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

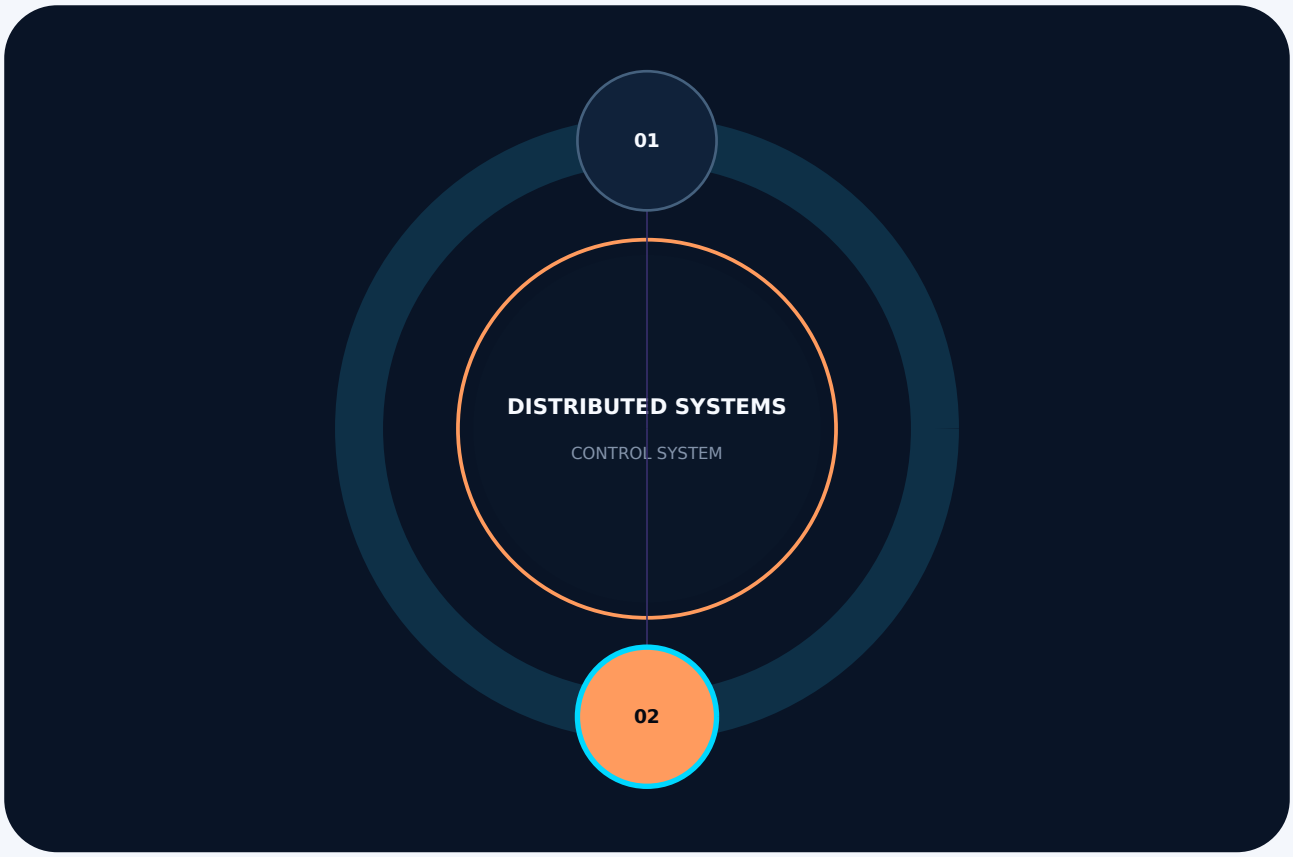
SCHEDULED REVIEW
2027-01-24

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



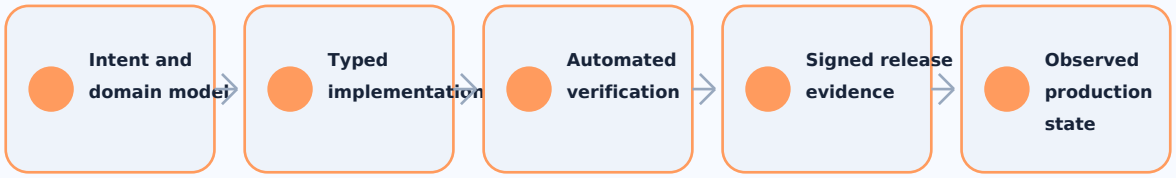
Connected assurance	Architecture, implementation, operations, evidence, and lifecycle are evaluated as one controlled system.
Specialization rule	Companion standards may add precision, but they cannot silently weaken this publication's requirements.

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

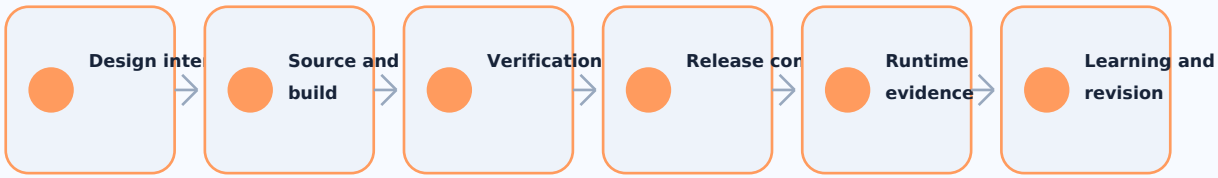
CONTROL TOPOLOGY

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

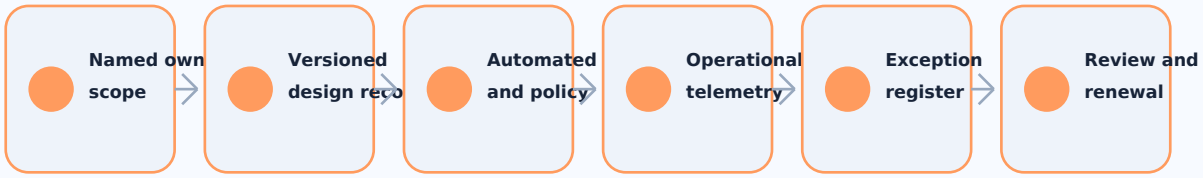
1. Plaintext Injection: 100+ tests attempting to bypass the sidecar and call a downstream service via plaintext HTTP, verifying the service refuses the connection.
1. Authz Bypass Test: 50+ tests attempting to call a restricted gRPC method (`DeleteUser`) from an unauthorized service, verifying the OPA/Envoy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