
4	Zero-trust: services refuse to communicate with any peer lacking a valid SVID, even within the same pod/node
5	Perfect zero-trust: formally verified cryptographic identity, zero plaintext internal traffic, mathematical proof of peer authentication

L7 Authorization and Policy (Weight: 15%)

Access Control

Are internal API calls authorized at the method level?

Score	Criteria
0	Any service can call any method on any other service
1	Network policies (L3/L4) restrict IP communication between namespaces
2	L7 policies exist, but hardcoded in application code
3	Declarative L7 AuthorizationPolicies (Istio) or OPA sidecars enforcing method-level access (e.g., `Service A` can call `GET /users` but not `DELETE /users`)
4	Context-aware policies: access decisions based on request headers, JWT claims, or SPIFFE IDs
5	Perfect authorization: formally verified policy bounds, zero unauthorized method calls possible, mathematical proof of least-privilege east-west traffic

Declarative Traffic Management (Weight: 15%)

Resiliency Control

How are retries, timeouts, and circuit breakers managed?

Score	Criteria
0	Hardcoded in application libraries (e.g., RetryingHttpClient)
1	Configurable via application config files, but requires restart
2	Managed by the mesh, but requires manual YAML updates for tuning
3	Fully declarative: mesh control plane dynamically applies circuit breakers, timeouts, and retry policies without application restarts
4	Automated traffic shifting (e.g., canary deployments, fault injection) managed declaratively
5	Perfect control: formally verified resiliency bounds, zero cascading failures, mathematical optimization of retry backoff strategies

Observability and Tracing (Weight: 10%)

Telemetry Pipeline

Does the mesh provide automatic, out-of-the-box distributed tracing?

Score	Criteria
0	Developers must manually instrument tracing headers in code

Score	Criteria
1	Mesh provides basic L4 metrics (bytes sent/received)
2	Mesh provides L7 metrics and basic distributed tracing (requires code instrumentation for context propagation)
3	Mesh automatically injects/propagates trace context (W3C Trace Context); zero-code distributed tracing
4	Real-time service topology maps and latency percentiles (p99) generated from mesh telemetry
5	Perfect observability: formally verified trace completeness, zero blind spots in east-west traffic, cryptographic attestation of telemetry integrity

The Mythos API & Mesh Suite (MAMS)

Traditional network benchmarks measure TCP throughput. The MAMS evaluates schema breakage, mTLS enforcement, and circuit breaker response.

Suite Composition

MAMS-Contract

1. Breaking Change Test: 100+ tests attempting to merge a PR that alters a Protobuf field type without a deprecated wrapper, verifying the CI/CD pipeline blocks it.
1. Runtime Fuzz Test: 50+ tests sending malformed binary payloads to a gRPC endpoint, verifying the Protobuf parser rejects them without crashing the service.

MAMS-Mesh

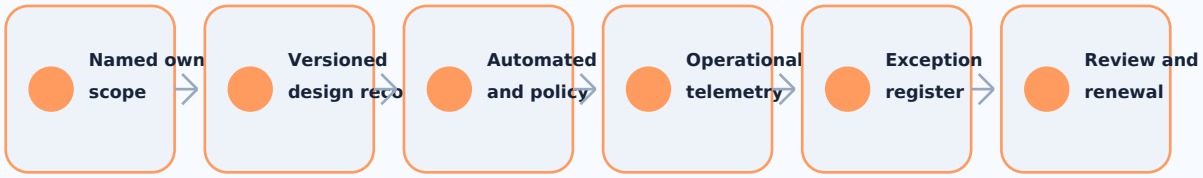
1. Plaintext Injection: 100+ tests attempting to bypass the sidecar and call a downstream service via plaintext HTTP, verifying the service refuses the connection.
1. Authz Bypass Test: 50+ tests attempting to call a restricted gRPC method (`DeleteUser`) from an unauthorized service, verifying the OPA/Envoy proxy blocks the call.

Execution Protocol

ASSURANCE AND ADOPTION

Implementation evidence and review gates

Required assurance chain



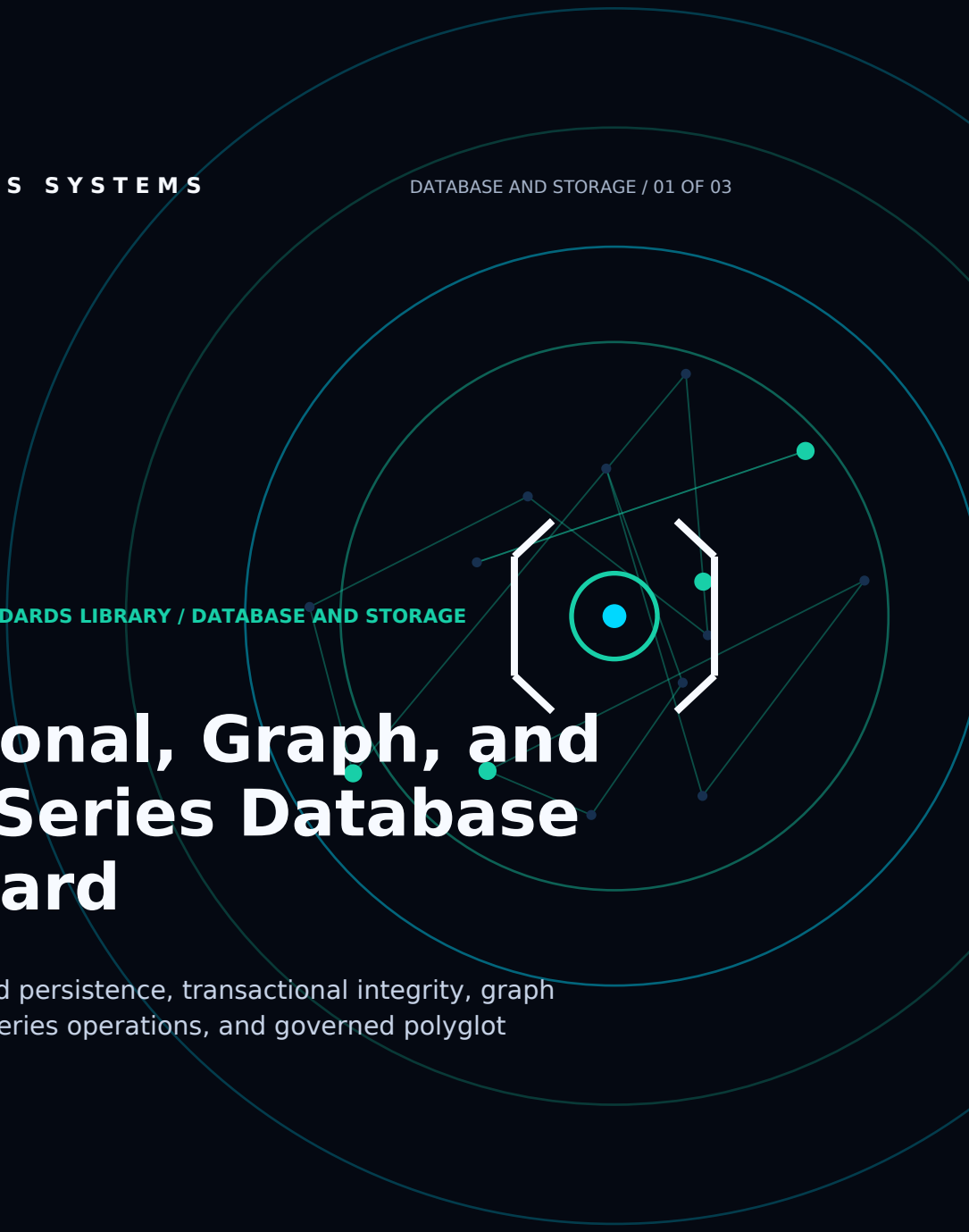
A passing score is not sufficient without current evidence. High-assurance adoption requires traceable design decisions, reproducible builds, independently reviewable tests, controlled secrets, runtime observability, failure exercises, and a dated exception register.

Assurance maturity spectrum

Current authority register

Authority	Version / snapshot	Applicability	Review by
TLA+ Foundation: TLA+ Language and Tools Documentation	Rolling documentation snapshot	High	2027-01-24
OpenAPI Initiative: OpenAPI Specification	3.2.0	High	2027-01-24
AsyncAPI Initiative: AsyncAPI Specification	3.1.0	High	2027-01-24
Kubernetes Project: Kubernetes Documentation	v1.36 current documentation	High	2026-10-22
OpenTelemetry Project: OpenTelemetry Documentation and Specification	Rolling specification snapshot; profiles public alpha and declarative configuration stable in 2026	High	2026-10-22
Cloud Native Computing Foundation: Cloud Native Landscape and Project Documentation	Rolling catalog snapshot	Moderate	2026-08-23

Research limitation. Official documentation establishes published interfaces and declared behavior. It does not prove the performance, security, interoperability, cost, or operational suitability of a specific implementation. Those claims require controlled testing and retained evidence.



ENGINEERING STANDARDS LIBRARY / DATABASE AND STORAGE

Relational, Graph, and Time-Series Database Standard

Workload-aligned persistence, transactional integrity, graph traversal, time-series operations, and governed polyglot persistence.

PERMANENT DOCUMENT ID

MYTHOS-DBS-0001

PUBLICATION

Candidate Standard
Version 1.0.0-candidate

ASSURANCE PROFILE

Standard / Advanced
High Assurance

PERMANENT PUBLICATION IDENTITY

Relational, Graph, and Time-Series Database Standard

DOCUMENT ID
MYTHOS-DBS-0001

VERSION
1.0.0-candidate

STATUS
Candidate Standard

PUBLISHER
Mythos Systems

PUBLICATION DATE
2026-07-24

SCHEDULED REVIEW
2027-01-24

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

Publication doctrine. This publication establishes durable guidance or requirements for its defined scope. Interpretation must preserve the permanent document identity, recorded evidence, controlled exceptions, and formal revision history.

DOMAIN CONTROL SYSTEM

MYTHOS-DBS-0001 inside the database and storage standard constellation

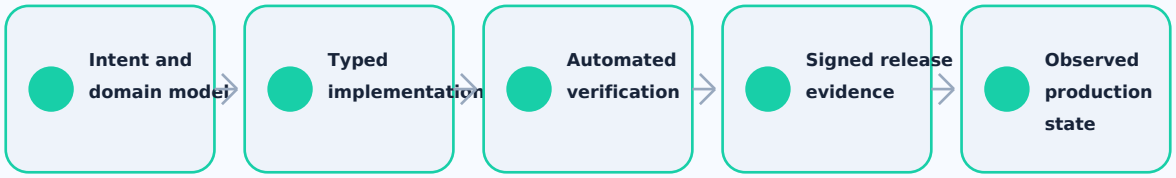
Connected assurance	Architecture, implementation, operations, evidence, and lifecycle are evaluated as one controlled system.
Specialization rule	Companion standards may add precision, but they cannot silently weaken this publication's requirements.

EXECUTIVE POSITION

Relational, Graph, and Time-Series Database Standard

The architecture of data persistence has failed.

Evidence-bearing delivery path



Assurance maturity spectrum

0	1	2	3	4	5
Absent	Ad hoc	Repeatable	Controlled	Measured	Adaptive

CONTENTS

Publication navigation

MYTHOS-DBS-0001 inside the database and storage standard constellation . . 3

Relational, Graph, and Time-Series Database Standard 4

Assurance model and evaluation surface 7

Criteria and evidence map 8

Requirements, evaluation methodology, and implementation guidance 9

Executive Mandate 9

Scope and Applicability 9

In Scope 9

Out of Scope 10

Audience 10

Definitions and Taxonomy 10

Persistence Dimensions 10

The Database Doctrine 10

Evaluation Methodology 11

Scoring Model 11

Weighting 11

Evaluation Categories 11

Polyglot Data Modeling (Weight: 20%) 11

Data Shape Alignment 11

Connection Pooling and Proxying (Weight: 20%) 12

Resource Management 12

High Availability and Failover (Weight: 15%) 12

Resilience 12

Scalability and Sharding (Weight: 15%) 12

Write Throughput 12

Tenant Isolation (RLS) (Weight: 15%) 13

Security Boundaries 13

Backup and Point-in-Time Recovery (Weight: 15%) 13

Data Durability 13

The Mythos Database Suite (MDBS) 14

Suite Composition 14

MDBS-Pool 14

MDBS-Isolation 14

Execution Protocol 14

Implementation evidence and review gates 15

Current authority register 15

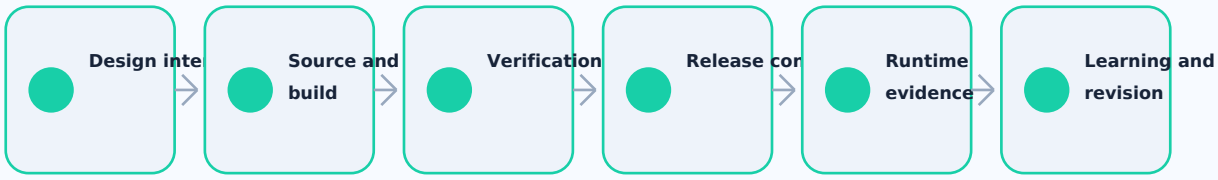
CONTROL TOPOLOGY

Assurance model and evaluation surface

The standard is assessed as an evidence-bearing system. Capability scores do not replace mandatory gates, implementation proof, or operational validation.

Capability family	Evaluation nodes
Polyglot Data Modeling (Weight: 20%)	1
Connection Pooling and Proxying (Weight: 20%)	1
High Availability and Failover (Weight: 15%)	1
Scalability and Sharding (Weight: 15%)	1
Tenant Isolation (RLS) (Weight: 15%)	1
Backup and Point-in-Time Recovery (Weight: 15%)	1
Suite Composition	2
Additional Evaluation Dimension	4

Lifecycle control loop



EVALUATION INDEX

Criteria and evidence map

#	Capability family	Criterion
1	Polyglot Data Modeling (Weight: 20%)	Data Shape Alignment
2	Connection Pooling and Proxying (Weight: 20%)	Resource Management
3	High Availability and Failover (Weight: 15%)	Resilience
4	Scalability and Sharding (Weight: 15%)	Write Throughput
5	Tenant Isolation (RLS) (Weight: 15%)	Security Boundaries
6	Backup and Point-in-Time Recovery (Weight: 15%)	Data Durability
7	Suite Composition	MDBS-Pool
8	Suite Composition	MDBS-Isolation
9	Additional Evaluation Dimension	Persistence Dimensions
10	Additional Evaluation Dimension	The Database Doctrine
11	Additional Evaluation Dimension	Scoring Model
12	Additional Evaluation Dimension	Suite Composition

NORMATIVE PUBLICATION

Requirements, evaluation methodology, and implementation guidance

The following body preserves the substantive source standard while normalizing publication status, identity, navigation, and visual presentation.

Executive Mandate

The architecture of data persistence has failed.

Organizations force highly connected social graph data into relational tables, attempting deep traversals via recursive JOINs that cripple the CPU. They push high-cardinality IoT telemetry into standard relational databases, watching the index bloat until the system grinds to a halt. They allow microservices to open thousands of direct database connections, causing "connection storms" that exhaust database memory and crash the cluster. Furthermore, they rely on single-node databases with manual backup scripts, accepting hours of data loss (RPO) when a disk fails.

This standard exists because:

1. One Database Cannot Serve All Data Shapes: A relational database is optimized for set-based operations, not deep relationship traversals. A time-series database is optimized for high-write throughput and time-bucketed aggregations, not ACID transactions. Systems MUST adopt polyglot persistence, matching the data shape to the database engine.
2. Direct Connections are a Memory DoS: A microservice cluster spinning up 100 pods with 10 connection pools each will attempt to open 1,000 direct connections to the database. The database will spend 80% of its CPU managing connection state instead of executing queries. Systems MUST implement connection pooling proxies (e.g., PgBouncer, ProxySQL) to multiplex thousands of application connections into tens of database connections.
3. Single-Node Scaling is a Dead End: A monolithic database that scales vertically (bigger AWS instance) hits a hard hardware ceiling. Systems MUST implement horizontal scalability via sharding or distributed SQL (e.g., Vitess, Citus, CockroachDB) for write-heavy workloads.
4. Tenant Isolation Must Be Enforced by the Database: Trusting the application layer to filter data by `tenant_id` guarantees a cross-tenant data breach when a developer forgets a `WHERE` clause. Systems MUST enforce Row-Level Security (RLS) at the database engine level.

This document establishes the Mythos Database Standard (MDBS), a comprehensive, evidence-based methodology for evaluating database architectures against polyglot modeling, connection hygiene, and horizontal resilience.

Scope and Applicability

In Scope

This standard covers the evaluation of:

- Polyglot Persistence: Relational (PostgreSQL), Graph (Neo4j), Time-Series (InfluxDB)
- Connection pooling and proxy architectures (PgBouncer, Odyssey, ProxySQL)
- Horizontal scalability: Sharding, distributed SQL, and read replicas
- High Availability (HA): Synchronous replication, automated failover (Patroni)
- Tenant isolation: Row-Level Security (RLS) and database-level multi-tenancy
- Backup and recovery: Point-in-Time Recovery (PITR) and immutable backups

Out of Scope

- Vector databases and local-first sync (covered in MYTHOS-DAT-0001)
- Event sourcing and CQRS patterns (covered in MYTHOS-DAT-0004)
- General API and service mesh security (covered in MYTHOS-DST-0002)

Audience

- Database administrators (DBAs) and data architects
- Platform engineers managing stateful Kubernetes workloads
- Backend engineers designing microservice data stores
- Security engineers evaluating tenant isolation and data exfiltration risks
- Standards bodies seeking a reference database methodology

Definitions and Taxonomy

Persistence Dimensions

Dimension	Definition
Polyglot Persistence	The architectural paradigm of using different database technologies to handle different data storage needs within a single system (e.g., Postgres for transactions, Neo4j for relationships, InfluxDB for metrics).
Connection Multiplexing	The use of a proxy (e.g., PgBouncer) to maintain a small pool of active database connections, routing thousands of lightweight client requests over them to prevent database memory exhaustion.
Sharding	The process of horizontally partitioning large datasets into smaller, distributed blocks (shards) across multiple database nodes to parallelize write and read throughput.
Row-Level Security (RLS)	A database feature that restricts which rows a user can access based on a policy defined within the database itself, rather than relying on application logic.
Point-in-Time Recovery (PITR)	A backup strategy that restores a database to its state at a specific moment in time using continuous write-ahead log (WAL) archiving.

The Database Doctrine

> A recursive JOIN is not a graph traversal. An unbounded connection pool is a self-inflicted DoS. An application-level `tenant_id` filter is a breach waiting to happen. If the database cannot scale horizontally, the application cannot scale at all.

Evaluation Methodology

Scoring Model

Each capability is scored from 0 to 5.

Score	Meaning
0	Fails basic task; monolithic DB, direct connections, single-node, no RLS
1	Basic read replicas, but relies on vertical scaling and app-level tenancy
2	Functional polyglot setup, but lacks connection pooling or sharding
3	Solid production performance; Polyglot, PgBouncer, automated HA/replicas, RLS enforced
4	Strong performance; Distributed SQL/Sharding, PITR backups, formally verified RLS
5	Best-in-class; sets the standard for mathematically verified, zero-downtime data persistence

Weighting

Category	Weight	Rationale
Polyglot Data Modeling	20%	Forcing all data into SQL causes catastrophic performance failures
Connection Pooling & Proxying	20%	Direct connections from microservices exhaust database memory
High Availability & Failover	15%	Single-node databases are a critical single point of failure
Scalability & Sharding	15%	Vertical scaling hits hardware limits; horizontal is required
Tenant Isolation (RLS)	15%	App-level filtering guarantees cross-tenant data leaks
Backup & Point-in-Time Recovery	15%	Snapshot backups guarantee data loss between backups

Total: 100%

A system MUST score at least 3.0 in Connection Pooling and High Availability to be considered for production use.

Evaluation Categories

Polyglot Data Modeling (Weight: 20%)

Data Shape Alignment

Is the database engine matched to the data structure and access pattern?

Score	Criteria
0	All data (transactions, relationships, telemetry) forced into a single MySQL/Postgres instance
1	Graph data stored in relational tables with heavy JOINs; metrics stored in SQL

Score	Criteria
2	Time-series data offloaded to InfluxDB, but graphs remain in SQL
3	Full polyglot persistence: PostgreSQL (ACID), Neo4j (Graph), InfluxDB (Telemetry)
4	Event-sourced aggregates (CQRS) utilized to pre-compute complex reads
5	Perfect modeling: formally verified data routing, zero performance-degrading access patterns, mathematical proof of optimal engine selection

Connection Pooling and Proxying (Weight: 20%)

Resource Management

How do microservices connect to the database without exhausting resources?

Score	Criteria
0	Microservices open direct TCP connections to the database; connection limits frequently hit
1	Application-level connection pooling (e.g., HikariCP), but 100 pods still open 1000 connections
2	Connection proxy (PgBouncer/ProxySQL) deployed, but configured with static, unoptimized limits
3	Transaction-level pooling via PgBouncer; database connection count strictly bounded (< 100) regardless of app scale
4	Dynamic pool sizing based on real-time database load and CPU metrics
5	Perfect pooling: formally verified connection bounds, zero connection storms possible, mathematical proof of optimal multiplexing

High Availability and Failover (Weight: 15%)

Resilience

Can the database survive the loss of its primary node without data loss or downtime?

Score	Criteria
0	Single-node database; manual recovery required on crash
1	Asynchronous replication to a standby node; manual promotion required
2	Automated failover via Patroni/Stolon, but accepts some data loss (async)
3	Synchronous replication ensuring zero data loss (RPO=0); automated, sub-minute failover (RTO < 60s)
4	Multi-region active-active or distributed SQL (CockroachDB) surviving a region loss
5	Perfect resilience: formally verified consensus algorithms, zero split-brain risk, mathematical proof of continuous availability

Scalability and Sharding (Weight: 15%)

Write Throughput

How does the database handle write loads that exceed a single machine's capacity?

Score	Criteria
0	Vertical scaling only (buying a bigger EC2 instance)
1	Read replicas used to offload reads, but writes bottlenecked on primary
2	Manual application-level sharding (hard to maintain)
3	Distributed SQL (CockroachDB) or transparent sharding (Citus/Vitess) handling writes horizontally
4	Dynamic resharding without downtime; hotspots automatically rebalanced
5	Perfect scaling: formally verified data distribution, zero write bottlenecks, linear horizontal scalability

Tenant Isolation (RLS) (Weight: 15%)

Security Boundaries

How is cross-tenant data access prevented?

Score	Criteria
0	Application code filters by `tenant_id` (guarantees eventual breach)
1	Database views used to filter data, but views can be bypassed
2	Database roles created per tenant, but application manages switching roles
3	Row-Level Security (RLS) enforced at the database engine level; application cannot bypass policies even with raw SQL
4	Context-aware RLS: database reads session variables (e.g., `app.current_tenant`) set by the connection proxy
5	Perfect isolation: formally verified RLS policies, zero ability to query cross-tenant data, cryptographic proof of tenant boundaries

Backup and Point-in-Time Recovery (Weight: 15%)

Data Durability

Can the database be restored to any specific second in the past?

Score	Criteria
0	Nightly snapshot backups only; up to 24 hours of data loss possible
1	WAL archiving enabled, but restoration is manual and untested
2	Point-in-Time Recovery (PITR) automated and tested quarterly
3	Continuous WAL archiving to immutable, write-once storage (S3 Object Lock)
4	Automated, daily restoration tests in an isolated environment verifying backup integrity
5	Perfect durability: formally verified recovery bounds, zero data loss (RPO=0), cryptographic attestation of backup integrity

The Mythos Database Suite (MDBS)

Traditional benchmarks measure TPS (Transactions Per Second). The MDBS evaluates connection storm survival, tenant isolation, and zero-data-loss failover.

Suite Composition

MDBS-Pool

1. Connection Storm Test: 100+ tests spawning 5,000 concurrent microservice connections, verifying the PgBouncer proxy bounds the database connections to < 100 without dropping transactions.
1. Failover Test: 50+ tests physically killing the primary database node during peak write load, verifying Patroni promotes a standby with zero data loss (synchronous) and sub-minute recovery.

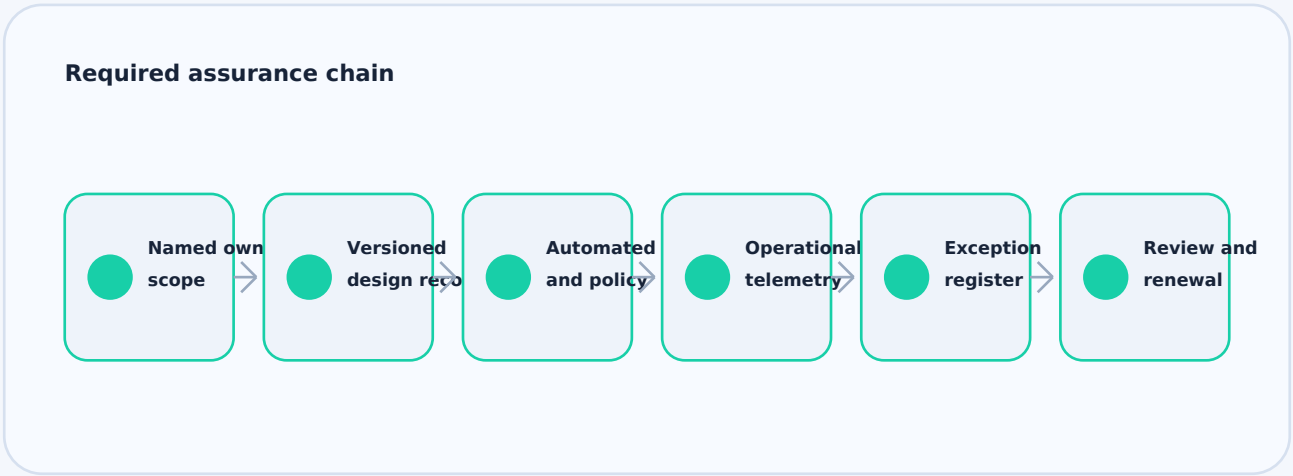
MDBS-Isolation

1. Tenant Bypass Test: 100+ tests attempting to execute raw SQL queries without tenant context or with a spoofed tenant ID, verifying the database RLS policies block the queries.
1. PITR Test: 50+ tests restoring the database to a random timestamp 3 hours in the past, verifying the WAL archive is continuous and the restoration is byte-perfect.

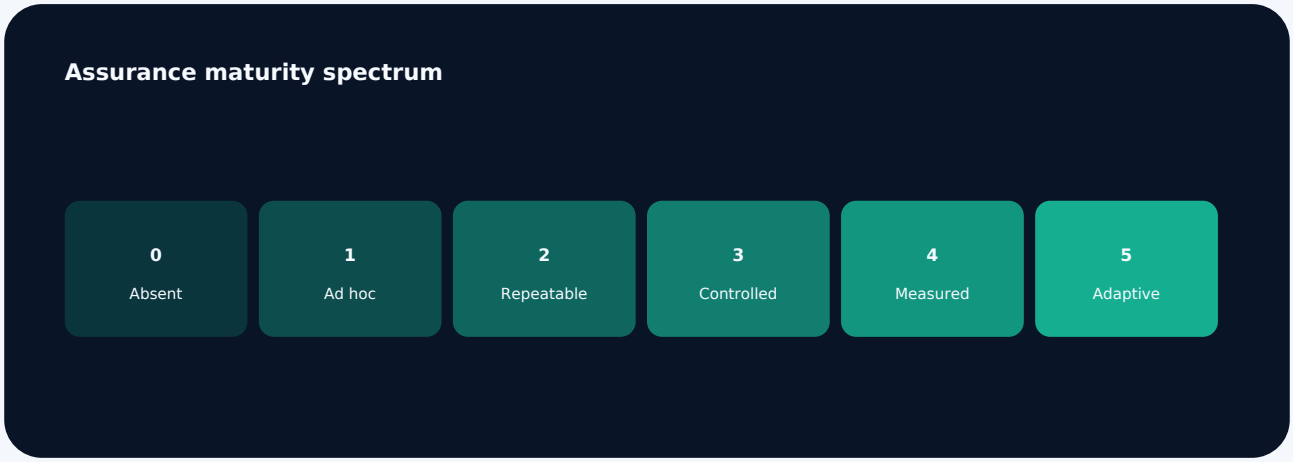
Execution Protocol

ASSURANCE AND ADOPTION

Implementation evidence and review gates



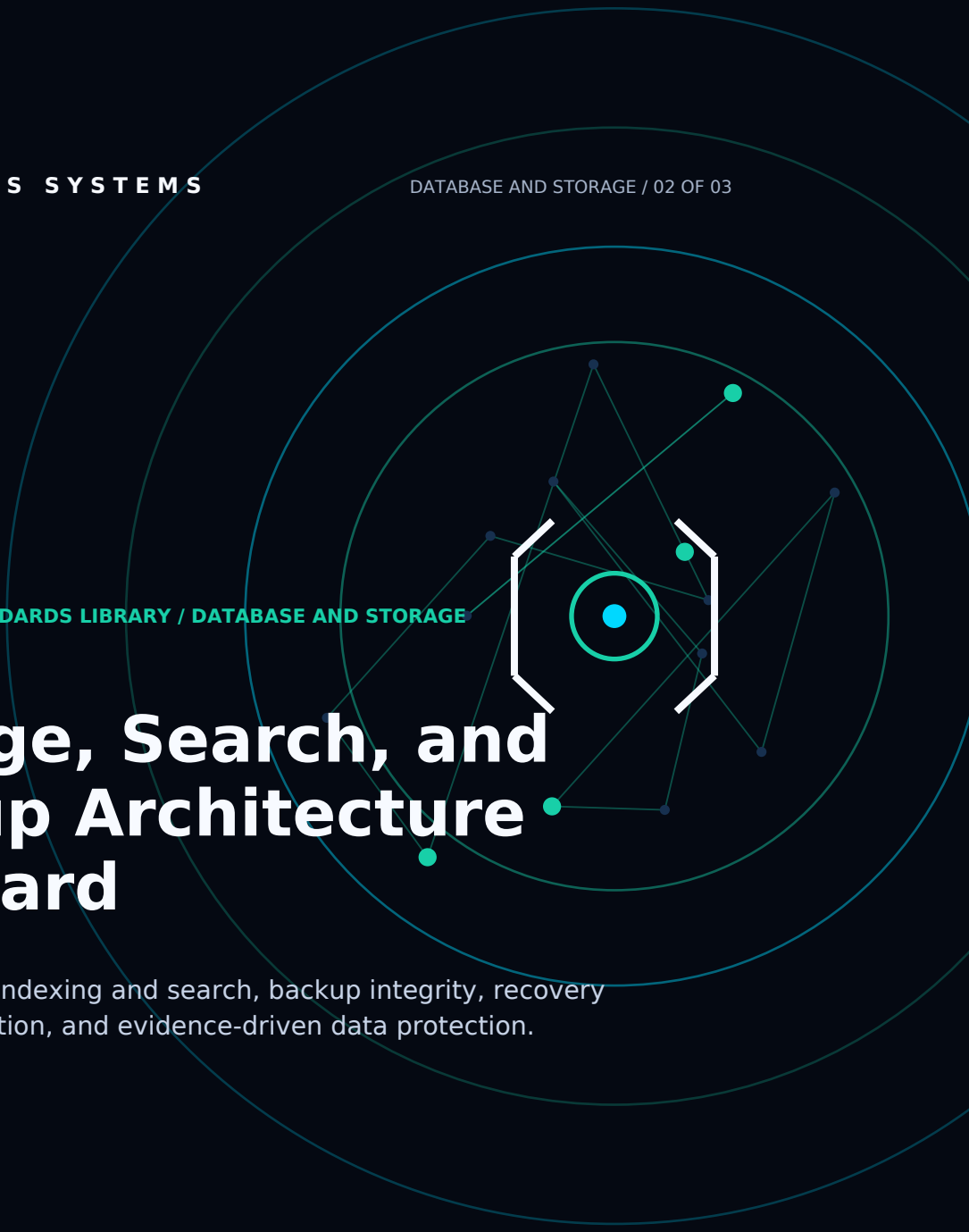
A passing score is not sufficient without current evidence. High-assurance adoption requires traceable design decisions, reproducible builds, independently reviewable tests, controlled secrets, runtime observability, failure exercises, and a dated exception register.



Current authority register

Authority	Version / snapshot	Applicability	Review by
OpenTelemetry Project: OpenTelemetry Documentation and Specification	Rolling specification snapshot; profiles public alpha and declarative configuration stable in 2026	High	2026-10-22
PostgreSQL Global Development Group: PostgreSQL Documentation	Rolling official documentation snapshot	High	2026-10-22

Research limitation. Official documentation establishes published interfaces and declared behavior. It does not prove the performance, security, interoperability, cost, or operational suitability of a specific implementation. Those claims require controlled testing and retained evidence.



ENGINEERING STANDARDS LIBRARY / DATABASE AND STORAGE

Storage, Search, and Backup Architecture Standard

Storage tiering, indexing and search, backup integrity, recovery validation, retention, and evidence-driven data protection.

PERMANENT DOCUMENT ID

MYTHOS-DBS-0002

PUBLICATION

Candidate Standard
Version 1.0.0-candidate

ASSURANCE PROFILE

Standard / Advanced
High Assurance

PERMANENT PUBLICATION IDENTITY

Storage, Search, and Backup Architecture Standard

DOCUMENT ID
MYTHOS-DBS-0002

VERSION
1.0.0-candidate

STATUS
Candidate Standard

PUBLISHER
Mythos Systems

PUBLICATION DATE
2026-07-24

SCHEDULED REVIEW
2027-01-24

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

Publication doctrine. This publication establishes durable guidance or requirements for its defined scope. Interpretation must preserve the permanent document identity, recorded evidence, controlled exceptions, and formal revision history.

DOMAIN CONTROL SYSTEM

MYTHOS-DBS-0002 inside the database and storage standard constellation

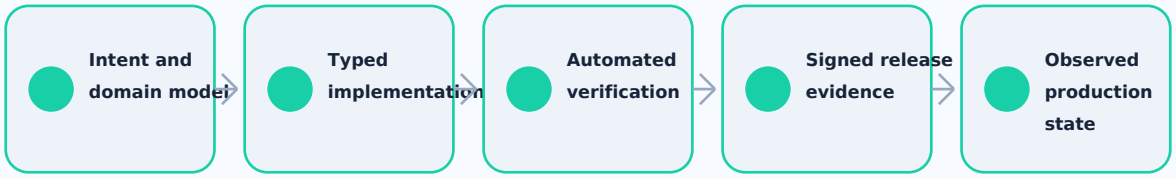
Connected assurance	Architecture, implementation, operations, evidence, and lifecycle are evaluated as one controlled system.
Specialization rule	Companion standards may add precision, but they cannot silently weaken this publication's requirements.

EXECUTIVE POSITION

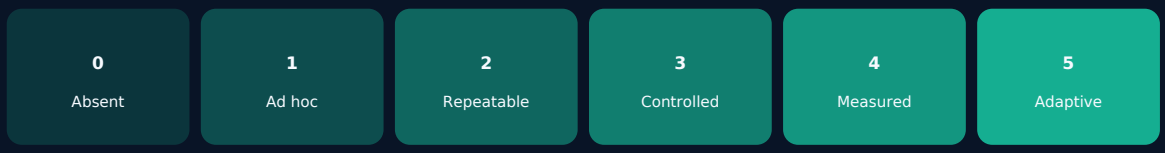
Storage, Search, and Backup Architecture Standard

The architecture of data storage, search, and backup has failed.

Evidence-bearing delivery path



Assurance maturity spectrum



CONTENTS

Publication navigation

MYTHOS-DBS-0002 inside the database and storage standard constellation . . 3

Storage, Search, and Backup Architecture Standard 4

Assurance model and evaluation surface 7

Criteria and evidence map 8

Requirements, evaluation methodology, and implementation guidance 9

Executive Mandate 9

Scope and Applicability 9

In Scope 9

Out of Scope 10

Audience 10

Definitions and Taxonomy 10

Storage & Search Dimensions 10

The Storage & Backup Doctrine 10

Evaluation Methodology 11

Scoring Model 11

Weighting 11

Evaluation Categories 11

Backup Immutability and Ransomware Resilience (Weight: 20%) 11

WORM Enforcement 11

Restore Testing and DR Validation (Weight: 20%) 12

Recovery Assurance 12

Search Architecture and CDC Decoupling (Weight: 15%) 12

Query Performance 12

Storage Tiering and Lifecycle Management (Weight: 15%) 13

Economic Optimization 13

Data Sanitization (Crypto-Shredding) (Weight: 15%) 13

Secure Deletion 13

High Availability and Quorum Protection (Weight: 15%) 13

Split-Brain Prevention 13

The Mythos Storage & Backup Suite (MSBS) 14

Suite Composition 14

MSBS-Ransomware 14

MSBS-Search 14

Execution Protocol 14

Implementation evidence and review gates 15

Current authority register 15

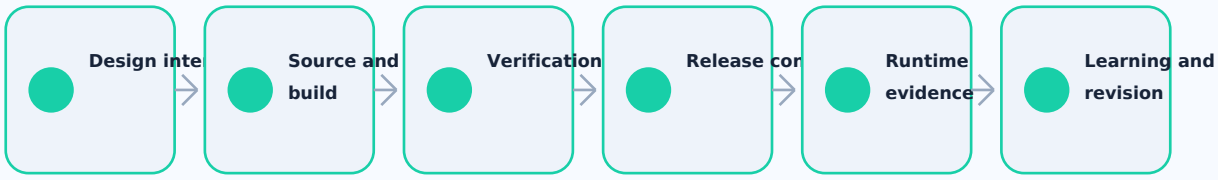
CONTROL TOPOLOGY

Assurance model and evaluation surface

The standard is assessed as an evidence-bearing system. Capability scores do not replace mandatory gates, implementation proof, or operational validation.

Capability family	Evaluation nodes
Backup Immutability and Ransomware Resilience (Weight: 20%)	1
Restore Testing and DR Validation (Weight: 20%)	1
Search Architecture and CDC Decoupling (Weight: 15%)	1
Storage Tiering and Lifecycle Management (Weight: 15%)	1
Data Sanitization (Crypto-Shredding) (Weight: 15%)	1
High Availability and Quorum Protection (Weight: 15%)	1
Suite Composition	2
Additional Evaluation Dimension	4

Lifecycle control loop



EVALUATION INDEX

Criteria and evidence map

#	Capability family	Criterion
1	Backup Immutability and Ransomware Resilience (Weight: 20%)	WORM Enforcement
2	Restore Testing and DR Validation (Weight: 20%)	Recovery Assurance
3	Search Architecture and CDC Decoupling (Weight: 15%)	Query Performance
4	Storage Tiering and Lifecycle Management (Weight: 15%)	Economic Optimization
5	Data Sanitization (Crypto-Shredding) (Weight: 15%)	Secure Deletion
6	High Availability and Quorum Protection (Weight: 15%)	Split-Brain Prevention
7	Suite Composition	MSBS-Ransomware
8	Suite Composition	MSBS-Search
9	Additional Evaluation Dimension	Storage & Search Dimensions
10	Additional Evaluation Dimension	The Storage & Backup Doctrine
11	Additional Evaluation Dimension	Scoring Model
12	Additional Evaluation Dimension	Suite Composition

NORMATIVE PUBLICATION

Requirements, evaluation methodology, and implementation guidance

The following body preserves the substantive source standard while normalizing publication status, identity, navigation, and visual presentation.

Executive Mandate

The architecture of data storage, search, and backup has failed.

Organizations treat backups as a nightly chore, writing snapshots to the same mutable cloud storage bucket as their production data. When ransomware hits, the backups are encrypted alongside the primary data. They attempt to implement full-text search by running complex `LIKE` queries against their relational database, crippling the CPU and destroying transactional performance. They store cold archival data on expensive hot block storage, wasting millions in capital.

This standard exists because:

1. A Mutable Backup is a Ransomware Target: A backup that can be overwritten or deleted by the same credentials that manage the production system provides zero resilience against ransomware. Backups MUST be written to Write-Once-Read-Many (WORM) immutable storage with Object Lock, completely air-gapped from production credentials.
2. A SQL `LIKE` Query is Not a Search Engine: Relational databases are optimized for set-based ACID transactions, not inverted index traversals. Forcing full-text search through SQL causes lock contention and catastrophic latency. Systems MUST decouple search into dedicated engines (OpenSearch/Elasticsearch) via Change Data Capture (CDC).
3. Untested Backups are Placebos: An untested backup is a hope, not a guarantee. Organizations discover their backups are corrupted only when a disaster occurs. Systems MUST automate daily restoration tests in isolated environments, mathematically verifying data integrity.
4. Storage Tiers Must Match Data Temperature: Storing 5-year-old compliance logs on NVMe SSDs is economic malpractice. Systems MUST implement automated data lifecycle policies, tiering cold data to cheap object storage (e.g., AWS Glacier) and utilizing crypto-shredding for secure deletion.

This document establishes the Mythos Storage & Backup Standard (MSBS), a comprehensive, evidence-based methodology for evaluating storage and search architectures against ransomware resilience, query performance, and data durability.

Scope and Applicability

In Scope

This standard covers the evaluation of:

1. Object storage architectures (S3, MinIO, Azure Blob) and WORM (Object Lock)
1. Block and file storage tiering (NVMe, HDD, archival tape/glacier)

1. Search engines and inverted indexes (OpenSearch, Elasticsearch, Solr)
1. Change Data Capture (CDC) for decoupled search indexing
1. Backup strategies (3-2-1-1-0 rule, immutable vaults, Velero/Veeam)
1. Disaster Recovery (DR) testing and automated restore validation
1. Data sanitization (Crypto-shredding, secure erasure)

Out of Scope

1. Relational and time-series database internal scaling (covered in MYTHOS-DBS-0001)
1. General data sovereignty and egress controls (covered in MYTHOS-DAT-0003)
1. General infrastructure-as-code for storage provisioning (covered in MYTHOS-PLT-0002)

Audience

1. Storage architects and backup engineers
1. Platform engineers managing stateful Kubernetes and object storage
1. Security engineers designing ransomware resilience
1. Data architects building search pipelines
1. Standards bodies seeking a reference storage and backup methodology

Definitions and Taxonomy

Storage & Search Dimensions

Dimension	Definition
WORM (Write-Once-Read-Many)	A storage paradigm (e.g., S3 Object Lock in Compliance Mode) where data, once written, cannot be overwritten or deleted by any user, including root admins, until the retention period expires.
Change Data Capture (CDC)	A pattern that continuously monitors a database's transaction log (WAL) and streams row-level changes to downstream systems (like a search engine) without burdening the primary database.
3-2-1-1-0 Backup Rule	3 copies of data, on 2 different media, 1 offsite, 1 offline/immutable, 0 errors verified via testing.
Crypto-Shredding	The process of rendering data permanently unreadable by securely destroying the encryption keys used to encrypt it, rather than attempting to overwrite the physical storage.
Inverted Index	A database index data structure storing a mapping from content (words) to its locations in a document, enabling sub-second full-text search.

The Storage & Backup Doctrine

A SQL `LIKE` query is not a search engine. A mutable snapshot is a ransomware target. An untested backup is a placebo. If the storage tier doesn't match the data temperature, the architecture is economically unsustainable.

Evaluation Methodology

Scoring Model

Each capability is scored from 0 to 5.

Score	Meaning
0	Fails basic task; SQL searches, mutable backups, untested restores, single-tier storage
1	Basic search engine exists, but backups are still mutable
2	Functional backups, but lacks WORM immutability or automated restore testing
3	Solid production performance; Decoupled search (CDC), 3-2-1-1-0 backups, WORM Object Lock, lifecycle tiering
4	Strong performance; Crypto-shredding, cross-region immutable vaults, zero-downtime search re-indexing
5	Best-in-class; sets the standard for mathematically verified, ransomware-proof data architectures

Weighting

Category	Weight	Rationale
Backup Immutability & Ransomware Resilience	20%	Mutable backups guarantee total loss during a ransomware attack
Restore Testing & DR Validation	20%	An untested backup is a liability, not a recovery mechanism
Search Architecture & CDC Decoupling	15%	SQL-based search destroys transactional database performance
Storage Tiering & Lifecycle Management	15%	Keeping cold data on hot storage destroys cloud budgets
Data Sanitization (Crypto-Shredding)	15%	Physical deletion in distributed object storage is impossible
High Availability & Quorum Protection	15%	Split-brain in search clusters causes massive data loss

Total: 100%

A system MUST score at least 3.0 in Backup Immutability and Restore Testing to be considered for production use.

Evaluation Categories

Backup Immutability and Ransomware Resilience (Weight: 20%)

WORM Enforcement

Are backups protected against modification or deletion by compromised admin credentials?

Score	Criteria
0	Backups stored on the same SAN/NAS as production; root admin can delete them
1	Backups pushed to a different cloud region, but using the same IAM credentials
2	Dedicated backup account with separate credentials, but data is still mutable
3	WORM (Write-Once-Read-Many) Object Lock enforced; even root admins cannot delete backups until retention expires
4	Air-gapped immutable vaults; backups cannot be accessed via the production network plane
5	Perfect resilience: formally verified immutable boundaries, zero ability to tamper with historical data, cryptographic proof of backup integrity

Restore Testing and DR Validation (Weight: 20%)

Recovery Assurance

Is the organization mathematically certain that backups can be restored?

Score	Criteria
0	Backups run nightly, but restores are never tested
1	Manual restore tests conducted annually (guarantees gaps in coverage)
2	Automated restore tests run monthly
3	Automated daily restore tests in an isolated environment; hash verification (checksums) validates data integrity
4	Full DR failover tests executed quarterly, verifying RTO/RPO bounds
5	Perfect assurance: formally verified restore pipelines, zero silent corruption, mathematical proof of zero data loss

Search Architecture and CDC Decoupling (Weight: 15%)

Query Performance

How is full-text and semantic search implemented across enterprise data?

Score	Criteria
0	Full-text search executed via SQL `LIKE` queries against the primary database
1	Periodic batch jobs dump data from SQL to Elasticsearch (high latency, stale data)
2	Application-level dual-writes to both SQL and search engine (risk of data inconsistency)
3	Change Data Capture (CDC) via Debezium streams WAL changes to OpenSearch; zero application burden
4	Search cluster utilizes vector indexing (k-NN) for hybrid (text + semantic) search
5	Perfect architecture: formally verified index synchronization, zero query latency impact on transactions, mathematical proof of search consistency

Storage Tiering and Lifecycle Management (Weight: 15%)

Economic Optimization

Is data automatically moved to the cheapest viable storage tier?

Score	Criteria
0	All data (logs, backups, active records) stored on expensive NVMe/SSP block storage
1	Manual migration of old data to cheaper storage
2	Basic S3 lifecycle policies moving objects to Standard-IA after 30 days
3	Automated tiering: hot data on NVMe, warm on HDD, cold data to Glacier/Archive within hours
4	AI-driven lifecycle policies predicting data access patterns and pre-tiering
5	Perfect optimization: formally verified data temperature bounds, zero wasted capital, mathematical proof of optimal tier placement

Data Sanitization (Crypto-Shredding) (Weight: 15%)

Secure Deletion

How is data permanently destroyed in a distributed object storage system?

Score	Criteria
0	Relies on `DELETE` API calls (data actually persists on degraded drives for months)
1	Multi-pass overwrite tools used on block storage
2	Basic object deletion with versioning suspended
3	Crypto-shredding: all objects encrypted with unique Data Encryption Keys (DEKs); deletion = destroying the KEK (Key Encryption Key)
4	Automated key destruction policies triggered by data retention expiry
5	Perfect sanitization: formally verified zero-recoverability, cryptographic attestation of key destruction, zero plaintext remnants

High Availability and Quorum Protection (Weight: 15%)

Split-Brain Prevention

How do search and storage clusters survive network partitions?

Score	Criteria
0	Single-node search/storage clusters (guaranteed data loss on crash)
1	Multi-node clusters, but no quorum enforcement (split-brain corrupts indices)
2	Basic replica shards, but failover is manual
3	Consensus-based quorum (Raft/Paxos) ensuring a single write master; automated failover
4	Cross-zone replication ensuring RPO=0 for search indices
5	Perfect resilience: formally verified consensus bounds, zero split-brain risk, self-healing index rebalancing

The Mythos Storage & Backup Suite (MSBS)

Traditional benchmarks measure backup write speed (MB/s). The MSBS evaluates ransomware resilience, restore integrity, and search decoupling.

Suite Composition

MSBS-Ransomware

1. Root Admin Attack: 100+ tests using compromised root credentials to attempt deletion of recent backups, verifying WORM Object Lock prevents the deletion.
1. Silent Corruption Test: 50+ tests injecting bit-rot into backup archives, verifying the restore pipeline detects checksum mismatches and quarantines the corrupt blocks.

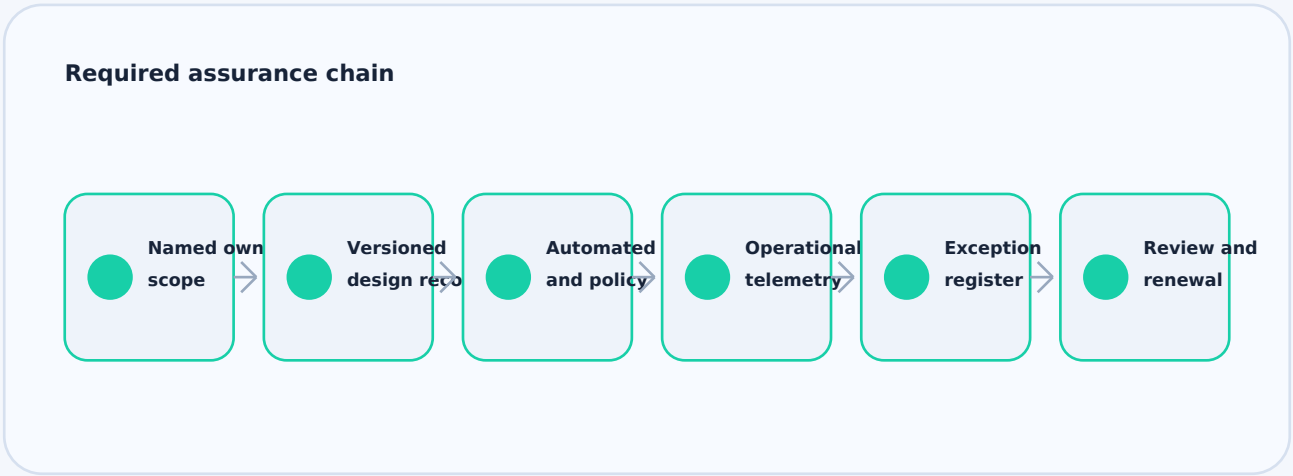
MSBS-Search

1. Dual-Write Elimination Test: 100+ tests verifying the application does not write directly to the search engine, and that CDC latency from SQL to OpenSearch is < 2 seconds.
1. Failover Test: 50+ tests destroying the primary search node during peak query load, verifying the cluster rebalances shards without dropping user requests.

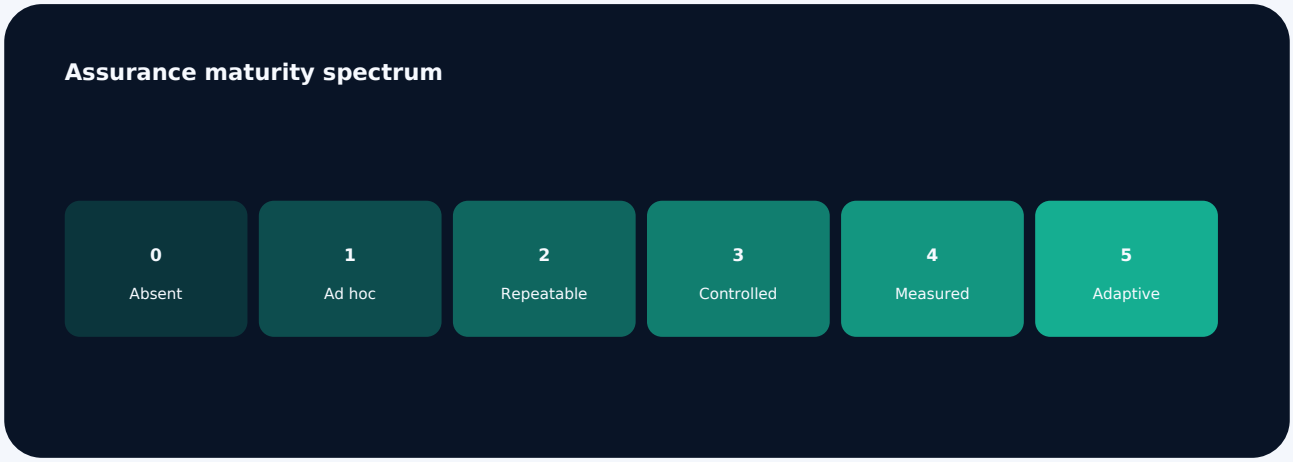
Execution Protocol

ASSURANCE AND ADOPTION

Implementation evidence and review gates



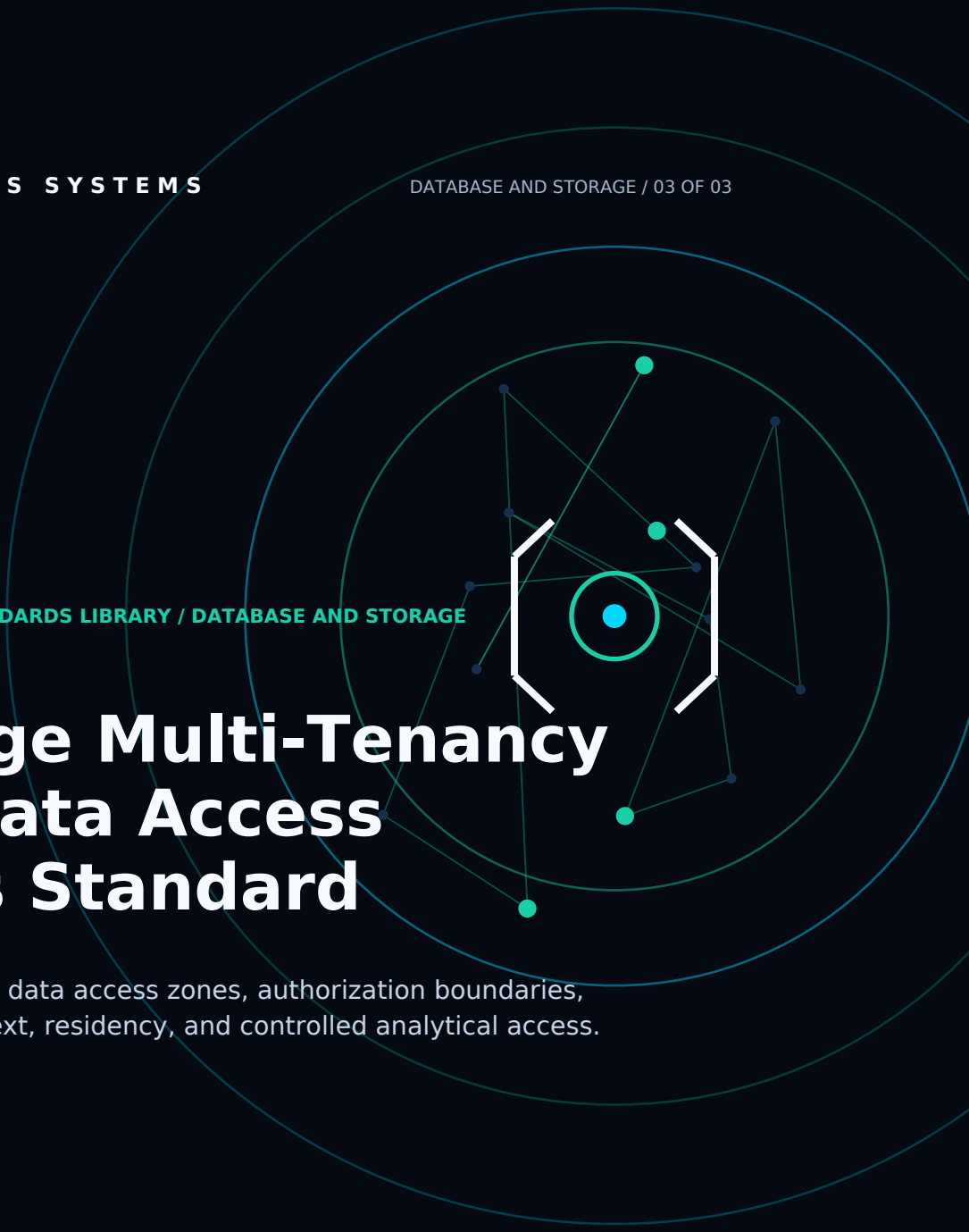
A passing score is not sufficient without current evidence. High-assurance adoption requires traceable design decisions, reproducible builds, independently reviewable tests, controlled secrets, runtime observability, failure exercises, and a dated exception register.



Current authority register

Authority	Version / snapshot	Applicability	Review by
OpenTelemetry Project: OpenTelemetry Documentation and Specification	Rolling specification snapshot; profiles public alpha and declarative configuration stable in 2026	High	2026-10-22
PostgreSQL Global Development Group: PostgreSQL Documentation	Rolling official documentation snapshot	High	2026-10-22

Research limitation. Official documentation establishes published interfaces and declared behavior. It does not prove the performance, security, interoperability, cost, or operational suitability of a specific implementation. Those claims require controlled testing and retained evidence.



ENGINEERING STANDARDS LIBRARY / DATABASE AND STORAGE

Storage Multi-Tenancy and Data Access Zones Standard

Tenant isolation, data access zones, authorization boundaries, encryption context, residency, and controlled analytical access.

PERMANENT DOCUMENT ID

MYTHOS-DBS-0003

PUBLICATION

Candidate Standard
Version 1.0.0-candidate

ASSURANCE PROFILE

Standard / Advanced
High Assurance

PERMANENT PUBLICATION IDENTITY

Storage Multi-Tenancy and Data Access Zones Standard

DOCUMENT ID
MYTHOS-DBS-0003

VERSION
1.0.0-candidate

STATUS
Candidate Standard

PUBLISHER
Mythos Systems

PUBLICATION DATE
2026-07-24

SCHEDULED REVIEW
2027-01-24

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

Publication doctrine. This publication establishes durable guidance or requirements for its defined scope. Interpretation must preserve the permanent document identity, recorded evidence, controlled exceptions, and formal revision history.

DOMAIN CONTROL SYSTEM

MYTHOS-DBS-0003 inside the database and storage standard constellation

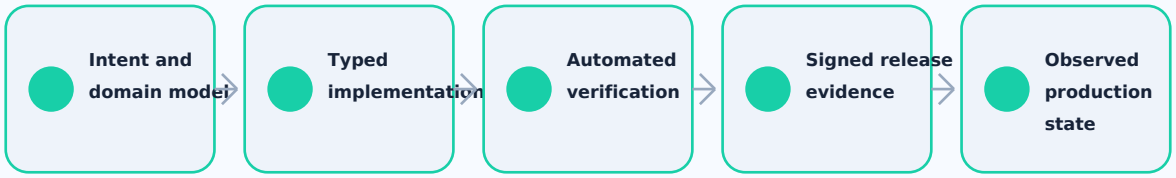
Connected assurance	Architecture, implementation, operations, evidence, and lifecycle are evaluated as one controlled system.
Specialization rule	Companion standards may add precision, but they cannot silently weaken this publication's requirements.

EXECUTIVE POSITION

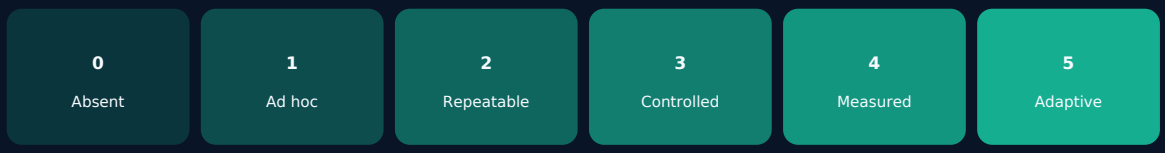
Storage Multi-Tenancy and Data Access Zones Standard

The architecture of multi-tenant storage has failed.

Evidence-bearing delivery path



Assurance maturity spectrum



CONTENTS

Publication navigation

MYTHOS-DBS-0003 inside the database and storage standard constellation . . 3

Storage Multi-Tenancy and Data Access Zones Standard 4

Assurance model and evaluation surface 7

Criteria and evidence map 8

Requirements, evaluation methodology, and implementation guidance 9

Executive Mandate 9

Scope and Applicability 9

In Scope 9

Out of Scope 10

Audience 10

Definitions and Taxonomy 10

Storage Tenancy Dimensions 10

The Storage Tenancy Doctrine 11

Evaluation Methodology 11

Scoring Model 11

Weighting 11

Evaluation Categories 12

Logical Isolation (DAZ Boundaries) (Weight: 20%) 12

Tenant Visibility 12

Independent Authentication Mapping (Weight: 20%) 12

Identity Boundary 12

Dedicated Network Groupnets (Weight: 15%) 12

Network Traffic Isolation 12

Kubernetes CSI Integration (Weight: 15%) 13

Cloud-Native Storage 13

Per-Zone Encryption (KMS) (Weight: 15%) 13

Cryptographic Sovereignty 13

Observability and Auditability (Weight: 15%) 14

Per-Tenant Telemetry 14

The Mythos Storage Suite (MSTS) 14

Suite Composition 14

MSTS-Isolation 14

MSTS-Crypto 14

Execution Protocol 14

Implementation evidence and review gates 15

Current authority register 15

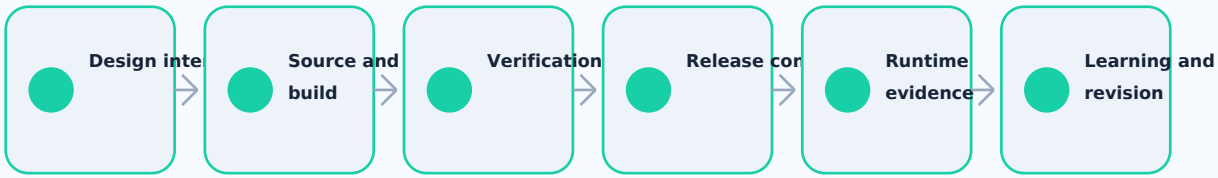
CONTROL TOPOLOGY

Assurance model and evaluation surface

The standard is assessed as an evidence-bearing system. Capability scores do not replace mandatory gates, implementation proof, or operational validation.

Capability family	Evaluation nodes
Logical Isolation (DAZ Boundaries) (Weight: 20%)	1
Independent Authentication Mapping (Weight: 20%)	1
Dedicated Network Groupnets (Weight: 15%)	1
Kubernetes CSI Integration (Weight: 15%)	1
Per-Zone Encryption (KMS) (Weight: 15%)	1
Observability and Auditability (Weight: 15%)	1
Suite Composition	2
Additional Evaluation Dimension	4

Lifecycle control loop



EVALUATION INDEX

Criteria and evidence map

#	Capability family	Criterion
1	Logical Isolation (DAZ Boundaries) (Weight: 20%)	Tenant Visibility
2	Independent Authentication Mapping (Weight: 20%)	Identity Boundary
3	Dedicated Network Groupnets (Weight: 15%)	Network Traffic Isolation
4	Kubernetes CSI Integration (Weight: 15%)	Cloud-Native Storage
5	Per-Zone Encryption (KMS) (Weight: 15%)	Cryptographic Sovereignty
6	Observability and Auditability (Weight: 15%)	Per-Tenant Telemetry
7	Suite Composition	MSTS-Isolation
8	Suite Composition	MSTS-Crypto
9	Additional Evaluation Dimension	Storage Tenancy Dimensions
10	Additional Evaluation Dimension	The Storage Tenancy Doctrine
11	Additional Evaluation Dimension	Scoring Model
12	Additional Evaluation Dimension	Suite Composition

NORMATIVE PUBLICATION

Requirements, evaluation methodology, and implementation guidance

The following body preserves the substantive source standard while normalizing publication status, identity, navigation, and visual presentation.

Executive Mandate

The architecture of multi-tenant storage has failed.

Organizations attempt to consolidate massive unstructured data lakes onto a single scale-out storage cluster (e.g., Dell PowerScale/Isilon, NetApp ONTAP) to maximize hardware utilization. To isolate tenants, they rely on filesystem permissions (POSIX ACLs, NTFS) and export rules. When a tenant's application is compromised, or an administrator misconfigures a share, the attacker can mount the cluster and map the directory structures of every other tenant on the physical fabric. Furthermore, because the entire cluster shares a single authentication provider (Active Directory), a compromise of the storage cluster's service account grants instant access to all data across all tenants.

This standard exists because:

1. Shared Namespaces are a Liability: Relying on filesystem permissions to isolate tenants guarantees eventual data leakage. Systems MUST implement logical Data Access Zones (DAZ) that partition the cluster at the OS and network levels, mathematically preventing cross-tenant visibility.
2. Authentication Must Be Bounded: A single cluster-wide authentication realm is a critical blast radius. Systems MUST map independent authentication providers (AD/LDAP) to each DAZ. A compromised credential in Tenant A's Active Directory MUST NOT be valid in Tenant B's Access Zone.
3. Network Traffic Must Be Isolated: Tenants sharing the same storage IP interfaces allows for packet sniffing and IP-spoofing based attacks. Systems MUST utilize dedicated network groupnets, subnets, and IP pools per DAZ, enforced by network microsegmentation.
4. Cloud-Native Storage Must Be Zone-Aware: Kubernetes persistent volumes (PVs) provisioned via Container Storage Interface (CSI) drivers frequently default to broad cluster access. Systems MUST bind CSI storage classes to specific DAZs, ensuring containerized workloads are physically incapable of mounting volumes outside their tenant boundary.

This document establishes the Mythos Storage Tenancy Standard (MSTS), a comprehensive, evidence-based methodology for evaluating enterprise storage clusters against absolute tenant isolation, cryptographic data sovereignty, and zero-trust access boundaries.

Scope and Applicability

In Scope

This standard covers the evaluation of:

1. Scale-out storage architectures (Dell PowerScale/Isilon, NetApp, Pure FlashBlade)
1. Data Access Zones (DAZ) and logical cluster partitioning
1. Multi-tenant authentication mapping (AD/LDAP/IdP integration)
1. Network groupnets, subnet pools, and SmartConnect zones
1. Kubernetes Container Storage Interface (CSI) multi-tenancy
1. Per-zone Key Management Service (KMS) integration and cryptographic erasure

Out of Scope

1. General relational database scaling (covered in MYTHOS-DBS-0001)
1. General network microsegmentation policy (covered in MYTHOS-SEC-0007)
1. General identity and PKI infrastructure (covered in MYTHOS-ID-0001)

Audience

1. Storage architects and infrastructure engineers
1. Platform engineers managing Kubernetes stateful workloads
1. Security engineers evaluating multi-tenant data isolation
1. Compliance teams managing sovereign data boundaries
1. Standards bodies seeking a reference storage tenancy methodology

Definitions and Taxonomy

Storage Tenancy Dimensions

Dimension	Definition
Data Access Zone (DAZ)	A logical partition within a physical storage cluster that isolates data, network configurations, and authentication systems, rendering tenants invisible to one another.
Groupnet	A networking construct (specifically in PowerScale/Isilon) that defines a distinct subnet, IP address pool, and DNS zone for a specific Access Zone, ensuring network traffic isolation.
Auth Mapping	The architectural practice of binding a specific, external authentication provider (e.g., a dedicated Active Directory forest or LDAP tree) exclusively to a single Data Access Zone.
Zone-Aware CSI	A Kubernetes Container Storage Interface driver configuration where StorageClasses are explicitly bound to a specific tenant's Access Zone, restricting PV provisioning to that isolated domain.
Cryptographic Erasure	The process of rendering a tenant's data permanently unreadable by securely destroying the specific Data Encryption Key (DEK) assigned to their Access Zone, rather than overwriting the physical disks.

The Storage Tenancy Doctrine

> A filesystem permission is a suggestion; an Access Zone is a boundary. A shared authentication realm is a critical blast radius. A shared IP interface is a sniffing vector. If Tenant A can map Tenant B's directory structure, the architecture has failed.

Evaluation Methodology

Scoring Model

Each capability is scored from 0 to 5.

Score	Meaning
0	Fails basic task; single namespace, shared AD, filesystem ACLs for isolation
1	Basic share-level access, but shared authentication and network planes
2	Functional isolation, but lacks independent auth mapping or CSI integration
3	Solid production performance; DAZ implemented, independent AD, dedicated groupnets, zone-aware CSI
4	Strong performance; Per-zone KMS encryption, formally verified network isolation
5	Best-in-class; sets the standard for mathematically proven, zero-trust storage tenancy

Weighting

Category	Weight	Rationale
Logical Isolation (DAZ Boundaries)	20%	Shared namespaces guarantee eventual cross-tenant data leakage
Independent Authentication Mapping	20%	Shared AD realms allow compromised credentials to pivot across tenants
Dedicated Network Groupnets	15%	Shared IP interfaces allow packet sniffing and ARP spoofing
Kubernetes CSI Integration	15%	Default CSI configurations allow containers to mount any PV
Per-Zone Encryption (KMS)	15%	Cluster-wide encryption keys prevent secure cryptographic erasure
Observability & Auditability	15%	Multi-tenant environments require per-tenant audit trails

Total: 100%

A system MUST score at least 3.0 in Logical Isolation and Independent Authentication to be considered for production.

Evaluation Categories

Logical Isolation (DAZ Boundaries) (Weight: 20%)

Tenant Visibility

Is the physical storage cluster partitioned into logically invisible Access Zones?

Score	Criteria
0	Single namespace; tenants share the same root directory and can browse each other's folders
1	Separate shares/export points, but the underlying namespace is visible
2	Separate filesystems/subdirectories, but managed via a single cluster control plane
3	Full Data Access Zone (DAZ) implementation: Tenants connect to zone-specific IP interfaces and can only see their own root directory; cross-zone visibility is abolished
4	Zone-specific SMB shares and NFS exports with strict host-based access controls mapped to DAZ network pools
5	Perfect isolation: Formally verified DAZ boundaries, zero ability for a tenant to enumerate or access another tenant's data structures, cryptographic proof of tenancy separation

Independent Authentication Mapping (Weight: 20%)

Identity Boundary

Does each tenant utilize a dedicated authentication provider, or do they share a cluster-wide AD?

Score	Criteria
0	Single cluster-wide Active Directory/LDAP integration for all tenants
1	Multiple AD domains, but joined into a single trust forest on the storage cluster
2	Separate AD domains, but the storage cluster uses a shared service account to join them all
3	Independent Auth Mapping: Each DAZ is joined to a completely separate, tenant-specific AD/LDAP/IdP with no trust relationship
4	Per-zone local service accounts; cluster management plane utilizes ZTNA and PAM, decoupled from tenant data access
5	Perfect boundary: Formally verified identity isolation, zero cross-tenant credential validity, cryptographic non-repudiation of per-zone access

Dedicated Network Groupnets (Weight: 15%)

Network Traffic Isolation

How is storage network traffic isolated between tenants?

Score	Criteria
0	All tenants connect to the same storage IP interfaces (SmartConnect zones)
1	Separate VLANs, but routed through the same storage cluster interfaces

Score	Criteria
2	Dedicated IP pools per tenant, but sharing the same physical L2 broadcast domain
3	Dedicated Groupnets: Each DAZ possesses its own subnet, IP address pool, and DNS zone, mapped to dedicated physical or logically partitioned network interfaces
4	Storage interfaces mapped to tenant-specific VRFs (Virtual Routing and Forwarding) or physical DPUs, enforcing L3 isolation
5	Perfect isolation: Formally verified network boundaries, zero ability to sniff or spoof neighboring tenant traffic, mathematical proof of groupnet segregation

Kubernetes CSI Integration (Weight: 15%)

Cloud-Native Storage

How are Kubernetes persistent volumes (PVs) provisioned and bound to tenants?

Score	Criteria
0	Single default StorageClass used by all namespaces; any pod can request a PV
1	Multiple StorageClasses, but they share the same underlying storage pool and access zone
2	Tenant-specific StorageClasses, but lacking network or auth isolation on the backend
3	Zone-Aware CSI: StorageClasses are explicitly parameterized to provision volumes within a specific DAZ; Kubernetes RBAC restricts namespaces to their corresponding StorageClass
4	CSI drivers utilize per-tenant cloud credentials or Kerberos tickets to mount volumes, authenticating directly to the tenant's DAZ
5	Perfect integration: Formally verified CSI RBAC, zero ability for a container to mount a cross-tenant PV, mathematical proof of cloud-native tenancy isolation

Per-Zone Encryption (KMS) (Weight: 15%)

Cryptographic Sovereignty

Is data encrypted at rest with per-tenant keys?

Score	Criteria
0	No encryption at rest
1	Cluster-wide encryption utilizing a single shared key
2	Self-encrypting drives (SEDs) with a single global key
3	Per-zone KMS integration: Each DAZ is bound to a tenant-specific Key Encryption Key (KEK) in an external KMS (e.g., HashiCorp Vault, AWS KMS)
4	Per-volume/per-file encryption utilizing tenant-specific Data Encryption Keys (DEKs) wrapped by the zone KEK
5	Perfect sovereignty: Formally verified cryptographic boundaries, zero ability to read tenant data without their specific KEK, mathematical proof of secure cryptographic erasure upon tenant offboarding

Observability and Auditability (Weight: 15%)

Per-Tenant Telemetry

Can security teams attribute every storage access event to a specific tenant and zone?

Score	Criteria
0	Single cluster-wide audit log; impossible to distinguish tenant actions
1	Basic logging, but lacks zone context or user attribution
2	Per-zone logging, but requires manual extraction and correlation
3	Per-DAZ audit logs streamed to a centralized SIEM, tagged with tenant ID, zone ID, and user identity
4	Real-time anomaly detection per tenant (e.g., detecting a sudden spike in reads indicative of data exfiltration)
5	Perfect observability: Formally verified audit trails, zero blind spots in cross-tenant telemetry, cryptographic attestation of per-zone data access

The Mythos Storage Suite (MSTS)

Traditional benchmarks measure IOPS (Input/Output Operations Per Second). The MSTS evaluates cross-tenant visibility, auth isolation, and cryptographic sovereignty.

Suite Composition

MSTS-Isolation

1. Cross-Zone Mount Test: 100+ tests attempting to mount Tenant B's NFS export or SMB share using Tenant A's credentials and network interface, verifying the DAZ and groupnet boundaries drop the connection.
1. Namespace Enumeration Test: 50+ tests attempting to browse the root directory of the storage cluster from a tenant context, verifying the DAZ restricts visibility to only the tenant's specific directory.

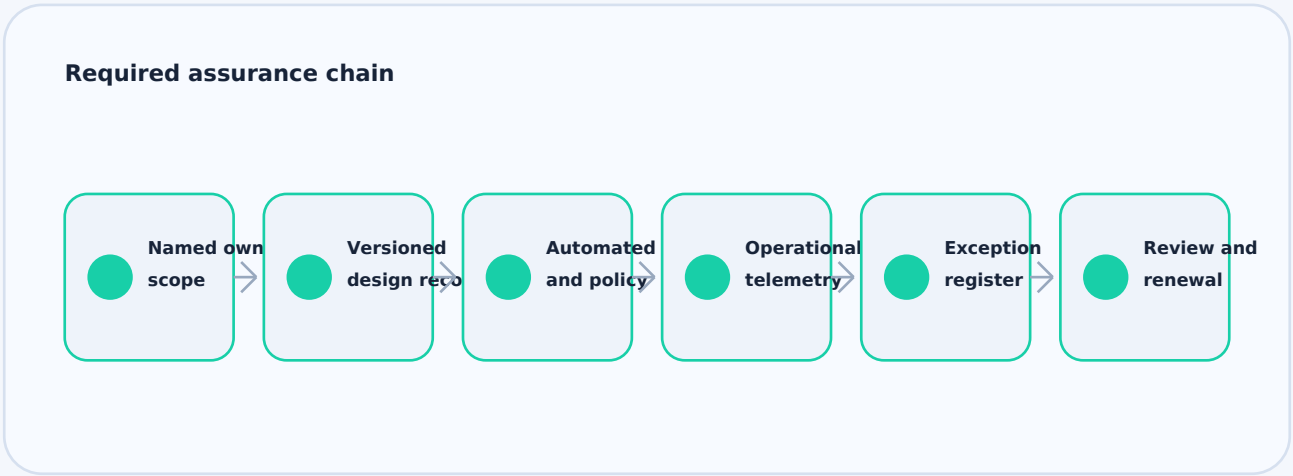
MSTS-Crypto

1. Cryptographic Erasure Test: 100+ tests simulating the offboarding of a tenant, verifying that destroying their KMS key immediately renders all their data on the physical cluster permanently unreadable (zero plaintext remnants).
1. CSI Privilege Escalation Test: 50+ tests attempting to provision a Persistent Volume in Tenant A's Kubernetes namespace using Tenant B's StorageClass, verifying RBAC and zone-aware CSI block the action.

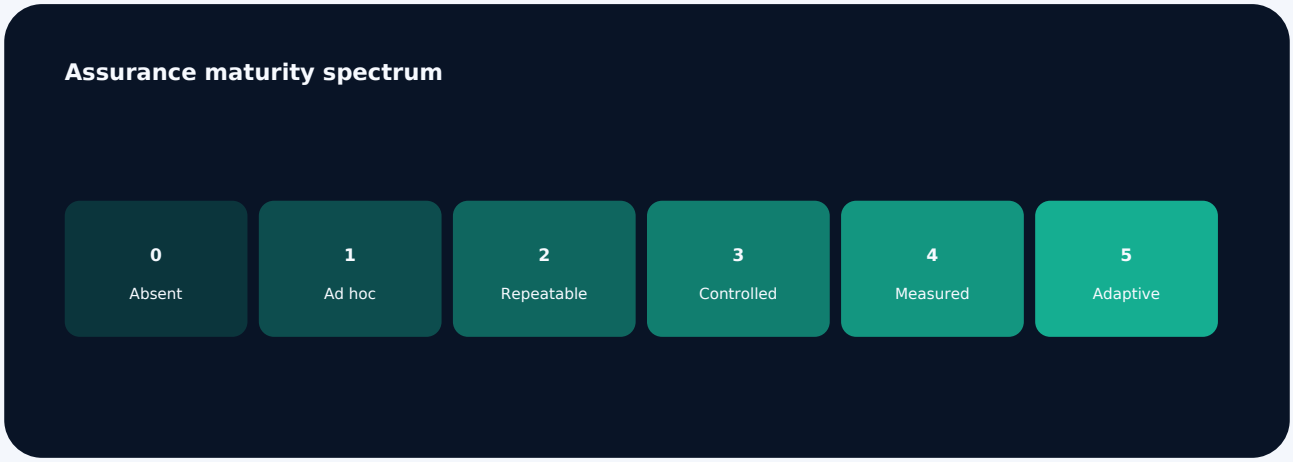
Execution Protocol

ASSURANCE AND ADOPTION

Implementation evidence and review gates



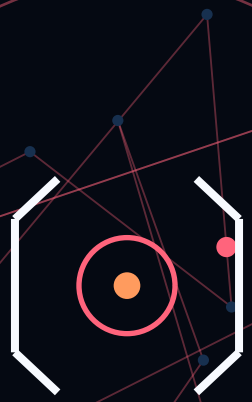
A passing score is not sufficient without current evidence. High-assurance adoption requires traceable design decisions, reproducible builds, independently reviewable tests, controlled secrets, runtime observability, failure exercises, and a dated exception register.



Current authority register

Authority	Version / snapshot	Applicability	Review by
OpenTelemetry Project: OpenTelemetry Documentation and Specification	Rolling specification snapshot; profiles public alpha and declarative configuration stable in 2026	High	2026-10-22
PostgreSQL Global Development Group: PostgreSQL Documentation	Rolling official documentation snapshot	High	2026-10-22

Research limitation. Official documentation establishes published interfaces and declared behavior. It does not prove the performance, security, interoperability, cost, or operational suitability of a specific implementation. Those claims require controlled testing and retained evidence.



ENGINEERING STANDARDS LIBRARY / RELIABILITY ENGINEERING

SLOs, Error Budgets, and Capacity Planning Standard

Service objectives, error-budget governance, capacity models, demand forecasting, and reliability investment decisions.

PERMANENT DOCUMENT ID

MYTHOS-SRE-0001

PUBLICATION

Candidate Standard
Version 1.0.0-candidate

ASSURANCE PROFILE

Standard / Advanced
High Assurance

PERMANENT PUBLICATION IDENTITY

SLOs, Error Budgets, and Capacity Planning Standard

DOCUMENT ID
MYTHOS-SRE-0001

VERSION
1.0.0-candidate

STATUS
Candidate Standard

PUBLISHER
Mythos Systems

PUBLICATION DATE
2026-07-24

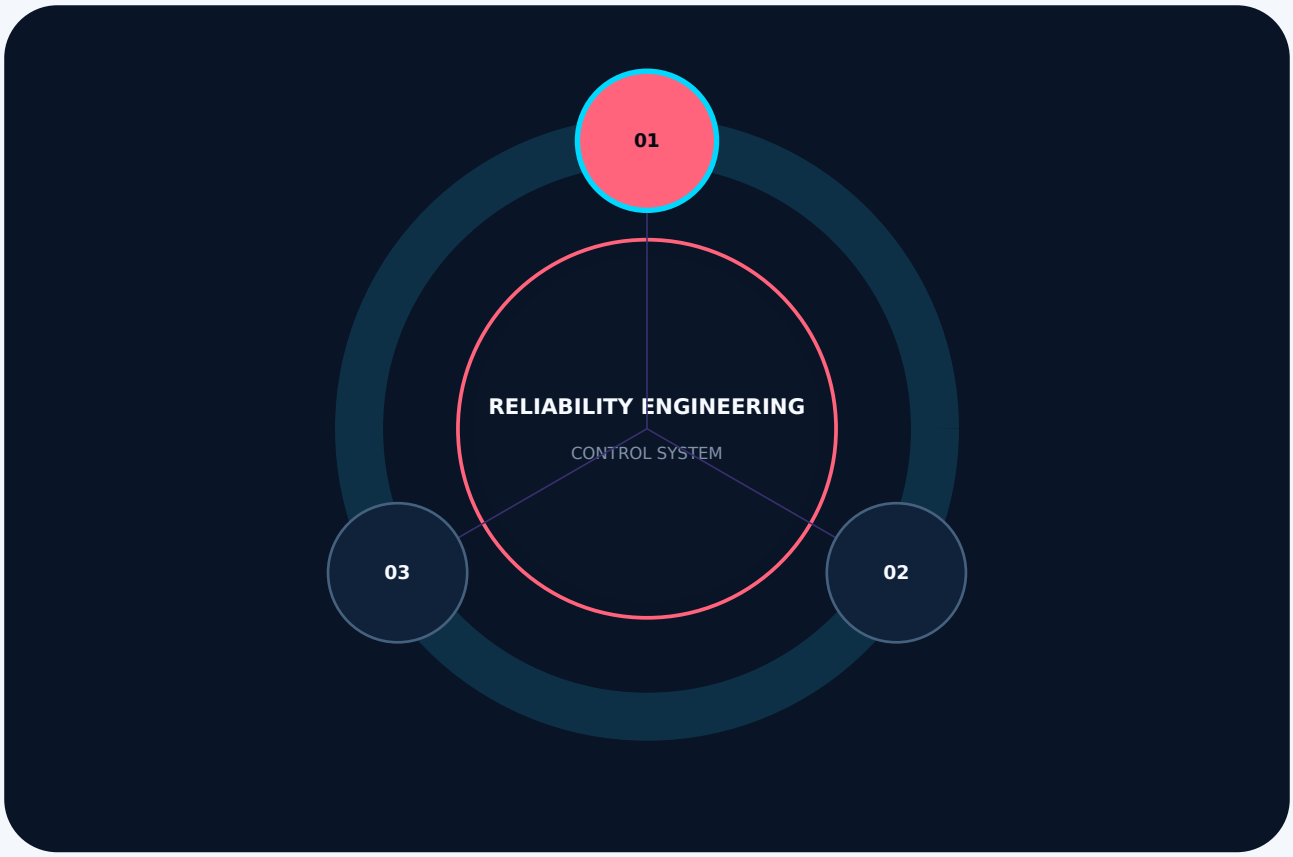
SCHEDULED REVIEW
2027-01-24

11	6	4	1
ATOMIC CRITERIA	ASSURANCE VIEWS	IMPLEMENTATION PROFILES	PERMANENT IDENTITY

Publication doctrine. This publication establishes durable guidance or requirements for its defined scope. Interpretation must preserve the permanent document identity, recorded evidence, controlled exceptions, and formal revision history.

DOMAIN CONTROL SYSTEM

MYTHOS-SRE-0001 inside the reliability engineering standard constellation



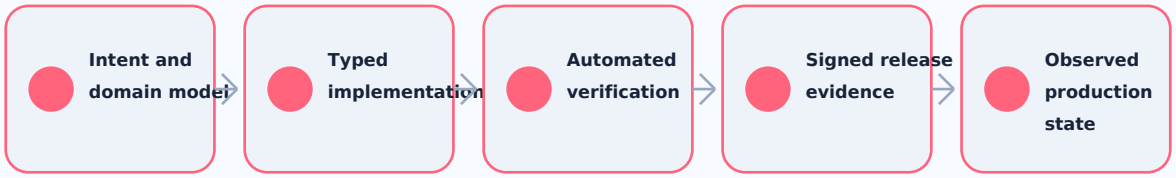
Connected assurance	Architecture, implementation, operations, evidence, and lifecycle are evaluated as one controlled system.
Specialization rule	Companion standards may add precision, but they cannot silently weaken this publication's requirements.

EXECUTIVE POSITION

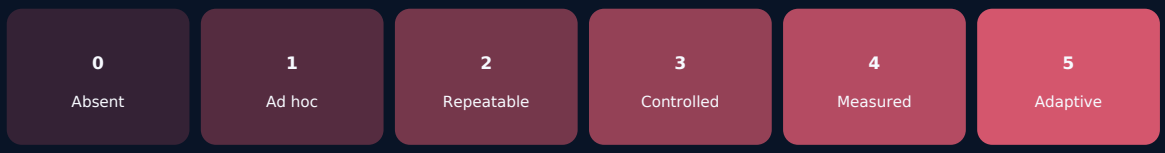
SLOs, Error Budgets, and Capacity Planning Standard

The architecture of reliability and capacity management has failed.

Evidence-bearing delivery path



Assurance maturity spectrum



CONTENTS

Publication navigation

MYTHOS-SRE-0001 inside the reliability engineering standard constellation . . 3

SLOs, Error Budgets, and Capacity Planning Standard 4

Assurance model and evaluation surface 7

Criteria and evidence map 8

Requirements, evaluation methodology, and implementation guidance 9

Executive Mandate 9

Scope and Applicability 9

In Scope 9

Out of Scope 10

Audience 10

Definitions and Taxonomy 10

Reliability Dimensions 10

The SRE Doctrine 10

Evaluation Methodology 11

Scoring Model 11

Weighting 11

Evaluation Categories 11

SLI/SLO Definition (User-Centric) (Weight: 20%) 11

Measurement Precision 11

Error Budget Enforcement (Weight: 20%) 12

Velocity Governance 12

Multi-Window Burn-Rate Alerting (Weight: 20%) 12

Alerting Precision 12

Predictive Capacity Planning (Weight: 20%) 12

Scaling Mechanics 12

Toil Reduction and Automation (Weight: 10%) 13

Operational Overhead 13

Reliability Culture and Velocity (Weight: 10%) 13

Business Alignment 13

The Mythos SRE Suite (MSRES) 14

Suite Composition 14

MSRES-SLO 14

MSRES-Capacity 14

Execution Protocol 14

Implementation evidence and review gates 15

Current authority register 16

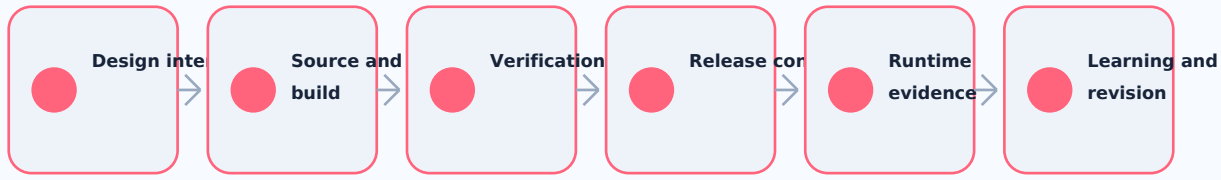
CONTROL TOPOLOGY

Assurance model and evaluation surface

The standard is assessed as an evidence-bearing system. Capability scores do not replace mandatory gates, implementation proof, or operational validation.

Capability family	Evaluation nodes
SLI/SLO Definition (User-Centric) (Weight: 20%)	1
Error Budget Enforcement (Weight: 20%)	1
Multi-Window Burn-Rate Alerting (Weight: 20%)	1
Predictive Capacity Planning (Weight: 20%)	1
Toil Reduction and Automation (Weight: 10%)	1
Reliability Culture and Velocity (Weight: 10%)	1
Suite Composition	2
Additional Evaluation Dimension	3

Lifecycle control loop



EVALUATION INDEX

Criteria and evidence map

#	Capability family	Criterion
1	SLI/SLO Definition (User-Centric) (Weight: 20%)	Measurement Precision
2	Error Budget Enforcement (Weight: 20%)	Velocity Governance
3	Multi-Window Burn-Rate Alerting (Weight: 20%)	Alerting Precision
4	Predictive Capacity Planning (Weight: 20%)	Scaling Mechanics
5	Toil Reduction and Automation (Weight: 10%)	Operational Overhead
6	Reliability Culture and Velocity (Weight: 10%)	Business Alignment
7	Suite Composition	MSRES-SLO
8	Suite Composition	MSRES-Capacity
9	Additional Evaluation Dimension	Reliability Dimensions
10	Additional Evaluation Dimension	The SRE Doctrine
11	Additional Evaluation Dimension	Scoring Model

NORMATIVE PUBLICATION

Requirements, evaluation methodology, and implementation guidance

The following body preserves the substantive source standard while normalizing publication status, identity, navigation, and visual presentation.

Executive Mandate

The architecture of reliability and capacity management has failed.

Organizations measure system health by tracking whether a server is pingable or if CPU utilization is below 80%. When a user clicks "Checkout" and the API takes 10 seconds to load, the monitoring dashboard glows green because the server is "up." Engineers are paged at 3 AM for high CPU, but users are perfectly happy. Meanwhile, the system runs out of memory during a marketing spike and crashes because capacity planning was based on last month's static traffic averages.

This standard exists because:

1. Uptime is a Vanity Metric: A server returning HTTP 500 errors is "up." A database returning data in 30 seconds is "up." If the user cannot complete their transaction, the system is down. Reliability MUST be measured by user-centric Service Level Indicators (SLIs) and Service Level Objectives (SLOs).
2. Alerting on Symptoms Causes Fatigue: Paging engineers because CPU is high is paging on a symptom. If the user experience is not degraded, the high CPU is irrelevant. Systems MUST utilize multi-window, multi-burn-rate alerting based *only* on the consumption of the error budget.
3. Error Budgets Must Govern Velocity: If the system is perfectly reliable, you are not moving fast enough. If the system is failing, you must stop pushing features. The error budget MUST be a mathematical construct that automatically freezes CI/CD deployments when exhausted, shifting focus to reliability work.
4. Reactive Scaling is a Dead End: Waiting for CPU to hit 80% to trigger an autoscaler guarantees latency spikes during the provisioning window. Capacity planning MUST be predictive, utilizing time-series forecasting and scaling based on SLO headroom, not raw infrastructure metrics.

This document establishes the Mythos SRE Standard (MSRES), a comprehensive, evidence-based methodology for evaluating reliability programs against user-centric measurement, error budget enforcement, and predictive capacity scaling.

Scope and Applicability

In Scope

This standard covers the evaluation of:

1. SLI (Service Level Indicator) and SLO (Service Level Objective) definitions
1. Error budget calculation, tracking, and policy enforcement (CI/CD freezes)

1. Multi-window, multi-burn-rate alerting strategies
1. Predictive capacity planning and time-series forecasting
1. Autoscaling mechanisms (Kubernetes HPA/VPA, Cloud Autoscaling Groups)
1. Toil reduction and reliability engineering culture

Out of Scope

1. General observability semantic conventions (covered in MYTHOS-DAT-0002)
1. General disaster recovery and chaos engineering (covered in MYTHOS-SRE-0002)
1. General cloud cost management (covered in MYTHOS-SRE-0003)

Audience

1. Site Reliability Engineers (SREs) and platform architects
1. DevOps and infrastructure engineers managing autoscaling
1. Engineering managers and product owners governing feature velocity
1. Security engineers evaluating availability attack surfaces
1. Standards bodies seeking a reference SRE methodology

Definitions and Taxonomy

Reliability Dimensions

Dimension	Definition
SLI (Service Level Indicator)	A quantitative measure of the user experience (e.g., the ratio of successful HTTP requests to total HTTP requests, or the percentage of requests under 200ms).
SLO (Service Level Objective)	The target value for an SLI over a specific time window (e.g., 99.9% of requests will succeed over 30 days).
Error Budget	The inverse of the SLO (e.g., 0.1%). The maximum allowable amount of unreliability or bad user experience before the system breaches its SLO.
Burn Rate	The rate at which the error budget is being consumed relative to time. A burn rate of 1 means the budget will be exhausted exactly at the end of the SLO window.
Multi-Window Alerting	An alerting strategy that combines a short window (e.g., 5 minutes) and a long window (e.g., 1 hour) to detect both fast-burning and slow-burning error budget consumption without alert fatigue.
Predictive Scaling	The use of time-series forecasting and machine learning to provision capacity ahead of predicted demand spikes, rather than reacting to current load.

The SRE Doctrine

> Uptime is a vanity metric. CPU utilization is a symptom. An error budget that doesn't stop feature deployments is just math. If capacity scaling is reactive, you are already down.

Evaluation Methodology

Scoring Model

Each capability is scored from 0 to 5.

Score	Meaning
0	Fails basic task; CPU alerts, uptime tracking, manual scaling, no SLOs
1	Basic SLOs defined, but not enforced; alerting on static thresholds
2	Functional SLOs, but lacks error budget enforcement or predictive scaling
3	Solid production performance; User-centric SLOs, error budget CI/CD gates, burn-rate alerts, predictive scaling
4	Strong performance; Multi-window alerting, AI-driven capacity forecasting, zero toil
5	Best-in-class; sets the standard for mathematically verified, autonomous reliability

Weighting

Category	Weight	Rationale
SLI/SLO Definition (User-Centric)	20%	Measuring infrastructure instead of user experience guarantees false positives
Error Budget Enforcement	20%	SLOs without error budget enforcement are merely suggestions
Multi-Window Burn-Rate Alerting	20%	Static threshold alerts cause fatigue or miss slow-burning failures
Predictive Capacity Planning	20%	Reactive scaling guarantees latency spikes during provisioning delays
Toil Reduction & Automation	10%	Manual scaling and ticket-driven capacity requests are unsustainable
Reliability Culture & Velocity	10%	SRE must balance reliability with feature velocity, not just say "no"

Total: 100%

A system MUST score at least 3.0 in SLI/SLO Definition and Error Budget Enforcement to be considered for production use.

Evaluation Categories

SLI/SLO Definition (User-Centric) (Weight: 20%)

Measurement Precision

Are reliability metrics based on user experience or infrastructure health?

Score	Criteria
0	Alerts based on CPU, RAM, and Disk utilization; "Uptime" tracked via ping
1	Basic HTTP 5xx error rate tracked
2	SLIs track latency (p99) and availability, but SLOs are arbitrary ("99.9% because it sounds good")

Score	Criteria
3	SLIs strictly define user journeys (e.g., "Checkout API < 500ms"); SLOs derived from business requirements
4	SLIs incorporate cognitive load (e.g., UI render time, agent tool execution latency)
5	Perfect measurement: formally verified SLI fidelity, mathematical proof that metrics perfectly reflect user experience, zero infrastructure vanity metrics

Error Budget Enforcement (Weight: 20%)

Velocity Governance

Does the error budget actually govern engineering behavior?

Score	Criteria
0	No error budget concept; features deployed regardless of reliability
1	Error budget tracked, but only reviewed in monthly meetings
2	Alerts fire when error budget is low, but no automated enforcement
3	Automated CI/CD freeze: deploys blocked when error budget is exhausted; focus shifts to reliability
4	Tiered policy: fast burns freeze immediately; slow burns require post-mortems; unused budget allows riskier deployments
5	Perfect enforcement: formally verified budget bounds, zero ability to bypass CI/CD freezes, mathematical proof of optimal velocity-to-reliability ratio

Multi-Window Burn-Rate Alerting (Weight: 20%)

Alerting Precision

Are engineers paged only when the user experience is actively degrading?

Score	Criteria
0	Static threshold alerts (e.g., "If errors > 1% for 5 minutes, page")
1	Basic burn-rate alerting, but single-window (prone to false positives)
2	Multi-window alerting (e.g., 1h/5m), but high alert fatigue
3	Multi-window, multi-burn-rate alerting (e.g., 1h/5m for fast burn, 6h/30m for slow burn); zero alerts for sub-budget consumption
4	Alerting incorporates business impact (e.g., checkout failures page faster than search failures)
5	Perfect precision: formally verified burn-rate bounds, zero false positives, mathematical proof that every page requires human intervention

Predictive Capacity Planning (Weight: 20%)

Scaling Mechanics

How does the system handle load spikes?

Score	Criteria
0	Manual capacity planning; servers provisioned based on static monthly averages
1	Reactive autoscaling based on CPU/RAM (too slow, causes initial latency spikes)
2	Autoscaling based on queue depth or latency (reactive but better)
3	Predictive scaling using time-series forecasting (e.g., scaling up before the morning rush based on historical data)
4	AI-driven scaling incorporating external signals (e.g., marketing campaign schedules, weather)
5	Perfect planning: formally verified capacity bounds, zero provisioning latency, mathematical proof of optimal resource allocation

Toil Reduction and Automation (Weight: 10%)

Operational Overhead

Is human effort required to maintain scale and reliability?

Score	Criteria
0	Manual scaling, manual failover, ticket-driven capacity requests
1	Basic scripts automate common tasks, but require human initiation
2	Autoscaling handles compute, but database/storage scaling is manual
3	Fully automated scaling for compute, network, and storage; toil budget tracked (< 50% of SRE time)
4	Self-healing systems automatically remediate common failures (e.g., restarting unhealthy pods, clearing caches)
5	Perfect automation: formally verified zero-touch operations, zero manual capacity intervention, mathematical proof of minimal toil

Reliability Culture and Velocity (Weight: 10%)

Business Alignment

Does SRE act as a gatekeeper or a partner to product engineering?

Score	Criteria
0	SRE/Ops says "no" to all deployments; pure gatekeeper mentality
1	SRE allows deployments but blames devs when things break
2	SLOs exist, but product teams ignore them to hit feature deadlines
3	Shared ownership: product and SRE teams agree on SLOs and error budget policies together
4	Unused error budget explicitly incentivizes developers to take calculated risks (e.g., database migrations)
5	Perfect alignment: formally verified reliability incentives, zero friction between velocity and stability, mathematical proof of business value delivery

The Mythos SRE Suite (MSRES)

Traditional benchmarks measure alert response time. The MSRES evaluates SLI fidelity, error budget enforcement, and predictive scaling accuracy.

Suite Composition

MSRES-SLO

1. User Impact Simulation: 100+ tests simulating high-latency API responses (p99 > 500ms) while keeping CPU low, verifying the SLO alerting fires based on latency, not CPU.
1. Budget Exhaustion Test: 50+ tests consuming the error budget to zero, verifying the CI/CD pipeline automatically blocks new feature deployments.

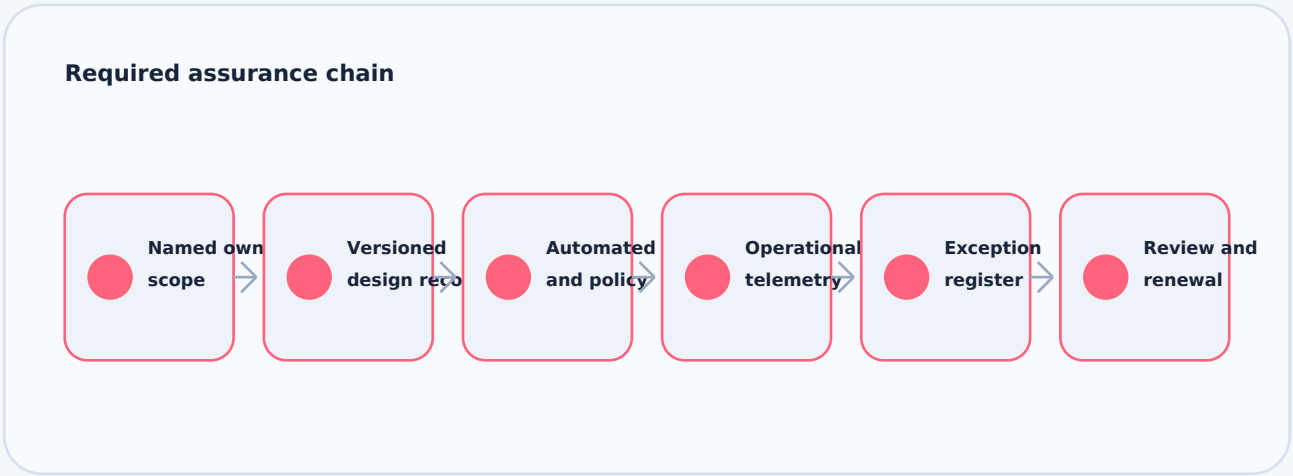
MSRES-Capacity

1. Traffic Spike Test: 100+ tests simulating a 500% traffic spike over 1 minute, verifying predictive scaling has pre-provisioned capacity and SLOs are not breached.
1. Slow Burn Test: 50+ tests simulating a gradual 10% increase in error rate over 6 hours, verifying the multi-window alerting catches the slow burn without paging for transient 5-minute blips.

Execution Protocol

ASSURANCE AND ADOPTION

Implementation evidence and review gates



A passing score is not sufficient without current evidence. High-assurance adoption requires traceable design decisions, reproducible builds, independently reviewable tests, controlled secrets, runtime observability, failure exercises, and a dated exception register.

Current authority register

Authority	Version / snapshot	Applicability	Review by
TLA+ Foundation: TLA+ Language and Tools Documentation	Rolling documentation snapshot	High	2027-01-24
Kubernetes Project: Kubernetes Documentation	v1.36 current documentation	High	2026-10-22
OpenTelemetry Project: OpenTelemetry Documentation and Specification	Rolling specification snapshot; profiles public alpha and declarative configuration stable in 2026	High	2026-10-22
Cloud Native Computing Foundation: Cloud Native Landscape and Project Documentation	Rolling catalog snapshot	Moderate	2026-08-23
FinOps Foundation: FinOps Framework and FOCUS Specification	Rolling official documentation snapshot	High	2026-10-22

Research limitation. Official documentation establishes published interfaces and declared behavior. It does not prove the performance, security, interoperability, cost, or operational suitability of a specific implementation. Those claims require controlled testing and retained evidence.

ENGINEERING STANDARDS LIBRARY / RELIABILITY ENGINEERING

Disaster Recovery, Multi-Region, and Failure Testing Standard

Failure-domain architecture, recovery objectives, multi-region design, restore validation, game days, and crisis readiness.

PERMANENT DOCUMENT ID

MYTHOS-SRE-0002

PUBLICATION

Candidate Standard
Version 1.0.0-candidate

ASSURANCE PROFILE

Standard / Advanced
High Assurance

PERMANENT PUBLICATION IDENTITY

Disaster Recovery, Multi-Region, and Failure Testing Standard

DOCUMENT ID
MYTHOS-SRE-0002

VERSION
1.0.0-candidate

STATUS
Candidate Standard

PUBLISHER
Mythos Systems

PUBLICATION DATE
2026-07-24

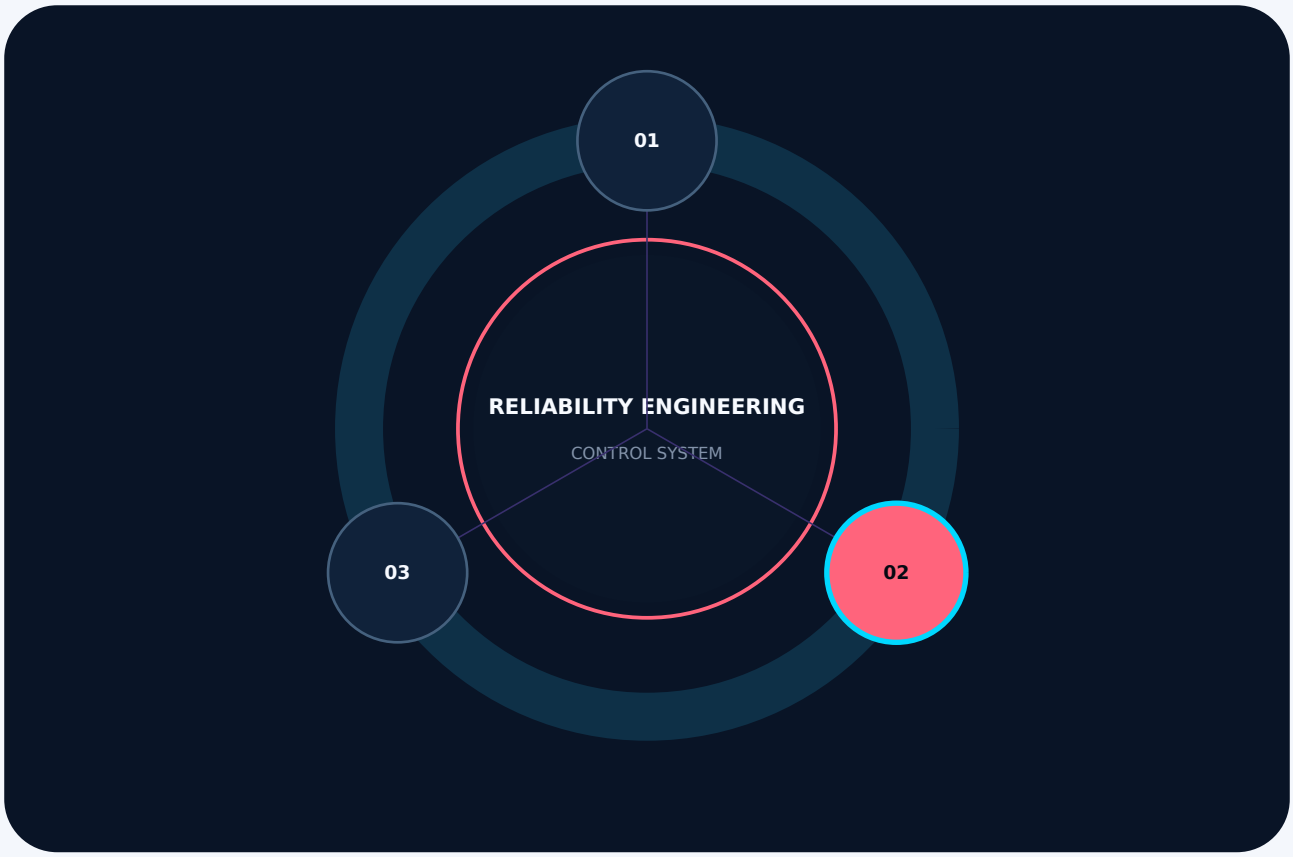
SCHEDULED REVIEW
2027-01-24

11	6	4	1
ATOMIC CRITERIA	ASSURANCE VIEWS	IMPLEMENTATION PROFILES	PERMANENT IDENTITY

Publication doctrine. This publication establishes durable guidance or requirements for its defined scope. Interpretation must preserve the permanent document identity, recorded evidence, controlled exceptions, and formal revision history.

DOMAIN CONTROL SYSTEM

MYTHOS-SRE-0002 inside the reliability engineering standard constellation

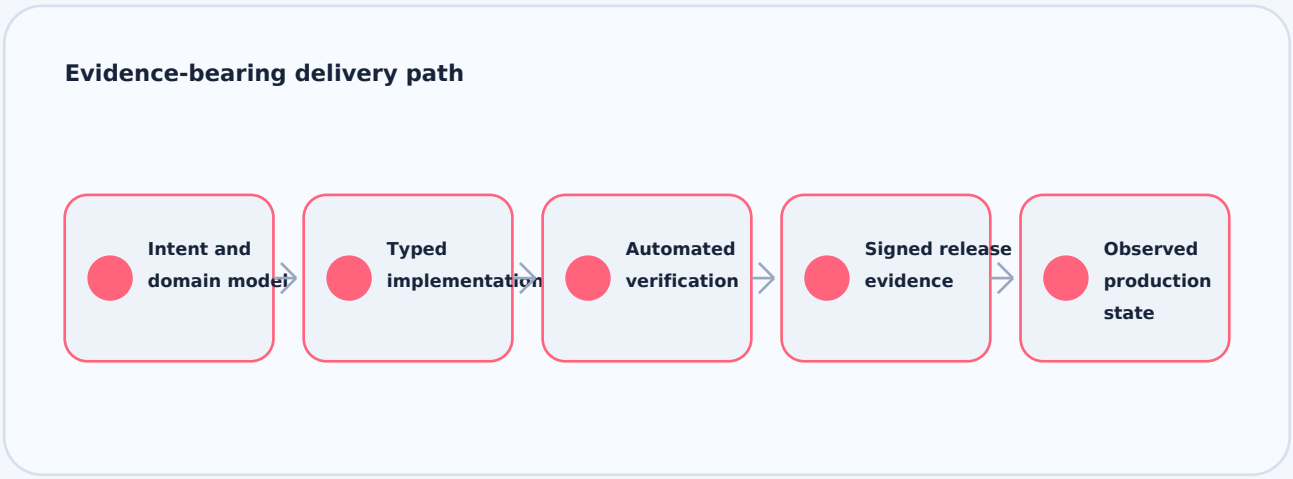


Connected assurance	Architecture, implementation, operations, evidence, and lifecycle are evaluated as one controlled system.
Specialization rule	Companion standards may add precision, but they cannot silently weaken this publication's requirements.

EXECUTIVE POSITION

Disaster Recovery, Multi-Region, and Failure Testing Standard

The architecture of disaster recovery has failed.



CONTENTS

Publication navigation

MYTHOS-SRE-0002 inside the reliability engineering standard constellation . . 3

Disaster Recovery, Multi-Region, and Failure Testing Standard 4

Assurance model and evaluation surface 7

Criteria and evidence map 8

Requirements, evaluation methodology, and implementation guidance 9

Executive Mandate 9

Scope and Applicability 9

In Scope 9

Out of Scope 10

Audience 10

Definitions and Taxonomy 10

DR Dimensions 10

The DR Doctrine 10

Evaluation Methodology 11

Scoring Model 11

Weighting 11

Evaluation Categories 11

Multi-Region Topology & Traffic Shifting (Weight: 20%) 11

Global Routing 11

RTO/RPO Enforcement & State Integrity (Weight: 20%) 12

Data Loss Bounds 12

Chaos Engineering & Fault Injection (Weight: 20%) 12

Resilience Validation 12

Stateful Recovery & Durable Replay (Weight: 15%) 12

Workflow Continuity 12

Blast Radius & Cell-Based Architecture (Weight: 15%) 13

Containment 13

Automation & Runbook Execution (Weight: 10%) 13

Operator Intervention 13

The Mythos DR Suite (MDRS) 13

Suite Composition 14

MDRS-Blackout 14

MDRS-Chaos 14

Execution Protocol 14

Implementation evidence and review gates 15

Current authority register 16

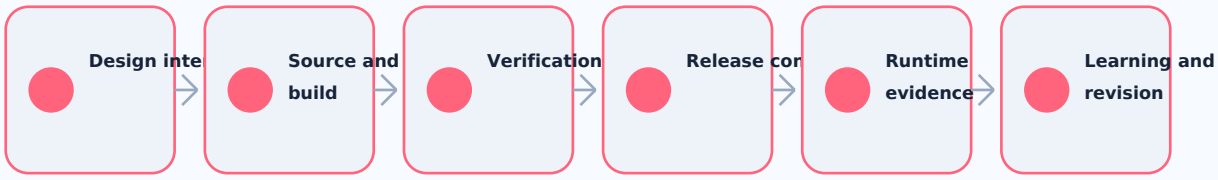
CONTROL TOPOLOGY

Assurance model and evaluation surface

The standard is assessed as an evidence-bearing system. Capability scores do not replace mandatory gates, implementation proof, or operational validation.

Capability family	Evaluation nodes
Multi-Region Topology & Traffic Shifting (Weight: 20%)	1
RTO/RPO Enforcement & State Integrity (Weight: 20%)	1
Chaos Engineering & Fault Injection (Weight: 20%)	1
Stateful Recovery & Durable Replay (Weight: 15%)	1
Blast Radius & Cell-Based Architecture (Weight: 15%)	1
Automation & Runbook Execution (Weight: 10%)	1
Suite Composition	2
Additional Evaluation Dimension	3

Lifecycle control loop



EVALUATION INDEX

Criteria and evidence map

#	Capability family	Criterion
1	Multi-Region Topology & Traffic Shifting (Weight: 20%)	Global Routing
2	RTO/RPO Enforcement & State Integrity (Weight: 20%)	Data Loss Bounds
3	Chaos Engineering & Fault Injection (Weight: 20%)	Resilience Validation
4	Stateful Recovery & Durable Replay (Weight: 15%)	Workflow Continuity
5	Blast Radius & Cell-Based Architecture (Weight: 15%)	Containment
6	Automation & Runbook Execution (Weight: 10%)	Operator Intervention
7	Suite Composition	MDRS-Blackout
8	Suite Composition	MDRS-Chaos
9	Additional Evaluation Dimension	DR Dimensions
10	Additional Evaluation Dimension	The DR Doctrine
11	Additional Evaluation Dimension	Scoring Model

NORMATIVE PUBLICATION

Requirements, evaluation methodology, and implementation guidance

The following body preserves the substantive source standard while normalizing publication status, identity, navigation, and visual presentation.

Executive Mandate

The architecture of disaster recovery has failed.

Organizations write 100-page DR runbooks that are never executed until a catastrophic outage occurs. When the cloud provider's primary region goes down, they attempt to failover to the passive region, only to discover the DNS TTL is too high, the database replicas are 5 hours stale, and the application cannot start because it hardcodes the primary region's S3 bucket name.

This standard exists because:

1. An Untested DR Plan is Fiction: A DR strategy that relies on a PDF document and manual operator intervention during a 3 AM crisis is guaranteed to fail. DR plans MUST be continuously automated and tested in production via chaos engineering.
2. Active-Passive is a Legacy Crutch: A hot-standby region that sits idle is a waste of capital and a reliability risk. When failover occurs, the passive region is instantly overwhelmed by production traffic it has never handled. Architectures MUST be active-active, serving live traffic from all regions simultaneously.
3. RTO/RPO Must Be Mathematically Bounded, Not Hoped For: "We aim for zero data loss" is a wish, not an architecture. Recovery Time Objective (RTO) and Recovery Point Objective (RPO) MUST be mathematically enforced by synchronous replication, automated traffic shifting, and durable execution runtimes.
4. Stateful Failover is the Hardest Problem: Stateless compute fails over easily; databases do not. Relying on asynchronous database replication guarantees data loss (RPO > 0). Systems MUST utilize distributed SQL, CRDTs, or event sourcing to ensure state convergence across regions without split-brain.

This document establishes the Mythos DR & Resilience Standard (MDRS), a comprehensive, evidence-based methodology for evaluating disaster recovery architectures against active-active topology, chaos readiness, and mathematically proven state integrity.

Scope and Applicability

In Scope

This standard covers the evaluation of:

1. Multi-region architectures (Active-Active vs. Active-Passive)

1. Recovery Time Objective (RTO) and Recovery Point Objective (RPO) enforcement
1. Global traffic management and DNS failover (Route 53, Cloudflare)
1. Chaos engineering and fault injection (Gremlin, Chaos Mesh)
1. Stateful DR: Distributed SQL, synchronous replication, CRDTs
1. Durable execution replay and workflow recovery

Out of Scope

1. General SLO definitions and alerting (covered in MYTHOS-SRE-0001)
1. General event sourcing patterns (covered in MYTHOS-DAT-0004)
1. General database high availability (covered in MYTHOS-DBS-0001)

Audience

1. Site Reliability Engineers (SREs) and platform architects
1. Distributed systems engineers building multi-region services
1. Security engineers evaluating ransomware regional isolation
1. Compliance teams managing RTO/RPO mandates
1. Standards bodies seeking a reference DR methodology

Definitions and Taxonomy

DR Dimensions

Dimension	Definition
RTO (Recovery Time Objective)	The maximum acceptable time between a disaster and the restoration of service. Enforced by automated traffic shifting and infrastructure provisioning.
RPO (Recovery Point Objective)	The maximum acceptable amount of data loss, measured in time. Enforced by synchronous replication or durable event sourcing.
Active-Active Multi-Region	An architecture where multiple geographic regions simultaneously serve live production traffic and replicate state bi-directionally.
Chaos Engineering	The discipline of experimenting on a system by injecting failures (e.g., killing pods, severing network links) in production to verify resilience and validate DR assumptions.
Split-Brain	A failure scenario where a network partition causes two regions to both assume they are the primary write master, resulting in irreconcilable data conflicts.
CRDT (Conflict-free Replicated Data Type)	A data structure that can be replicated across multiple networked computers, updated independently and concurrently, and mathematically resolved without conflicts.

The DR Doctrine

> An untested DR plan is fiction. Active-passive is a waste of capital. An RPO of zero requires synchronous math, not hope. If you are not injecting failures in production, you are flying blind.

Evaluation Methodology

Scoring Model

Each capability is scored from 0 to 5.

Score	Meaning
0	Fails basic task; single region, paper DR plan, async DB replication, no chaos
1	Basic backups to another region, but manual restore required
2	Functional active-passive, but RPO > 0 and failover is manual/scripted
3	Solid production performance; Active-Active, RTO/RPO bounded, automated failover, chaos in staging
4	Strong performance; RPO=0 via sync replication, chaos in prod, CRDT/distributed SQL state
5	Best-in-class; sets the standard for mathematically verified, autonomous regional resilience

Weighting

Category	Weight	Rationale
Multi-Region Topology & Traffic Shifting	20%	Active-passive fails under sudden load shifts; DNS TTLs cause outages
RTO/RPO Enforcement & State Integrity	20%	Async replication guarantees data loss; RPO=0 requires synchronous math
Chaos Engineering & Fault Injection	20%	Untested failover mechanisms guarantee failure during a real disaster
Stateful Recovery & Durable Replay	15%	Stateless compute is easy; database failover is the true test
Blast Radius & Cell-Based Architecture	15%	A failure in one region or service must not cascade globally
Automation & Runbook Execution	10%	Manual operator intervention during a crisis is too slow

Total: 100%

A system MUST score at least 3.0 in Multi-Region Topology and Chaos Engineering to be considered for production use.

Evaluation Categories

Multi-Region Topology & Traffic Shifting (Weight: 20%)

Global Routing

How is traffic routed and failed over between regions?

Score	Criteria
0	Single region; no failover capability
1	Active-Passive; manual DNS updates required to failover (RTO > 1 hour)

Score	Criteria
2	Automated DNS failover, but high TTLs (minutes) cause partial outages
3	Active-Active: both regions serve live traffic; global load balancer routes based on latency/health
4	Sub-second traffic shifting via BGP Anycast or global mesh routing
5	Perfect topology: formally verified routing bounds, zero DNS caching delays, seamless regional evacuation

RTO/RPO Enforcement & State Integrity (Weight: 20%)

Data Loss Bounds

Are RTO and RPO mathematically guaranteed, or just hoped for?

Score	Criteria
0	Async DB replication; RPO is "whatever the replication lag is" (data loss guaranteed)
1	Periodic snapshots to backup region; RPO > 1 hour
2	Near-real-time replication, but accepts data loss on failover
3	Synchronous replication ensuring RPO = 0 (zero data loss)
4	Distributed SQL (CockroachDB/Spanner) or CRDTs providing globally consistent state
5	Perfect integrity: formally verified consensus algorithms, zero split-brain risk, mathematical proof of state parity

Chaos Engineering & Fault Injection (Weight: 20%)

Resilience Validation

Is the DR plan continuously tested via failure injection?

Score	Criteria
0	No testing; DR plan exists only in a document
1	Failover tested manually once a year
2	Chaos engineering in staging environment
3	Chaos engineering in production (e.g., killing pods, severing network links)
4	Automated Game Days simulating full regional blackouts
5	Perfect validation: formally verified blast radius bounds, continuous autonomous failure injection, mathematical proof of self-healing

Stateful Recovery & Durable Replay (Weight: 15%)

Workflow Continuity

Can long-running workflows and agent tasks survive a regional failure?

Score	Criteria
0	In-memory workflows; state lost on regional crash

Score	Criteria
1	Checkpoints exist, but manual restart required
2	Workflows recover, but side-effects may be re-executed (violating idempotency)
3	Durable execution runtime (Temporal) with global state store; workflows replay deterministically
4	Multi-region durable runtime; workflows seamlessly migrate to healthy region
5	Perfect continuity: formally verified deterministic replay, zero side-effect duplication, sub-second state migration

Blast Radius & Cell-Based Architecture (Weight: 15%)

Containment

Can a failure in one service or region be prevented from cascading globally?

Score	Criteria
0	Shared global dependency (e.g., single global database); failure cascades everywhere
1	Basic namespace isolation, but shared control plane
2	Regional isolation for compute, but shared global state
3	Cell-based architecture: each region/cell is fully self-contained and independent
4	Bulkheads and circuit breakers preventing cross-cell cascading failures
5	Perfect containment: formally verified blast radius limits, zero cross-regional cascade potential

Automation & Runbook Execution (Weight: 10%)

Operator Intervention

How much human intervention is required during a regional disaster?

Score	Criteria
0	100% manual; engineers must SSH into boxes and run scripts
1	Runbooks exist, but require manual step-by-step execution
2	Single-click failover scripts, but require human authorization
3	Fully automated failover triggered by health checks; humans only observe
4	Autonomous failover with automated rollback if the recovery fails
5	Perfect automation: formally verified runbook execution, zero human latency in the recovery path

The Mythos DR Suite (MDRS)

Traditional DR benchmarks measure backup restore speed. The MDRS evaluates regional blackout survival, state convergence, and zero-downtime traffic shifting.

Suite Composition

MDRS-Blackout

1. Region Kill Test: 100+ tests simulating a total AWS region failure (cutting all network and API access), verifying traffic shifts to the secondary region in < 60 seconds (RTO).
1. State Parity Test: 50+ tests verifying that the secondary region has zero data loss (RPO = 0) after the failover.

MDRS-Chaos

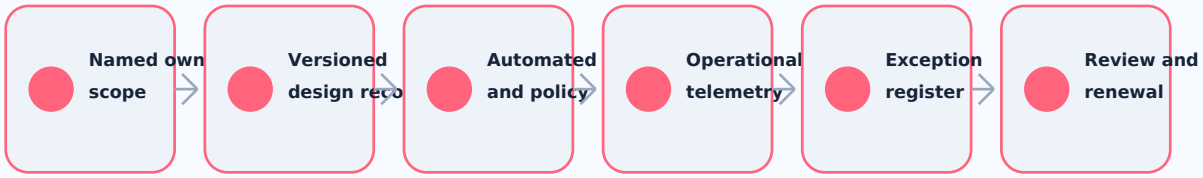
1. Latency Injection Test: 100+ tests injecting 500ms network latency between regions, verifying the distributed SQL/CRDT state converges without timing out.
1. Pod Destruction Test: 50+ tests randomly killing 30% of stateless pods in production, verifying the auto-scaler and mesh reroute traffic without breaching SLOs.

Execution Protocol

ASSURANCE AND ADOPTION

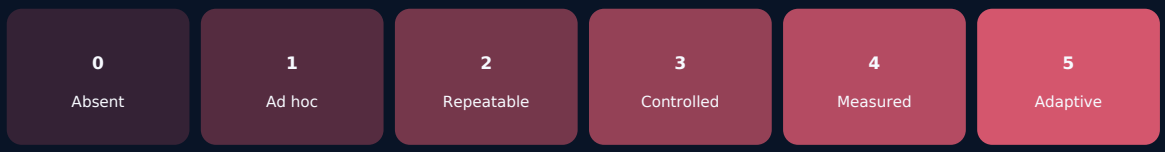
Implementation evidence and review gates

Required assurance chain



A passing score is not sufficient without current evidence. High-assurance adoption requires traceable design decisions, reproducible builds, independently reviewable tests, controlled secrets, runtime observability, failure exercises, and a dated exception register.

Assurance maturity spectrum



Current authority register

Authority	Version / snapshot	Applicability	Review by
TLA+ Foundation: TLA+ Language and Tools Documentation	Rolling documentation snapshot	High	2027-01-24
Kubernetes Project: Kubernetes Documentation	v1.36 current documentation	High	2026-10-22
OpenTelemetry Project: OpenTelemetry Documentation and Specification	Rolling specification snapshot; profiles public alpha and declarative configuration stable in 2026	High	2026-10-22
Cloud Native Computing Foundation: Cloud Native Landscape and Project Documentation	Rolling catalog snapshot	Moderate	2026-08-23
FinOps Foundation: FinOps Framework and FOCUS Specification	Rolling official documentation snapshot	High	2026-10-22

Research limitation. Official documentation establishes published interfaces and declared behavior. It does not prove the performance, security, interoperability, cost, or operational suitability of a specific implementation. Those claims require controlled testing and retained evidence.

ENGINEERING STANDARDS LIBRARY / RELIABILITY ENGINEERING

FinOps, Token Economics, and Energy Efficiency Standard

Unit economics, token and compute cost, capacity efficiency, carbon-aware operation, and value-based engineering governance.

PERMANENT DOCUMENT ID

MYTHOS-SRE-0003

PUBLICATION

Candidate Standard
Version 1.0.0-candidate

ASSURANCE PROFILE

Standard / Advanced
High Assurance

PERMANENT PUBLICATION IDENTITY

FinOps, Token Economics, and Energy Efficiency Standard

DOCUMENT ID
MYTHOS-SRE-0003

VERSION
1.0.0-candidate

STATUS
Candidate Standard

PUBLISHER
Mythos Systems

PUBLICATION DATE
2026-07-24

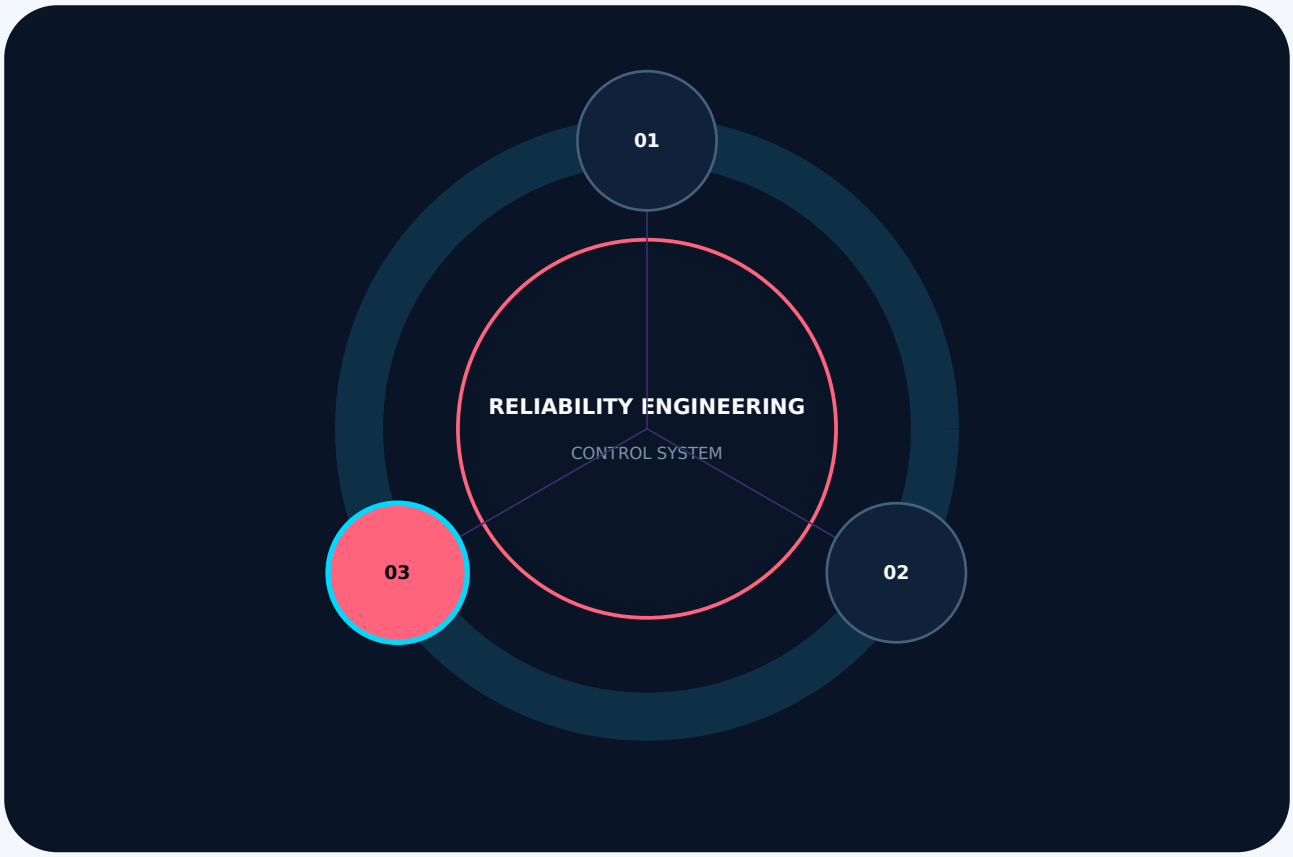
SCHEDULED REVIEW
2027-01-24

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

Publication doctrine. This publication establishes durable guidance or requirements for its defined scope. Interpretation must preserve the permanent document identity, recorded evidence, controlled exceptions, and formal revision history.

DOMAIN CONTROL SYSTEM

MYTHOS-SRE-0003 inside the reliability engineering standard constellation



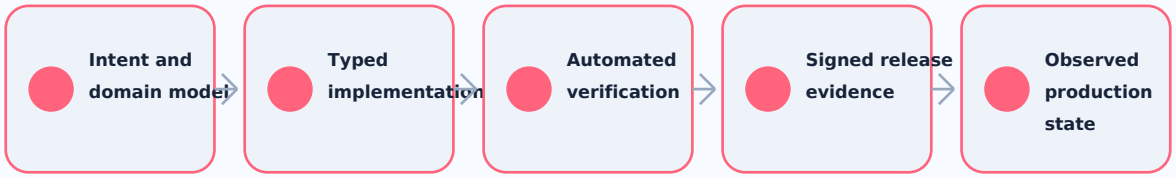
Connected assurance	Architecture, implementation, operations, evidence, and lifecycle are evaluated as one controlled system.
Specialization rule	Companion standards may add precision, but they cannot silently weaken this publication's requirements.

EXECUTIVE POSITION

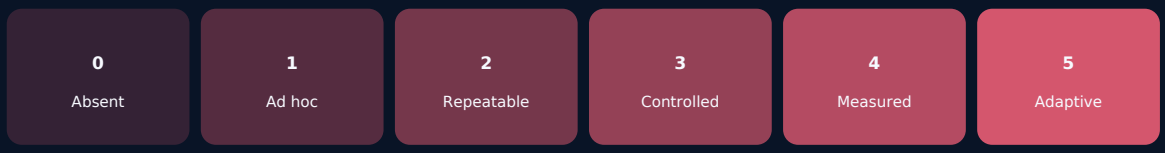
FinOps, Token Economics, and Energy Efficiency Standard

The architecture of cloud economics and energy management has failed.

Evidence-bearing delivery path



Assurance maturity spectrum



CONTENTS

Publication navigation

MYTHOS-SRE-0003 inside the reliability engineering standard constellation . . 3

FinOps, Token Economics, and Energy Efficiency Standard 4

Assurance model and evaluation surface 7

Criteria and evidence map 8

Requirements, evaluation methodology, and implementation guidance 9

Executive Mandate 9

Scope and Applicability 9

In Scope 9

Out of Scope 10

Audience 10

Definitions and Taxonomy 10

Economics & Energy Dimensions 10

The FinOps & Energy Doctrine 10

Evaluation Methodology 11

Scoring Model 11

Weighting 11

Evaluation Categories 11

Cost Attribution and Tagging (Weight: 15%) 11

Financial Observability 11

AI Token Budgets and Circuit Breakers (Weight: 20%) 12

Agent Economic Control 12

Scale-to-Zero and Utilization (Weight: 20%) 12

Idle Compute Elimination 12

Energy Efficiency (SCI/PUE) (Weight: 15%) 13

Sustainability Metrics 13

Carbon-Aware Workload Routing (Weight: 15%) 13

Green Compute Placement 13

Automated Cost Controls (Policy-as-Code) (Weight: 15%) 13

Guardrail Enforcement 13

The Mythos FinOps & Energy Suite (MFEES) 14

Suite Composition 14

MFEES-Token 14

MFEES-Energy 14

Execution Protocol 14

Implementation evidence and review gates 15

Current authority register 16

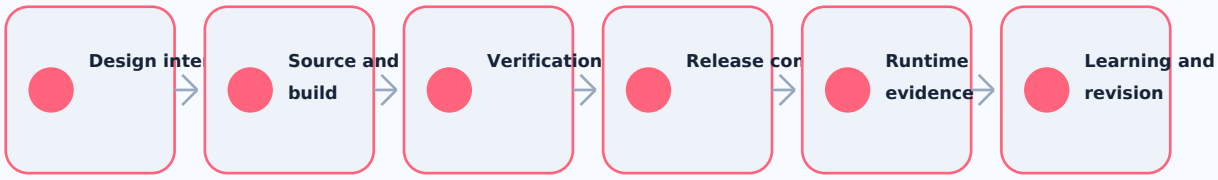
CONTROL TOPOLOGY

Assurance model and evaluation surface

The standard is assessed as an evidence-bearing system. Capability scores do not replace mandatory gates, implementation proof, or operational validation.

Capability family	Evaluation nodes
Cost Attribution and Tagging (Weight: 15%)	1
AI Token Budgets and Circuit Breakers (Weight: 20%)	1
Scale-to-Zero and Utilization (Weight: 20%)	1
Energy Efficiency (SCI/PUE) (Weight: 15%)	1
Carbon-Aware Workload Routing (Weight: 15%)	1
Automated Cost Controls (Policy-as-Code) (Weight: 15%)	1
Suite Composition	2
Additional Evaluation Dimension	4

Lifecycle control loop



EVALUATION INDEX

Criteria and evidence map

#	Capability family	Criterion
1	Cost Attribution and Tagging (Weight: 15%)	Financial Observability
2	AI Token Budgets and Circuit Breakers (Weight: 20%)	Agent Economic Control
3	Scale-to-Zero and Utilization (Weight: 20%)	Idle Compute Elimination
4	Energy Efficiency (SCI/PUE) (Weight: 15%)	Sustainability Metrics
5	Carbon-Aware Workload Routing (Weight: 15%)	Green Compute Placement
6	Automated Cost Controls (Policy-as-Code) (Weight: 15%)	Guardrail Enforcement
7	Suite Composition	MFEES-Token
8	Suite Composition	MFEES-Energy
9	Additional Evaluation Dimension	Economics & Energy Dimensions
10	Additional Evaluation Dimension	The FinOps & Energy Doctrine
11	Additional Evaluation Dimension	Scoring Model
12	Additional Evaluation Dimension	Suite Composition

NORMATIVE PUBLICATION

Requirements, evaluation methodology, and implementation guidance

The following body preserves the substantive source standard while normalizing publication status, identity, navigation, and visual presentation.

Executive Mandate

The architecture of cloud economics and energy management has failed.

Organizations migrate to the cloud and deploy autonomous AI agents under the assumption that compute is infinite and cheap. They allow an agent to enter a recursive loop, burning $50,000 in LLM API tokens in a weekend. They leave hundreds of idle GPU instances running 24/7 because scaling policies are misconfigured. They measure success by application latency, completely ignoring the massive carbon footprint and wasted capital of inefficient code.

This standard exists because:

1. A Cloud Bill is a Lagging Indicator of Architectural Failure: If you discover you spent $100,000 on untagged, untracked EC2 instances at the end of the month, the damage is done. Cost tracking MUST be real-time, attributed to specific domains/agents, and actionable.
2. AI Tokens are a Finite Economic Resource: LLM inference is not a flat-rate utility; it is a highly variable, token-based economy. An autonomous agent without a hard token budget is a financial liability. Systems MUST enforce per-task and per-agent token ceilings that automatically suspend execution when exceeded.
3. Idle Compute is Environmental Malpractice: Running a datacenter at 20% utilization wastes 80% of the power and cooling. Systems MUST implement aggressive scale-to-zero architectures and serverless paradigms for non-continuous workloads.
4. Efficiency Must Be Measured in Carbon, Not Just Dollars: As AI compute scales, its energy footprint becomes a critical operational and regulatory constraint. Systems MUST calculate Software Carbon Intensity (SCI) and prioritize workload placement in low-carbon-intensity regions.

This document establishes the Mythos FinOps & Energy Standard (MFEES), a comprehensive, evidence-based methodology for evaluating financial operations against real-time cost enforcement, token economics, and green software engineering.

Scope and Applicability

In Scope

This standard covers the evaluation of:

1. FinOps platforms and cost attribution (CloudHealth, Apptio, native cloud billing)
1. AI token economics, budgeting, and circuit breakers

- Green Software Foundation metrics (Software Carbon Intensity - SCI)
- Datacenter PUE (Power Usage Effectiveness) and DVFS (Dynamic Voltage/Frequency Scaling)
- Scale-to-zero architectures and serverless cost optimization
- Carbon-aware workload routing and placement

Out of Scope

- General SLO and error budget tracking (covered in MYTHOS-SRE-0001)
- General GPU hardware topologies (covered in MYTHOS-HW-0001)
- General inference serving optimization (covered in MYTHOS-AI-0014)

Audience

- FinOps practitioners and cloud financial analysts
- Platform engineers building autoscaling and scale-to-zero pipelines
- AI engineers building token-budgeted agent loops
- SREs managing datacenter power and cooling envelopes
- Sustainability officers tracking carbon footprints
- Standards bodies seeking a reference FinOps methodology

Definitions and Taxonomy

Economics & Energy Dimensions

Dimension	Definition
FinOps	The operational framework and cultural practice of maximizing business value in the cloud by bringing financial accountability to variable spend.
Token Economics	The tracking, budgeting, and enforcement of LLM inference costs, calculated by multiplying input/output tokens by the provider's per-token pricing model.
Software Carbon Intensity (SCI)	A metric defined by the Green Software Foundation representing the carbon emissions per functional unit of software (e.g., gCO2e per user request).
Scale-to-Zero	An architectural paradigm where compute resources provision dynamically upon request and deallocate entirely to zero instances during periods of inactivity.
Carbon-Aware Routing	The practice of dynamically shifting compute workloads to datacenter regions where the local power grid is currently powered by renewable (low-carbon) energy.

The FinOps & Energy Doctrine

> A cloud bill is a lagging indicator of failure. An AI agent without a token budget is a financial liability. An idle server is environmental malpractice. If the architecture is not carbon-aware, it is unsustainable.

Evaluation Methodology

Scoring Model

Each capability is scored from 0 to 5.

Score	Meaning
0	Fails basic task; untagged resources, monthly bill shock, idle compute, no token limits
1	Basic cost alerts, but manual tagging and no token budgets
2	Functional FinOps tracking, but lacks real-time enforcement or energy tracking
3	Solid production performance; Real-time domain attribution, hard token budgets, scale-to-zero, SCI tracking
4	Strong performance; Carbon-aware routing, AI-driven cost anomaly detection, formally verified budgets
5	Best-in-class; sets the standard for mathematically verified, zero-waste compute economics

Weighting

Category	Weight	Rationale
Cost Attribution & Tagging	15%	Untagged resources cannot be optimized or charged back
AI Token Budgets & Circuit Breakers	20%	Unbudgeted agents guarantee catastrophic financial surprises
Scale-to-Zero & Utilization	20%	Idle compute wastes capital and energy
Energy Efficiency (SCI/PUE)	15%	Datacenter power consumption is a hard physical limit
Carbon-Aware Workload Routing	15%	AI compute must follow renewable energy availability
Automated Cost Controls (Policy-as-Code)	15%	Manual cost optimization is reactionary and too slow

Total: 100%

A system MUST score at least 3.0 in AI Token Budgets and Scale-to-Zero to be considered for production use.

Evaluation Categories

Cost Attribution and Tagging (Weight: 15%)

Financial Observability

Can the organization attribute every dollar spent to a specific domain, team, or agent?

Score	Criteria
0	Single billing account; no tags; "shared infrastructure" black box

Score	Criteria
1	Manual tagging enforced via policy, but compliance is < 80%
2	Automated tagging on all resources, but cost allocation is delayed (monthly)
3	Real-time cost attribution mapped to DDD bounded contexts; per-domain chargeback models
4	Per-request/per-agent cost tracking (e.g., tracing the exact cloud cost of a single user transaction)
5	Perfect attribution: formally verified cost provenance, zero untagged resources, mathematical proof of domain-level financial parity

AI Token Budgets and Circuit Breakers (Weight: 20%)

Agent Economic Control

How is the cost of autonomous AI agents bounded?

Score	Criteria
0	No token tracking; agents run until they finish or crash, regardless of cost
1	Total token count logged per request, but no cost calculation
2	Cost calculated per request, but no hard budgets enforced
3	Hard token/cost budgets enforced per agent task; tasks are automatically suspended if budgets are exceeded
4	Predictive token budgeting: system estimates task cost before execution and asks for HITL approval if > $X
5	Perfect control: formally verified economic bounds, zero ability for runaway agents to cause bill shock, mathematical proof of cost efficiency

Scale-to-Zero and Utilization (Weight: 20%)

Idle Compute Elimination

How does the system handle periods of low or zero traffic?

Score	Criteria
0	Instances run 24/7/365 regardless of traffic; average utilization < 15%
1	Basic autoscaling, but minimum instance count is > 0
2	Scale-to-zero implemented for dev/staging, but production runs idle instances
3	Aggressive scale-to-zero for all non-critical paths; serverless (FaaS) utilized for bursty workloads
4	GPU scale-to-zero implemented with cold-start mitigation (e.g., keeping model weights in memory while suspending compute)
5	Perfect utilization: formally verified zero-idle waste, mathematical proof of optimal provision-to-request latency ratios

Energy Efficiency (SCI/PUE) (Weight: 15%)

Sustainability Metrics

Is the energy consumption and carbon footprint of software actively measured?

Score	Criteria
0	No energy or carbon tracking; PUE unknown
1	Datacenter PUE tracked, but no software-level metrics
2	Cloud provider carbon dashboards reviewed monthly
3	Software Carbon Intensity (SCI) score calculated per major workflow; DVFS enabled on hardware
4	Real-time SCI tracking integrated into observability dashboards alongside latency and cost
5	Perfect measurement: formally verified energy bounds, mathematical proof of minimal carbon per functional unit

Carbon-Aware Workload Routing (Weight: 15%)

Green Compute Placement

Does the system shift compute based on renewable energy availability?

Score	Criteria
0	Workloads run in a single region regardless of carbon intensity
1	Carbon intensity awareness exists, but requires manual routing
2	Batch jobs (e.g., AI training) scheduled to run during off-peak energy hours
3	Carbon-aware routing: batch workloads dynamically routed to regions with real-time surplus renewable energy (e.g., wind/solar peaks)
4	Real-time inference routing adjusted based on grid carbon intensity without violating SLOs
5	Perfect routing: formally verified carbon optimization, zero fossil-fuel-bound compute where renewables are available, mathematical proof of minimal carbon footprint

Automated Cost Controls (Policy-as-Code) (Weight: 15%)

Guardrail Enforcement

are financial guardrails enforced in the CI/CD pipeline?

Score	Criteria
0	No automated controls; engineers can provision $10,000/day GPU instances without approval
1	Basic cloud budget alerts sent to Slack
2	Hard spending limits configured in the cloud account, but blunt (blocks all provisioning)
3	Policy-as-Code (OPA/Sentinel) in CI/CD blocking the deployment of expensive instance types without peer review

Score	Criteria
4	Real-time anomaly detection: system automatically suspends workloads exhibiting cost spikes indicative of infinite loops
5	Perfect controls: formally verified financial guardrails, zero ability to bypass budget policies, mathematical proof of spend compliance

The Mythos FinOps & Energy Suite (MFEES)

Traditional FinOps reviews measure monthly bill reductions. The MFEES evaluates real-time token enforcement, scale-to-zero resilience, and carbon-aware routing.

Suite Composition

MFEES-Token

1. Runaway Agent Test: 100+ tests simulating an AI agent stuck in an infinite tool-calling loop, verifying the token circuit breaker suspends the task before exceeding the $10 budget.
1. Cost Attribution Test: 50+ tests verifying that the exact compute and token costs of a specific API request are tagged and attributable to the triggering user/domain.

MFEES-Energy

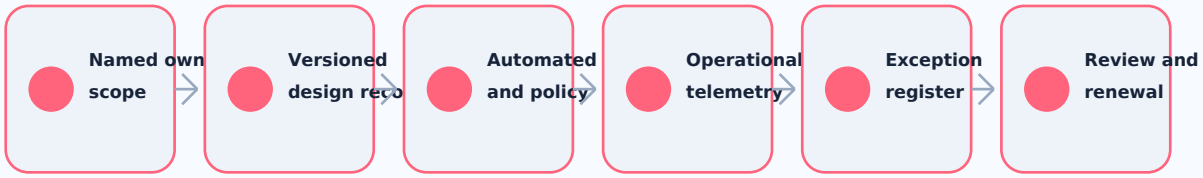
1. Scale-to-Zero Test: 100+ tests simulating 1 hour of zero traffic, verifying all non-critical compute instances deallocate to zero and stop accruing costs.
1. Carbon Routing Test: 50+ tests simulating a surge in renewable energy in Region B, verifying batch training workloads migrate from Region A to Region B automatically.

Execution Protocol

ASSURANCE AND ADOPTION

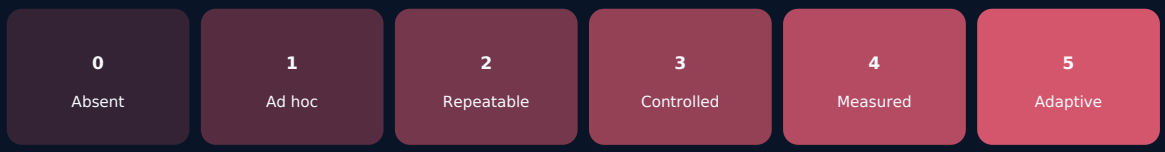
Implementation evidence and review gates

Required assurance chain



A passing score is not sufficient without current evidence. High-assurance adoption requires traceable design decisions, reproducible builds, independently reviewable tests, controlled secrets, runtime observability, failure exercises, and a dated exception register.

Assurance maturity spectrum



Current authority register

Authority	Version / snapshot	Applicability	Review by
TLA+ Foundation: TLA+ Language and Tools Documentation	Rolling documentation snapshot	High	2027-01-24
Kubernetes Project: Kubernetes Documentation	v1.36 current documentation	High	2026-10-22
OpenTelemetry Project: OpenTelemetry Documentation and Specification	Rolling specification snapshot; profiles public alpha and declarative configuration stable in 2026	High	2026-10-22
Cloud Native Computing Foundation: Cloud Native Landscape and Project Documentation	Rolling catalog snapshot	Moderate	2026-08-23
FinOps Foundation: FinOps Framework and FOCUS Specification	Rolling official documentation snapshot	High	2026-10-22

Research limitation. Official documentation establishes published interfaces and declared behavior. It does not prove the performance, security, interoperability, cost, or operational suitability of a specific implementation. Those claims require controlled testing and retained evidence.