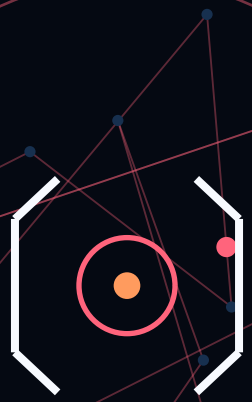


ENGINEERING STANDARDS LIBRARY / RELIABILITY ENGINEERING

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

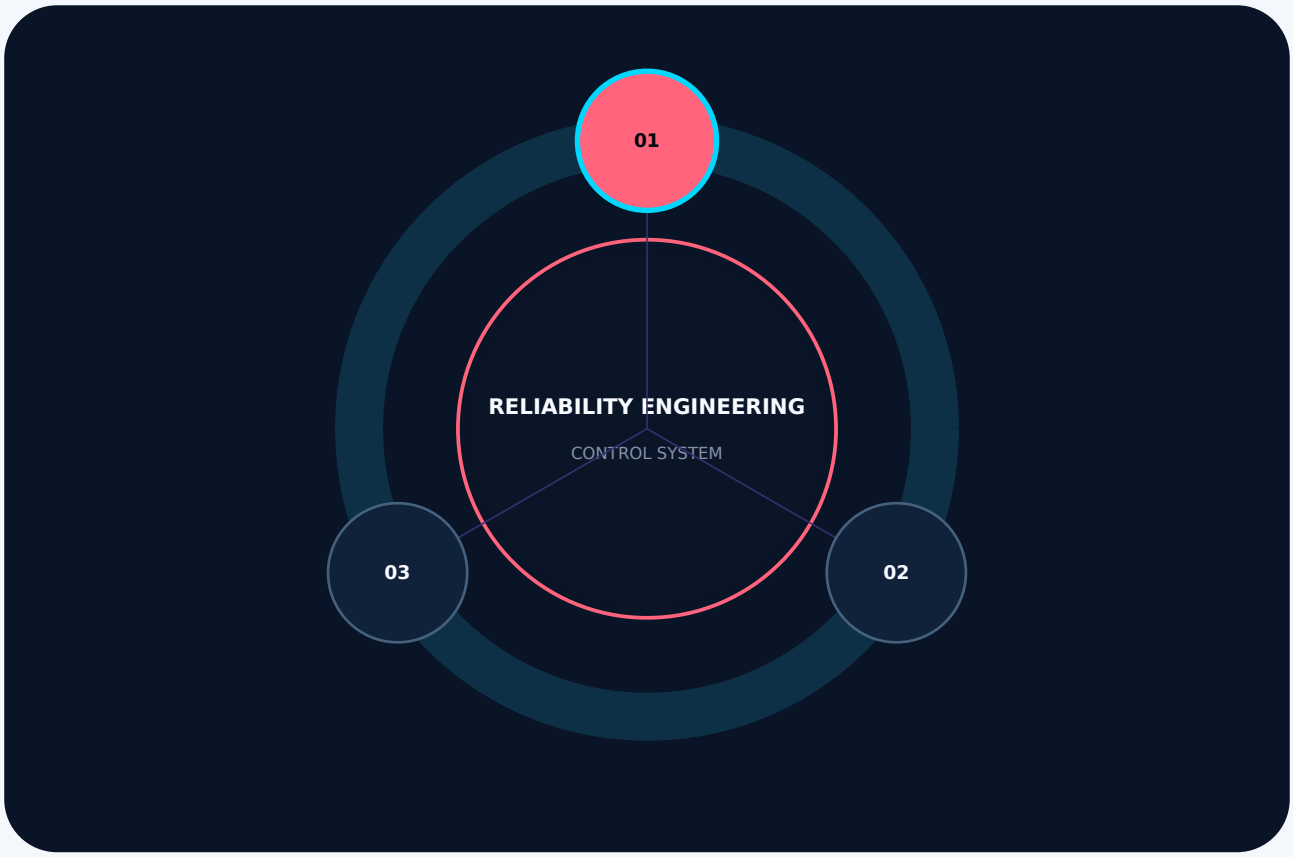
SCHEDULED REVIEW
2027-01-24

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



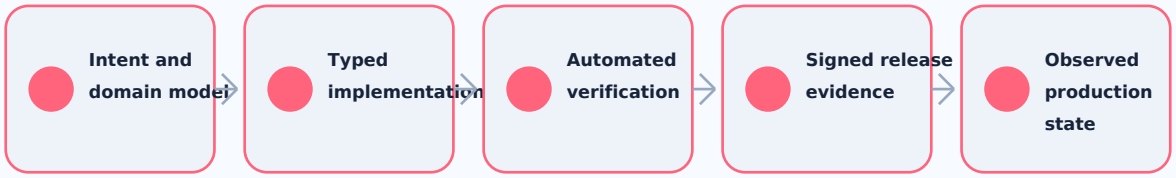
Connected assurance	Architecture, implementation, operations, evidence, and lifecycle are evaluated as one controlled system.
Specialization rule	Companion standards may add precision, but they cannot silently weaken this publication's requirements.

EXECUTIVE POSITION

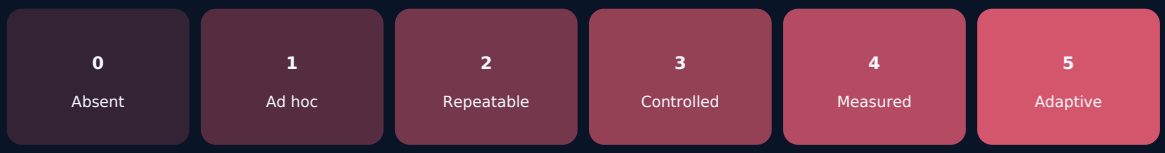
SLOs, Error Budgets, and Capacity Planning Standard

The architecture of reliability and capacity management has failed.

Evidence-bearing delivery path



Assurance maturity spectrum



CONTENTS

Publication navigation

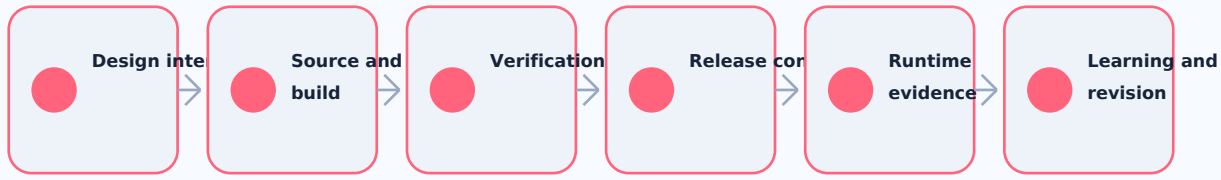
CONTROL TOPOLOGY

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.

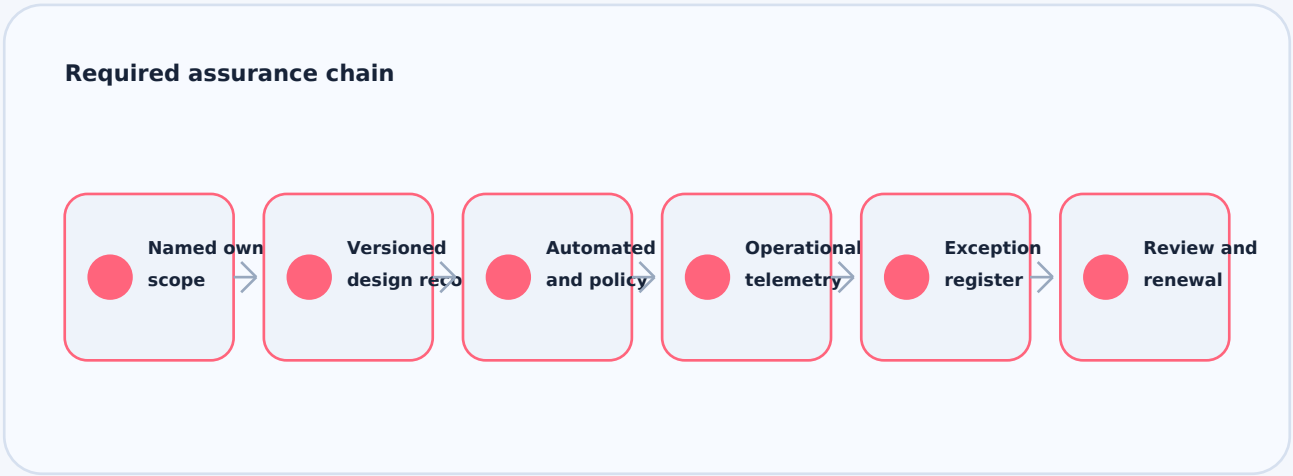
MSRES-Capacity

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.