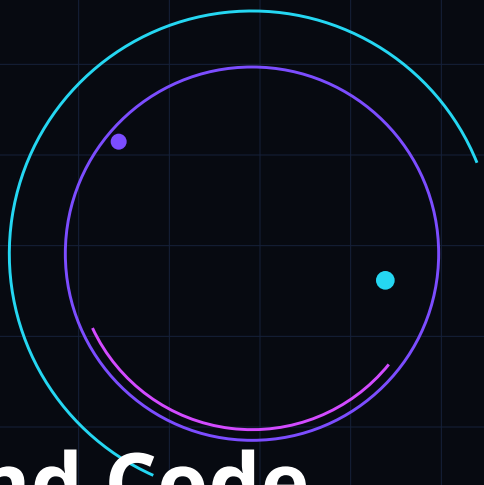
MYTHOS TECHNICAL LIBRARY

STANDARDS & BENCHMARKS / MYTHOS-AI



AI Software Engineer and Code Generation Benchmark Standard

Defines repository-level code generation, repair, review, test, security, debugging, provenance, and completion benchmarks.

MYTHOS-AI-0004

Candidate Mythos Standard / 2.0.0-candidate

Published 2026-07-26 | Review 2026-10-26 | Public

Mythos Systems

Permanent Identity and Edition Record

Permanent document ID	MYTHOS-AI-0004
Canonical title	AI Software Engineer and Code Generation Benchmark Standard
Publication class	Standards & Benchmarks
Primary domain	MYTHOS-AI - Artificial Intelligence & Agentic Systems
Version	2.0.0-candidate
Status	Candidate Mythos Standard
Publication date	2026-07-26
Scheduled review	2026-10-26
Publisher	Mythos Systems
Owner	Aaron Stovall
Classification	Public
Prior edition	1.0.0-candidate / 2026-07-24
Supersession	This edition supersedes the prior candidate while preserving the permanent identity and historical source.

Identity doctrine

The permanent document ID is immutable. Production sequence labels are not part of the public identity. Atomic standards are authoritative; portfolios, workbooks, and machine-readable exports are generated views.

Normative Language and Applicability

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL indicate the strength of provisions in this Mythos standard. The publication is a Mythos-authored candidate standard and does not claim approval by the cited external authorities.

Applicability rule

Every control is applicable unless the organization documents a specific architectural or lifecycle reason that the subject is absent. N/A decisions MUST name the decision owner, evidence, affected scope, review date, and triggers that invalidate the exclusion.

Conformance evidence hierarchy

Direct deterministic proof: schemas, tests, signatures, state reconciliation, access decisions, build or runtime evidence.

Reproducible technical evidence: benchmark fixtures, traces, artifacts, profiles, logs, metrics, and environment manifests.

Independent assessment: qualified human or separate evaluator using an explicit rubric and disclosed uncertainty.

Attestation or supplier claim: informative unless independently verified and current for the deployed configuration.

Demonstration or narrative: insufficient by itself for a conforming score above 2.

Critical-control doctrine

Critical controls cannot be averaged away. A failed or stale critical control constrains the maximum conformance level regardless of the aggregate score.

Scope, Audience, and Engineering Principles

Purpose. Defines repository-level code generation, repair, review, test, security, debugging, provenance, and completion benchmarks.

In-scope systems. AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses.

Primary audience. software engineering leaders, coding-agent developers, platform engineers, security teams, evaluators, and repository owners.

Required engineering principles

- Model capability and complete system behavior MUST be measured separately.
- Human, legal, production, financial, identity, and governance authority MUST remain external to model confidence or conversational fluency.
- Evaluation MUST include expected, prohibited, adversarial, degraded, recovery, and lifecycle-change conditions.
- Source authority, data rights, permissions, versions, and freshness MUST be visible in evidence.
- Critical outcomes MUST be supported by deterministic validators or independent review wherever feasible.
- The organization MUST preserve rollback, revocation, correction, deletion, and retirement paths.
- Optimization for score, latency, throughput, tokens, or cost MUST NOT hide safety, quality, privacy, accessibility, or reliability regressions.

Exclusions

This standard does not certify a model, provider, framework, benchmark, dataset, or product. Conformance is limited to the declared system, use case, version, environment, evidence period, and assessment scope.

Conformance and Scoring Model

Level	Weighted threshold	Critical-control condition	Meaning
No conformance	< 50	Any state	Materially incomplete, unverified, or unsafe.
Foundation	>= 50	No critical control scored 0	Defined baseline with partial implementation and evidence.
Operational	>= 65	All applicable critical controls >= 3	Implemented and operationally tested.
Assured	>= 80	All applicable critical controls >= 4	Independently verified with adversarial and recovery evidence.
Continuously Assured	>= 90	All applicable critical controls = 5	Continuous regression, monitoring, freshness, and incident learning.

Weighted score

For applicable controls: weighted contribution = (control score / 5) x control weight. N/A controls are removed from the denominator only after an approved applicability decision. The final score is normalized to 100.

A conformance claim MUST include scope, system and model versions, assessment date, evidence window, evaluator identity, scores by family, critical-control status, unresolved findings, exceptions, limitations, and next review.

Control Score Interpretation

Score	Maturity	Required evidence state
0	Not implemented	Not implemented: the assessed control is absent, materially unknown, or contradicted by evidence.
1	Initial	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Nonconformity classes

Critical: credible risk of serious harm, unauthorized high-impact action, material security or privacy failure, false assurance, or inability to recover.

Major: requirement absent or ineffective across a meaningful part of scope, with no sufficient compensating control.

Minor: localized or low-impact deficiency that does not invalidate the overall control objective.

Observation: improvement opportunity or emerging risk without current nonconformity.

Score validity

A score expires when evidence exceeds its approved freshness period or a material change invalidates the assessment. Current operation is not inferred from a historic assessment.

Standard Architecture and Control Model

01	**Govern**	Scope, ownership, authority, policy, impacts, risk tolerance, exceptions, and lifecycle decisions.
02	**Map**	System boundaries, actors, models, data, prompts, tools, interfaces, populations, dependencies, and failure domains.
03	**Measure**	Representative and hidden evaluation, deterministic validation, human or model judging, uncertainty, performance, and cost.
04	**Manage**	Release gates, least privilege, monitoring, incident response, correction, rollback, revocation, deletion, and retirement.
05	**Prove**	Traceable evidence bundles connecting claims to configurations, tests, artifacts, effects, decisions, and user outcomes.

Assessment unit

The assessment unit is the declared AI system or workflow, not the model name alone. It includes prompts, retrieval, memory, tools, policies, interface, evaluators, infrastructure, human oversight, and operating environment.

Benchmark Dimensions and Weights

Dimension	Weight	Assessment emphasis
Task and repository integrity	18%	Control effectiveness, evidence quality, critical failures, operational limits, and lifecycle behavior for task and repository integrity.
Implementation correctness	22%	Control effectiveness, evidence quality, critical failures, operational limits, and lifecycle behavior for implementation correctness.
Verification and testing	18%	Control effectiveness, evidence quality, critical failures, operational limits, and lifecycle behavior for verification and testing.
Security and isolation	14%	Control effectiveness, evidence quality, critical failures, operational limits, and lifecycle behavior for security and isolation.
Architecture and maintainability	10%	Control effectiveness, evidence quality, critical failures, operational limits, and lifecycle behavior for architecture and maintainability.
Provenance and reporting	10%	Control effectiveness, evidence quality, critical failures, operational limits, and lifecycle behavior for provenance and reporting.
Efficiency and lifecycle	8%	Control effectiveness, evidence quality, critical failures, operational limits, and lifecycle behavior for efficiency and lifecycle.

Benchmark scores MUST be reported by dimension and by critical-control status. A single aggregate ranking MUST NOT be used to conceal weak safety, authority, privacy, recovery, or evidence performance.

Contents and Control Index

Control Summary Matrix

Control	Requirement	Family	Wt.	Crit.
C01	Software Task and Acceptance Contract	Task and repository integrity	6	YES
C02	Repository and Environment Reproducibility	Task and repository integrity	6	YES
C03	Repository Context and Authority	Task and repository integrity	6	YES
C04	Benchmark Holdout and Contamination Control	Implementation correctness	6	YES
C05	Problem Understanding and Plan Quality	Implementation correctness	6	-
C06	Localization and Change-Impact Analysis	Implementation correctness	5	-
C07	Patch Correctness and Behavioral Completion	Implementation correctness	5	YES
C08	Build, Type, Schema, and Compatibility Validation	Verification and testing	6	YES
C09	Test Generation and Quality	Verification and testing	6	YES
C10	Security and Abuse Resistance	Verification and testing	6	YES

Control Summary Matrix - Continued

Control	Requirement	Family	Wt.	Crit.
C11	Code Quality, Architecture, and Maintainability	Security and isolation	5	-
C12	Independent Code and Architecture Review	Security and isolation	5	YES
C13	Debugging and Root-Cause Evidence	Security and isolation	4	-
C14	Multi-File, Data, and Migration Safety	Architecture and maintainability	5	-
C15	Tool, Sandbox, and Network Governance	Architecture and maintainability	5	YES
C16	Patch and Build Provenance	Provenance and reporting	5	YES
C17	Efficiency, Cost, and Human Effort	Provenance and reporting	5	-
C18	Benchmark Scoring, Reproducibility, and Reporting	Efficiency and lifecycle	4	YES
C19	Release, Monitoring, and Retirement	Efficiency and lifecycle	4	-

Weight integrity

The 19 applicable control weights sum to 100. Family weights match the benchmark dimension model. Critical controls are highlighted in the atomic control pages and assessment workbook.

MYTHOS-AI-0004-C01

Software Task and Acceptance Contract

Family: Task and repository integrity | Weight: 6 | Critical: YES

Normative requirement

The organization MUST establish, implement, verify, and maintain software task and acceptance contract for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. The benchmark must use repository-level tasks, authoritative context, isolated workspaces, hidden tests, independent review, secure build evidence, change-impact analysis, and regression protection. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that software task and acceptance contract is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

Establish a versioned control contract for software task and acceptance contract covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.

The benchmark must use repository-level tasks, authoritative context, isolated workspaces, hidden tests, independent review, secure build evidence, change-impact analysis, and regression protection.

Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.

Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.

Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.

Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C01

Software Task and Acceptance Contract

Family: Task and repository integrity | Weight: 6 | Critical: YES

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for software task and acceptance contract.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- negative, boundary, degraded, and recovery test results
- operational telemetry and current-state verification

Required metrics

- applicable control coverage
- critical-failure count
- evidence freshness
- open nonconformities

Score anchors

0	Not implemented: Software Task and Acceptance Contract is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for software task and acceptance contract.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C02

Repository and Environment Reproducibility

Family: Task and repository integrity | Weight: 6 | Critical: YES

Normative requirement

The organization MUST establish, implement, verify, and maintain repository and environment reproducibility for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. Reproduction must pin code, data, models, prompts, schemas, tools, dependencies, environment, seeds, hardware-relevant settings, and retained artifacts. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that repository and environment reproducibility is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for repository and environment reproducibility covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- Reproduction must pin code, data, models, prompts, schemas, tools, dependencies, environment, seeds, hardware-relevant settings, and retained artifacts.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C02

Repository and Environment Reproducibility

Family: Task and repository integrity | Weight: 6 | Critical: YES

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for repository and environment reproducibility.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- immutable repository revision and isolated environment manifest
- patch, build, hidden-test, review, and provenance artifacts

Required metrics

- hidden-test pass rate
- escaped-defect rate
- security finding rate
- review minutes per accepted change
- rework rate

Score anchors

0	Not implemented: Repository and Environment Reproducibility is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for repository and environment reproducibility.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C03

Repository Context and Authority

Family: Task and repository integrity | Weight: 6 | Critical: YES

Normative requirement

The organization MUST establish, implement, verify, and maintain repository context and authority for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. Authority must be enforceable outside the model through stable identities, least privilege, scoped credentials, separation of duties, revocation, and auditable approval. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that repository context and authority is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for repository context and authority covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- Authority must be enforceable outside the model through stable identities, least privilege, scoped credentials, separation of duties, revocation, and auditable approval.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C03

Repository Context and Authority

Family: Task and repository integrity | Weight: 6 | Critical: YES

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for repository context and authority.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- immutable repository revision and isolated environment manifest
- patch, build, hidden-test, review, and provenance artifacts

Required metrics

- hidden-test pass rate
- escaped-defect rate
- security finding rate
- review minutes per accepted change
- rework rate

Score anchors

0	Not implemented: Repository Context and Authority is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for repository context and authority.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C04

Benchmark Holdout and Contamination Control

Family: Implementation correctness | Weight: 6 | Critical: YES

Normative requirement

The organization MUST establish, implement, verify, and maintain benchmark holdout and contamination control for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. Evaluation must use representative and hidden cases, reproducible configuration, deterministic validators where possible, statistical uncertainty, critical-failure gates, and slice-level reporting. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that benchmark holdout and contamination control is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for benchmark holdout and contamination control covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- Evaluation must use representative and hidden cases, reproducible configuration, deterministic validators where possible, statistical uncertainty, critical-failure gates, and slice-level reporting.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C04

Benchmark Holdout and Contamination Control

Family: Implementation correctness | Weight: 6 | Critical: YES

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for benchmark holdout and contamination control.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- negative, boundary, degraded, and recovery test results
- operational telemetry and current-state verification

Required metrics

- task success with confidence interval
- critical-failure rate
- slice variance
- reproducibility delta
- human-review burden

Score anchors

0	Not implemented: Benchmark Holdout and Contamination Control is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for benchmark holdout and contamination control.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C05

Problem Understanding and Plan Quality

Family: Implementation correctness | Weight: 6 | Critical: NO

Normative requirement

The organization MUST establish, implement, verify, and maintain problem understanding and plan quality for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. Automation must separate intent, planned change, policy decision, approved scope, applied state, observed effect, post-change validation, and rollback while preventing stale or concurrent execution. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that problem understanding and plan quality is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for problem understanding and plan quality covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- Automation must separate intent, planned change, policy decision, approved scope, applied state, observed effect, post-change validation, and rollback while preventing stale or concurrent execution.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C05

Problem Understanding and Plan Quality

Family: Implementation correctness | Weight: 6 | Critical: NO

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for problem understanding and plan quality.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- negative, boundary, degraded, and recovery test results
- operational telemetry and current-state verification

Required metrics

- applicable control coverage
- critical-failure count
- evidence freshness
- open nonconformities

Score anchors

0	Not implemented: Problem Understanding and Plan Quality is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for problem understanding and plan quality.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C06

Localization and Change-Impact Analysis

Family: Implementation correctness | Weight: 5 | Critical: NO

Normative requirement

The organization MUST establish, implement, verify, and maintain localization and change-impact analysis for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. The control must identify accountable ownership, authoritative inputs, expected outputs, operating limits, dependencies, failure behavior, evidence, and reassessment triggers. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that localization and change-impact analysis is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for localization and change-impact analysis covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- The control must identify accountable ownership, authoritative inputs, expected outputs, operating limits, dependencies, failure behavior, evidence, and reassessment triggers.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C06

Localization and Change-Impact Analysis

Family: Implementation correctness | Weight: 5 | Critical: NO

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for localization and change-impact analysis.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- negative, boundary, degraded, and recovery test results
- operational telemetry and current-state verification

Required metrics

- applicable control coverage
- critical-failure count
- evidence freshness
- open nonconformities

Score anchors

0	Not implemented: Localization and Change-Impact Analysis is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for localization and change-impact analysis.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C07

Patch Correctness and Behavioral Completion

Family: Implementation correctness | Weight: 5 | Critical: YES

Normative requirement

The organization MUST establish, implement, verify, and maintain patch correctness and behavioral completion for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. The benchmark must use repository-level tasks, authoritative context, isolated workspaces, hidden tests, independent review, secure build evidence, change-impact analysis, and regression protection. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that patch correctness and behavioral completion is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

Establish a versioned control contract for patch correctness and behavioral completion covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.

The benchmark must use repository-level tasks, authoritative context, isolated workspaces, hidden tests, independent review, secure build evidence, change-impact analysis, and regression protection.

Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.

Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.

Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.

Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C07

Patch Correctness and Behavioral Completion

Family: Implementation correctness | Weight: 5 | Critical: YES

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for patch correctness and behavioral completion.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- immutable repository revision and isolated environment manifest
- patch, build, hidden-test, review, and provenance artifacts

Required metrics

- hidden-test pass rate
- escaped-defect rate
- security finding rate
- review minutes per accepted change
- rework rate

Score anchors

0	Not implemented: Patch Correctness and Behavioral Completion is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for patch correctness and behavioral completion.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C08

Build, Type, Schema, and Compatibility Validation

Family: Verification and testing | Weight: 6 | Critical: YES

Normative requirement

The organization MUST establish, implement, verify, and maintain build, type, schema, and compatibility validation for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. Evaluation must use representative and hidden cases, reproducible configuration, deterministic validators where possible, statistical uncertainty, critical-failure gates, and slice-level reporting. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that build, type, schema, and compatibility validation is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for build, type, schema, and compatibility validation covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- Evaluation must use representative and hidden cases, reproducible configuration, deterministic validators where possible, statistical uncertainty, critical-failure gates, and slice-level reporting.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C08

Build, Type, Schema, and Compatibility Validation

Family: Verification and testing | Weight: 6 | Critical: YES

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for build, type, schema, and compatibility validation.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- immutable repository revision and isolated environment manifest
- patch, build, hidden-test, review, and provenance artifacts

Required metrics

- task success with confidence interval
- critical-failure rate
- slice variance
- reproducibility delta
- human-review burden

Score anchors

0	Not implemented: Build, Type, Schema, and Compatibility Validation is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for build, type, schema, and compatibility validation.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C09

Test Generation and Quality

Family: Verification and testing | Weight: 6 | Critical: YES

Normative requirement

The organization MUST establish, implement, verify, and maintain test generation and quality for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. The benchmark must use repository-level tasks, authoritative context, isolated workspaces, hidden tests, independent review, secure build evidence, change-impact analysis, and regression protection. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that test generation and quality is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for test generation and quality covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- The benchmark must use repository-level tasks, authoritative context, isolated workspaces, hidden tests, independent review, secure build evidence, change-impact analysis, and regression protection.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C09

Test Generation and Quality

Family: Verification and testing | Weight: 6 | Critical: YES

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for test generation and quality.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- immutable repository revision and isolated environment manifest
- patch, build, hidden-test, review, and provenance artifacts

Required metrics

- hidden-test pass rate
- escaped-defect rate
- security finding rate
- review minutes per accepted change
- rework rate

Score anchors

0	Not implemented: Test Generation and Quality is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for test generation and quality.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C10

Security and Abuse Resistance

Family: Verification and testing | Weight: 6 | Critical: YES

Normative requirement

The organization MUST establish, implement, verify, and maintain security and abuse resistance for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. The control must test realistic adversaries, direct and indirect manipulation, multi-turn escalation, cross-component attacks, prohibited outcomes, control bypass, and safe containment. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that security and abuse resistance is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for security and abuse resistance covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- The control must test realistic adversaries, direct and indirect manipulation, multi-turn escalation, cross-component attacks, prohibited outcomes, control bypass, and safe containment.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C10

Security and Abuse Resistance

Family: Verification and testing | Weight: 6 | Critical: YES

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for security and abuse resistance.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- negative, boundary, degraded, and recovery test results
- operational telemetry and current-state verification

Required metrics

- applicable control coverage
- critical-failure count
- evidence freshness
- open nonconformities

Score anchors

0	Not implemented: Security and Abuse Resistance is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for security and abuse resistance.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C11

Code Quality, Architecture, and Maintainability

Family: Security and isolation | Weight: 5 | Critical: NO

Normative requirement

The organization MUST establish, implement, verify, and maintain code quality, architecture, and maintainability for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. The benchmark must use repository-level tasks, authoritative context, isolated workspaces, hidden tests, independent review, secure build evidence, change-impact analysis, and regression protection. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that code quality, architecture, and maintainability is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for code quality, architecture, and maintainability covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- The benchmark must use repository-level tasks, authoritative context, isolated workspaces, hidden tests, independent review, secure build evidence, change-impact analysis, and regression protection.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C11

Code Quality, Architecture, and Maintainability

Family: Security and isolation | Weight: 5 | Critical: NO

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for code quality, architecture, and maintainability.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- immutable repository revision and isolated environment manifest
- patch, build, hidden-test, review, and provenance artifacts

Required metrics

- hidden-test pass rate
- escaped-defect rate
- security finding rate
- review minutes per accepted change
- rework rate

Score anchors

0	Not implemented: Code Quality, Architecture, and Maintainability is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for code quality, architecture, and maintainability.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C12

Independent Code and Architecture Review

Family: Security and isolation | Weight: 5 | Critical: YES

Normative requirement

The organization MUST establish, implement, verify, and maintain independent code and architecture review for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. The benchmark must use repository-level tasks, authoritative context, isolated workspaces, hidden tests, independent review, secure build evidence, change-impact analysis, and regression protection. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that independent code and architecture review is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for independent code and architecture review covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- The benchmark must use repository-level tasks, authoritative context, isolated workspaces, hidden tests, independent review, secure build evidence, change-impact analysis, and regression protection.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C12

Independent Code and Architecture Review

Family: Security and isolation | Weight: 5 | Critical: YES

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for independent code and architecture review.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- immutable repository revision and isolated environment manifest
- patch, build, hidden-test, review, and provenance artifacts

Required metrics

- hidden-test pass rate
- escaped-defect rate
- security finding rate
- review minutes per accepted change
- rework rate

Score anchors

0	Not implemented: Independent Code and Architecture Review is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for independent code and architecture review.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C13

Debugging and Root-Cause Evidence

Family: Security and isolation | Weight: 4 | Critical: NO

Normative requirement

The organization MUST establish, implement, verify, and maintain debugging and root-cause evidence for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. Evidence must correlate system identity, version, configuration, inputs, outputs, actions, approvals, metrics, artifacts, findings, exceptions, and final outcomes with integrity protection. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that debugging and root-cause evidence is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for debugging and root-cause evidence covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- Evidence must correlate system identity, version, configuration, inputs, outputs, actions, approvals, metrics, artifacts, findings, exceptions, and final outcomes with integrity protection.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C13

Debugging and Root-Cause Evidence

Family: Security and isolation | Weight: 4 | Critical: NO

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for debugging and root-cause evidence.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- negative, boundary, degraded, and recovery test results
- operational telemetry and current-state verification

Required metrics

- applicable control coverage
- critical-failure count
- evidence freshness
- open nonconformities

Score anchors

0	Not implemented: Debugging and Root-Cause Evidence is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for debugging and root-cause evidence.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C14

Multi-File, Data, and Migration Safety

Family: Architecture and maintainability | Weight: 5 | Critical: NO

Normative requirement

The organization MUST establish, implement, verify, and maintain multi-file, data, and migration safety for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. Data must have documented origin, rights, purpose, classification, quality, lineage, transformation history, access restrictions, retention, and deletion obligations. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that multi-file, data, and migration safety is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for multi-file, data, and migration safety covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- Data must have documented origin, rights, purpose, classification, quality, lineage, transformation history, access restrictions, retention, and deletion obligations.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C14

Multi-File, Data, and Migration Safety

Family: Architecture and maintainability | Weight: 5 | Critical: NO

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for multi-file, data, and migration safety.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- dataset or source manifest with rights, lineage, quality, and split records
- privacy, retention, deletion, and contamination review

Required metrics

- attack success rate
- critical control bypass rate
- safe abstention or containment rate
- time to detection
- retest closure rate

Score anchors

0	Not implemented: Multi-File, Data, and Migration Safety is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for multi-file, data, and migration safety.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C15

Tool, Sandbox, and Network Governance

Family: Architecture and maintainability | Weight: 5 | Critical: YES

Normative requirement

The organization MUST establish, implement, verify, and maintain tool, sandbox, and network governance for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. Capabilities must use typed contracts, bounded side effects, explicit authorization, schema validation, progress and cancellation, result evidence, compatibility tests, and revocation. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that tool, sandbox, and network governance is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for tool, sandbox, and network governance covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- Capabilities must use typed contracts, bounded side effects, explicit authorization, schema validation, progress and cancellation, result evidence, compatibility tests, and revocation.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C15

Tool, Sandbox, and Network Governance

Family: Architecture and maintainability | Weight: 5 | Critical: YES

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for tool, sandbox, and network governance.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- tool-call and external-effect receipts
- failure-injection, idempotency, rollback, and post-change validation results

Required metrics

- applicable control coverage
- critical-failure count
- evidence freshness
- open nonconformities

Score anchors

0	Not implemented: Tool, Sandbox, and Network Governance is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for tool, sandbox, and network governance.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C16

Patch and Build Provenance

Family: Provenance and reporting | Weight: 5 | Critical: YES

Normative requirement

The organization MUST establish, implement, verify, and maintain patch and build provenance for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. Evidence must correlate system identity, version, configuration, inputs, outputs, actions, approvals, metrics, artifacts, findings, exceptions, and final outcomes with integrity protection. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that patch and build provenance is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for patch and build provenance covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- Evidence must correlate system identity, version, configuration, inputs, outputs, actions, approvals, metrics, artifacts, findings, exceptions, and final outcomes with integrity protection.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C16

Patch and Build Provenance

Family: Provenance and reporting | Weight: 5 | Critical: YES

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for patch and build provenance.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- immutable repository revision and isolated environment manifest
- patch, build, hidden-test, review, and provenance artifacts

Required metrics

- hidden-test pass rate
- escaped-defect rate
- security finding rate
- review minutes per accepted change
- rework rate

Score anchors

0	Not implemented: Patch and Build Provenance is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for patch and build provenance.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C17

Efficiency, Cost, and Human Effort

Family: Provenance and reporting | Weight: 5 | Critical: NO

Normative requirement

The organization MUST establish, implement, verify, and maintain efficiency, cost, and human effort for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. Measurements must disclose workload distribution, hardware, concurrency, warmup, queueing, tail latency, successful throughput, token or media use, cost, and overload behavior. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that efficiency, cost, and human effort is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for efficiency, cost, and human effort covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- Measurements must disclose workload distribution, hardware, concurrency, warmup, queueing, tail latency, successful throughput, token or media use, cost, and overload behavior.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C17

Efficiency, Cost, and Human Effort

Family: Provenance and reporting | Weight: 5 | Critical: NO

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for efficiency, cost, and human effort.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- raw benchmark telemetry with workload and hardware disclosure
- tail-latency, throughput, saturation, cost, and resource report

Required metrics

- p50/p95/p99 end-to-end latency
- successful units per second
- saturation and rejection rate
- cost per verified outcome
- resource efficiency

Score anchors

0	Not implemented: Efficiency, Cost, and Human Effort is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for efficiency, cost, and human effort.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C18

Benchmark Scoring, Reproducibility, and Reporting

Family: Efficiency and lifecycle | Weight: 4 | Critical: YES

Normative requirement

The organization MUST establish, implement, verify, and maintain benchmark scoring, reproducibility, and reporting for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. Evaluation must use representative and hidden cases, reproducible configuration, deterministic validators where possible, statistical uncertainty, critical-failure gates, and slice-level reporting. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that benchmark scoring, reproducibility, and reporting is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for benchmark scoring, reproducibility, and reporting covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- Evaluation must use representative and hidden cases, reproducible configuration, deterministic validators where possible, statistical uncertainty, critical-failure gates, and slice-level reporting.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C18

Benchmark Scoring, Reproducibility, and Reporting

Family: Efficiency and lifecycle | Weight: 4 | Critical: YES

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for benchmark scoring, reproducibility, and reporting.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- negative, boundary, degraded, and recovery test results
- operational telemetry and current-state verification

Required metrics

- task success with confidence interval
- critical-failure rate
- slice variance
- reproducibility delta
- human-review burden

Score anchors

0	Not implemented: Benchmark Scoring, Reproducibility, and Reporting is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for benchmark scoring, reproducibility, and reporting.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

MYTHOS-AI-0004-C19

Release, Monitoring, and Retirement

Family: Efficiency and lifecycle | Weight: 4 | Critical: NO

Normative requirement

The organization MUST establish, implement, verify, and maintain release, monitoring, and retirement for all in-scope AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses. Recovery must cover safe stop, checkpoint integrity, rollback or compensation, credential revocation, artifact restoration, evidence preservation, requalification, and controlled retirement. The control MUST define acceptance criteria, prohibited outcomes, evidence, exception authority, and reassessment triggers. Where automated enforcement is feasible, the organization SHOULD enforce the requirement outside the model or agent. Any residual manual dependency MUST be documented, assigned, time-bounded, and tested.

Control objective

Objective

Ensure that release, monitoring, and retirement is explicit, bounded, testable, observable, recoverable, and governed throughout the lifecycle.

Implementation requirements

- Establish a versioned control contract for release, monitoring, and retirement covering AI coding assistants, repository agents, autonomous software-engineering systems, review agents, repair systems, and development harnesses and name the accountable owner, approver, assessor, and affected stakeholders.
- Recovery must cover safe stop, checkpoint integrity, rollback or compensation, credential revocation, artifact restoration, evidence preservation, requalification, and controlled retirement.
- Implement the control through machine-enforceable policy, typed schemas, isolated runtime boundaries, validated procedures, or an approved combination appropriate to the risk.
- Define explicit nominal, negative, adversarial, malformed, degraded, overloaded, recovery, and rollback tests; critical cases must be hidden from the system under evaluation where feasible.
- Correlate evidence to stable system, model, data, prompt, repository, tool, environment, user or tenant, and release identities; protect records against unauthorized modification.
- Set review cadence and event-driven reassessment triggers for material changes, incidents, supplier updates, drift, failed controls, new affected populations, and retirement.

Applicability

Applicable unless a documented, approved, evidence-supported exclusion demonstrates that the capability, data, decision, interface, population, or lifecycle stage is absent.

MYTHOS-AI-0004-C19

Release, Monitoring, and Retirement

Family: Efficiency and lifecycle | Weight: 4 | Critical: NO

Assessment procedure

Examine the approved design, applicability decision, responsible roles, and current implementation for release, monitoring, and retirement.

Select representative normal, boundary, high-impact, and adversarial cases, including at least one prohibited outcome and one dependency or recovery failure.

Verify the exact model, data, prompt, tool, policy, runtime, repository, environment, and evaluator versions used during assessment.

Execute deterministic validators first, then calibrated model or human judgments only where deterministic proof is not available.

Reconcile expected behavior with raw outputs, trajectories, tool effects, telemetry, artifacts, user-visible outcomes, and unresolved contradictions.

Assign a 0-5 score, record evidence links and confidence, open nonconformities, define remediation ownership, and set the next verification date.

Minimum evidence

- approved control design or operating contract with accountable owner
- versioned configuration, schema, policy, model, prompt, dataset, or architecture record
- reproducible test plan with expected and observed results
- traceable decision, exception, approval, and remediation records
- negative, boundary, degraded, and recovery test results
- operational telemetry and current-state verification

Required metrics

- recovery success rate
- duplicate-effect rate
- time to safe state
- residual-effect count
- evidence preservation rate

Score anchors

0	Not implemented: Release, Monitoring, and Retirement is absent, materially unknown, or contradicted by evidence.
1	Initial: intent is documented, but ownership, repeatability, implementation, or evidence is incomplete.
2	Defined: approved procedure and design exist, but implementation or representative testing is partial.
3	Implemented: control operates for the declared scope and passes normal and basic negative tests.
4	Verified: independent assessment, hidden or adversarial cases, current evidence, and recovery validation demonstrate effectiveness.
5	Continuously assured: automated monitoring and regression, current evidence, incident learning, and timely reassessment sustain verified effectiveness.

Failure indicators

The organization cannot identify an authoritative owner, current scope, or complete implementation for release, monitoring, and retirement.

Only a successful demonstration, vendor claim, aggregate score, or model-generated judgment is available; raw evidence and negative tests are absent.

Critical cases can be hidden by averages, unsupported N/A designations, stale evidence, or changes in model, data, prompt, tool, policy, or environment.

Failure, rollback, revocation, deletion, containment, or retirement behavior has not been exercised under realistic conditions.

Required Benchmark Scenarios

MYTHOS-AI-0004-B01 - Representative production task

Demonstrate the declared use case using current configuration and representative inputs. Apply the scenario specifically to ai software engineering benchmark and preserve raw evidence.

MYTHOS-AI-0004-B02 - Hidden critical holdout

Execute high-impact cases withheld from development, tuning, and prompt iteration. Apply the scenario specifically to ai software engineering benchmark and preserve raw evidence.

MYTHOS-AI-0004-B03 - Malformed and adversarial inputs

Test schema violations, ambiguity, direct or indirect manipulation, and unsafe instructions. Apply the scenario specifically to ai software engineering benchmark and preserve raw evidence.

MYTHOS-AI-0004-B04 - Long-horizon or multi-step execution

Measure compounding error, context loss, looping, premature completion, and recovery. Apply the scenario specifically to ai software engineering benchmark and preserve raw evidence.

MYTHOS-AI-0004-B05 - Dependency and tool failure

Inject model, network, data, tool, credential, evaluator, or service failure. Apply the scenario specifically to ai software engineering benchmark and preserve raw evidence.

Required Benchmark Scenarios - Continued

MYTHOS-AI-0004-B06 - Resource pressure and overload

Exercise concurrency, long inputs, queueing, rate limits, memory, tokens, and cost limits. Apply the scenario specifically to ai software engineering benchmark and preserve raw evidence.

MYTHOS-AI-0004-B07 - Privacy and cross-boundary test

Attempt cross-tenant, secret, personal-data, restricted-source, cache, and egress violations. Apply the scenario specifically to ai software engineering benchmark and preserve raw evidence.

MYTHOS-AI-0004-B08 - Rollback, revocation, or restore

Prove safe stop, compensation, revocation, prior-state restoration, and residual-effect reconciliation. Apply the scenario specifically to ai software engineering benchmark and preserve raw evidence.

MYTHOS-AI-0004-B09 - Human intervention and disagreement

Test approval, interruption, correction, appeal, assessor disagreement, and minority findings. Apply the scenario specifically to ai software engineering benchmark and preserve raw evidence.

MYTHOS-AI-0004-B10 - Material change and regression

Change model, prompt, dataset, tool, provider, runtime, or policy and run requalification gates. Apply the scenario specifically to ai software engineering benchmark and preserve raw evidence.

Conformance Report and Evidence Bundle

Permanent document ID, standard version, assessment version, system and use-case scope.

System, model, provider, prompt, data, retrieval, memory, tool, evaluator, runtime, repository, and infrastructure identities.

Applicability decisions and normalized denominator.

Control-level scores, weights, critical status, evidence links, assessor, date, confidence, and finding disposition.

Benchmark scenario results with raw artifacts, deterministic validators, judge or expert records, uncertainty, failures, and limitations.

Family and total score, achieved conformance level, unresolved critical or major findings, approved exceptions, expiry, and next review.

Release, rollback, revocation, deletion, and retirement evidence where applicable.

Evidence bundle integrity

The evidence bundle SHOULD use content hashes, signatures or protected repository history, stable artifact IDs, access controls, retention policy, and independent verification. Evidence links alone are insufficient when the referenced object can change without detection.

Exceptions, Waivers, and Compensating Controls

An exception MUST NOT be used to relabel a failed requirement as conforming. Exceptions document accepted residual risk for a bounded scope and time; the underlying control score remains evidence-based.

- Identify the exact control, system scope, reason, affected users or assets, and current score.
- Describe the missing requirement, residual risk, foreseeable failure, and affected decisions.
- Define compensating controls, validation evidence, monitoring, owner, approver, start date, expiry, and remediation plan.
- Obtain approval from an authority independent of the implementation owner for critical or high-impact exceptions.
- Reassess on incident, material change, evidence expiry, control failure, or exception expiration.
- Publish exceptions in the conformance report; hidden exceptions invalidate the claim.

Critical exception cap

An unresolved exception against a critical control prevents Assured or Continuously Assured conformance and may prevent Operational conformance when the control score is below 3.

Source Authority and Freshness Register

NIST - Artificial Intelligence Risk Management Framework 1.0

Cross-sector AI risk governance, mapping, measurement, and management.

https://doi.org/10.6028/NIST.AI.100-1

NIST - NIST AI 600-1 - Generative Artificial Intelligence Profile

Generative-AI risks, controls, evaluation, and lifecycle actions.

https://doi.org/10.6028/NIST.AI.600-1

NIST - AI Resource Center and AI RMF Playbook

Current implementation resources and revision notices for the AI RMF.

https://airc.nist.gov/

ISO/IEC - ISO/IEC 42001:2023 - AI management systems

AI management-system requirements and continual improvement.

https://www.iso.org/standard/42001

ISO/IEC - ISO/IEC 23894:2023 - Guidance on AI risk management

AI risk-management guidance across lifecycle and stakeholders.

https://www.iso.org/standard/77304.html

ISO/IEC - ISO/IEC 42005:2025 - AI system impact assessment

Impact-assessment guidance for individuals and societies.

https://www.iso.org/standard/42005

MLCommons - MLCommons Benchmarks

Open performance, quality, and safety benchmark programs.

https://mlcommons.org/benchmarks/

Source Authority and Freshness Register - Continued

MLCommons - AI Risk and Reliability

Safety-test methodology and risk-oriented evaluation work.

https://mlcommons.org/working-groups/ai-risk-reliability/ai-risk-reliability/

Princeton NLP - SWE-bench

Repository-level issue-resolution benchmark and environment.

https://github.com/princeton-nlp/SWE-bench

NIST - SP 800-218A

Secure development practices for generative-AI model and system development.

https://csrc.nist.gov/pubs/sp/800/218/a/final

SLSA - Supply-chain Levels for Software Artifacts v1.2

Build provenance and artifact integrity framework.

https://slsa.dev/spec/v1.2/

OpenSSF - Scorecard

Automated software-supply-chain security checks.

https://securityscorecards.dev/

Freshness rule

Sources were verified for this candidate edition on 2026-07-26. The standard MUST be revalidated when a cited authority changes, becomes superseded, is withdrawn, or materially alters the control interpretation.

Limitations, Revision History, and Adoption Position

This candidate Mythos standard is designed for engineering governance and assessment. It is not a legal opinion, regulatory certification, safety guarantee, or substitute for domain-specific professional judgment. Model behavior remains probabilistic and system risk depends on data, users, tools, deployment, incentives, and operating context.

Revision history

Version	Date	Change
2.0.0-candidate	2026-07-26	Expanded normative edition with 19 weighted controls, benchmark scenarios, conformance levels, machine-readable catalogs, assessment workbook integration, source freshness, and explicit lineage.
1.0.0-candidate	2026-07-24	Earlier permanent-identity candidate retained in the release lineage archive.

Adoption position

Use this edition as a candidate control and benchmark baseline. Organizations SHOULD pilot it on bounded systems, retain assessment evidence, submit corrections, and avoid representing candidate conformance as external certification.