REFERENCE ARCHITECTURE / ENGINEERING STANDARDS LIBRARY

# AI-Assisted Software Engineering Pipeline

A governed engineering pipeline in which AI systems assist research, design, coding, testing, review, documentation, release, and operations without bypassing software assurance.

**PERMANENT DOCUMENT ID**
RA-007

**PUBLICATION**
Candidate Reference Architecture

**ARCHITECTURE SET**
Canonical Set

# Architecture identity and operating position

| ARCHITECTURE SET | DOMAIN | LAYERS | COMPONENTS | ACCEPTANCE | DEPENDENCIES |
|---|---|---|---|---|---|
| Canonical Set | Platform Engineering | 6 | 6 | 18 | 0 |

## SYSTEM CONTEXT


## EXECUTIVE OUTCOMES

1. Convert product intent into traceable requirements, architecture decisions, tasks, code, tests, evidence, and
2. release artifacts.
3. Use agents and models under repository, tool, secret, network, budget, and branch boundaries.
4. Verify generated work through compilers, tests, analyzers, review, runtime observation, and acceptance criteria.

**ARCHITECTURE DOCTRINE**

Interpret the architecture as a bounded operating model: explicit ownership, least authority, separable desired/runtime/observed/evidence state, safe degradation, reversible change, measurable outcomes, and evidence-backed acceptance.

# Layered architecture and component model

LAYER 01
**Intent, Requirements, and Architecture**

LAYER 02
**Developer Workspace and Agent Harness**

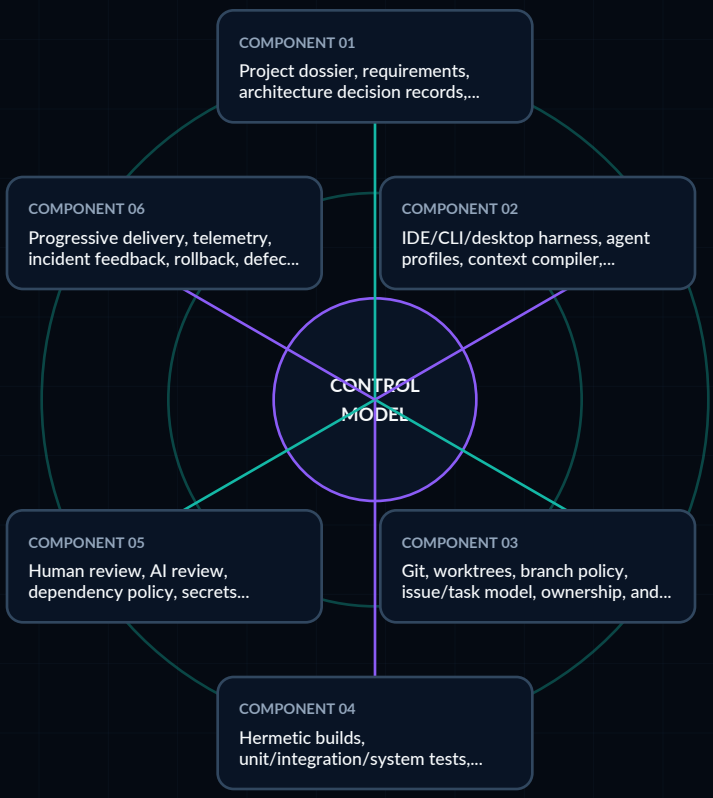
LAYER 03
**Source, Branch, and Change Management**

LAYER 04
**Build, Test, and Analysis**

LAYER 05
**Review, Security, and Release Policy**

LAYER 06
**Deployment, Observation, and Learning**

## CORE COMPONENTS




Each layer and component owns an explicit interface, state model, authority boundary, failure behavior, telemetry contract, recovery path, and evidence obligation.

# Primary sequence and control flow


**EVIDENCE THREAD** Identity + authorization + version + state transition + output + validation + error + timing + accountable disposition.

# State, data, authority, and trust flow



## TRUST BOUNDARY REGISTER

**TB-01**
Model context versus confidential repository content

**TB-02**
Agent tool access versus developer workstation and network

**TB-03**
Generated change versus protected branch

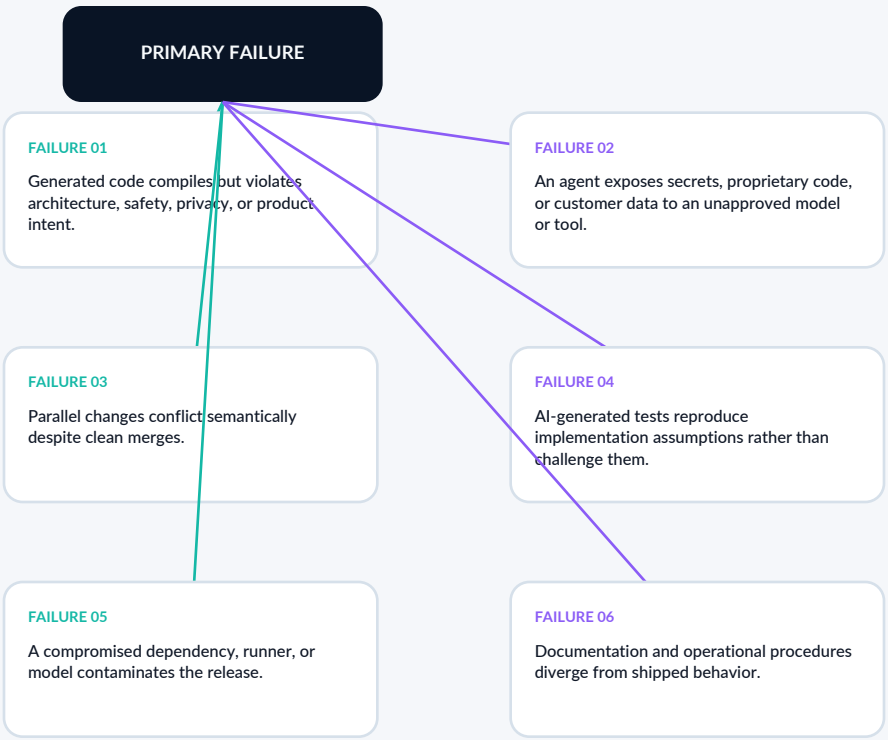
**TB-04**
Build environment versus release signing authority

**TB-05**
AI review versus accountable human approval

**TB-06**
Deployment automation versus production control plane

# Failure propagation and observability

## FAILURE PATHS




## OBSERVABILITY STACK

**OUTCOME**
User / mission result

**SERVICE**
SLOs / availability

**CONTROL**
Policy / authorization

**EXECUTION**
State / errors / retries

**DEPENDENCY**
External health / latency

**EVIDENCE**
Trace / artifact / receipt

**RECOVERY RULE** Closure requires containment, causal explanation, restored service, reconciled intended vs observed state, preserved evidence, and an explicit residual-risk decision.

# Deployment profiles and implementation roadmap

## DEPLOYMENT PROFILES

**PROFILE 01** — **INDIVIDUAL DEVELOPER ASSISTANT**

Bounded proof

**PROFILE 02** — **TEAM ENGINEERING PIPELINE**

Team-scale governance

**PROFILE 03** — **REGULATED PRODUCT RELEASE FACTORY**

Sovereign / high-assurance

**PROFILE 04** — **LARGE-SCALE MULTI-AGENT SOFTWARE DELIVERY PLATFORM**

Distributed enterprise

## IMPLEMENTATION ROADMAP

| 0-30 DAYS | 31-90 DAYS | 3-6 MONTHS | 6-12 MONTHS | ONGOING |
|---|---|---|---|---|
| Scope / baseline | Pilot / contracts | Scale / integrate | Assure / automate | Measure / revalidate |

Progression is evidence-gated. Architecture adoption does not advance because a component is installed or demonstrated; each stage requires measurable service, safety, security, recovery, and evidence outcomes.

# Verification plan, dependencies, and reading map

## ACCEPTANCE CRITERIA - REPRESENTATIVE SET

| | |
|---|---|
| AC-01 | The architecture has a named service owner, technical owner, security owner, data owner, and recovery authority. |
| AC-02 | The system boundary, external dependencies, retained responsibilities, and trust transitions are documented and |
| AC-03 | Identity, authorization, secrets, policy, and privileged operations are explicit and independently auditable. |
| AC-04 | Desired state, runtime state, observed state, and evidence state are distinguishable and reconciled. |
| AC-05 | Consequential changes require preconditions, approval policy, bounded execution, verification, and rollback. |
| AC-06 | Telemetry supports service health, security detection, forensic reconstruction, and conformance evidence. |
| AC-07 | Failure modes include safe degradation, dependency loss, corrupted state, operator error, and malicious action. |
| AC-08 | Backup, restoration, reconstitution, and exit procedures are exercised against defined recovery objectives. |

## RELATED REFERENCE ARCHITECTURES

No mandatory RA dependencies declared

## MAPPED MYTHOS STANDARDS

MYTHOS-SWE-0001, MYTHOS-SWE-0004, MYTHOS-SWE-0008, MYTHOS-PLT-0001, MYTHOS-PLT-0002, MYTHOS-AI-0007

## DETAILED SPECIFICATION READING MAP



The following pages preserve the detailed candidate architecture specification while this visual dossier adds explicit architecture diagrams, sequence views, state and evidence flows, observability, failure propagation, and implementation gates.

# PART I - EXECUTIVE DIRECTION

## OUTCOMES, RISKS, AND DECISIONS

**ARCHITECTURE MANDATE**

A governed engineering pipeline in which AI systems assist research, design, coding, testing, review, documentation, release, and operations without bypassing software assurance.

### EXECUTIVE OUTCOMES

- Convert product intent into traceable requirements, architecture decisions, tasks, code, tests, evidence, and release artifacts.
- Use agents and models under repository, tool, secret, network, budget, and branch boundaries.
- Verify generated work through compilers, tests, analyzers, review, runtime observation, and acceptance criteria.
- Produce signed, reproducible, attributable releases with safe rollback and operational feedback.

### DECISIONS REQUIRED

- Accountable service and risk ownership.
- Accepted architecture profile and consequence class.
- Authoritative identity, data, policy, configuration, and evidence systems.
- Mandatory approval, recovery, privacy, safety, and supplier-exit gates.
- Funding for operations, observability, validation, and lifecycle—not only implementation.

# PART II - SCOPE, PRINCIPLES, AND SYSTEM CONTEXT

## PLATFORM ENGINEERING

### IN SCOPE

A governed engineering pipeline in which AI systems assist research, design, coding, testing, review, documentation, release, and operations without bypassing software assurance.

### OUT OF SCOPE

Vendor selection, exact sizing, contractual obligations, detailed low-level design, product certification, regulatory approval, and proof that any named platform is implemented.

### ARCHITECTURE INVARIANTS

Explicit ownership; least authority; separable desired, runtime, observed, and evidence state; bounded automation; tested recovery; product-neutral interfaces; source and claim currency.

### ASSUMPTIONS

Organizations tailor the pattern through documented risk, environment, criticality, data, safety, regulatory, workforce, and supplier analysis.

# SYSTEM CONTEXT AND RESPONSIBILITY MODEL

## ACTORS, ENVIRONMENTS, AND EXTERNAL DEPENDENCIES

### ACTORS AND RESPONSIBILITY

- Service consumers and affected users
- Product, platform, network, security, data, reliability, and support teams
- Human administrators, operators, reviewers, incident commanders, and risk owners
- Machine identities, agents, workloads, devices, controllers, and external services
- Suppliers, providers, carriers, integrators, assessors, and regulators where applicable

### RESPONSIBILITY BOUNDARY

- The adopting organization retains accountability for purpose, risk, configuration, data, identity, operation, recovery, and evidence even where a provider performs underlying functions.
- Inherited controls are recorded with provider, scope, evidence, limitations, shared responsibility, incident path, and exit conditions.
- No component is considered trusted solely because it is internal, managed, signed, cloud-hosted, model-generated, or supplied by a recognized vendor.

**ARCHITECTURE POSITION**

AI accelerates engineering work but does not replace source control, reproducible builds, testing, security gates, review, ownership, or release authority.

# PART III - LOGICAL ARCHITECTURE

## LAYERS AND RESPONSIBILITIES

**01**

### Intent, Requirements, and Architecture

Intent, Requirements, and Architecture owns a bounded set of responsibilities, interfaces, state, controls, failure behavior, telemetry, recovery procedures, and evidence. It does not silently absorb authority assigned to another layer.

**02**

### Developer Workspace and Agent Harness

Developer Workspace and Agent Harness owns a bounded set of responsibilities, interfaces, state, controls, failure behavior, telemetry, recovery procedures, and evidence. It does not silently absorb authority assigned to another layer.

**03**

### Source, Branch, and Change Management

Source, Branch, and Change Management owns a bounded set of responsibilities, interfaces, state, controls, failure behavior, telemetry, recovery procedures, and evidence. It does not silently absorb authority assigned to another layer.

**04**

### Build, Test, and Analysis

Build, Test, and Analysis owns a bounded set of responsibilities, interfaces, state, controls, failure behavior, telemetry, recovery procedures, and evidence. It does not silently absorb authority assigned to another layer.

**05**

### Review, Security, and Release Policy

Review, Security, and Release Policy owns a bounded set of responsibilities, interfaces, state, controls, failure behavior, telemetry, recovery procedures, and evidence. It does not silently absorb authority assigned to another layer.

**06**

### Deployment, Observation, and Learning

Deployment, Observation, and Learning owns a bounded set of responsibilities, interfaces, state, controls, failure behavior, telemetry, recovery procedures, and evidence. It does not silently absorb authority assigned to another layer.

# CORE COMPONENTS - A

## COMPONENTS 01-03

### COMPONENT 01

Project dossier, requirements, architecture decision records, threat model, and acceptance tests

Required contract: owner, purpose, inputs, outputs, authority, data, dependencies, health, failure behavior, recovery, lifecycle, and evidence.

### COMPONENT 02

IDE/CLI/desktop harness, agent profiles, context compiler, sandbox, and tool broker

Required contract: owner, purpose, inputs, outputs, authority, data, dependencies, health, failure behavior, recovery, lifecycle, and evidence.

### COMPONENT 03

Git, worktrees, branch policy, issue/task model, ownership, and change provenance

Required contract: owner, purpose, inputs, outputs, authority, data, dependencies, health, failure behavior, recovery, lifecycle, and evidence.

# CORE COMPONENTS - B

## COMPONENTS 04-06

### COMPONENT 04

Hermetic builds, unit/integration/system tests, static/dynamic analysis, fuzzing, and artifact generation

Required contract: owner, purpose, inputs, outputs, authority, data, dependencies, health, failure behavior, recovery, lifecycle, and evidence.

### COMPONENT 05

Human review, AI review, dependency policy, secrets scanning, SBOM, provenance, signing, and release gates

Required contract: owner, purpose, inputs, outputs, authority, data, dependencies, health, failure behavior, recovery, lifecycle, and evidence.

### COMPONENT 06

Progressive delivery, telemetry, incident feedback, rollback, defect learning, and documentation maintenance

Required contract: owner, purpose, inputs, outputs, authority, data, dependencies, health, failure behavior, recovery, lifecycle, and evidence.

# PART IV - DATA, CONTROL, AUTHORITY, AND EVIDENCE FLOW

## FLOW CONTRACTS

1. Intent is decomposed into traceable work and testable outcomes.

   Evidence: request and actor identity, policy decision, versions, state transitions, outputs, tests, errors, timing, and accountable disposition.

2. Agents receive bounded repository views, tools, credentials, network access, time, and token budgets.

   Evidence: request and actor identity, policy decision, versions, state transitions, outputs, tests, errors, timing, and accountable disposition.

3. Changes occur in isolated branches or worktrees with continuous compile and test feedback.

   Evidence: request and actor identity, policy decision, versions, state transitions, outputs, tests, errors, timing, and accountable disposition.

4. Independent verification challenges implementation, security, UX, performance, and documentation.

   Evidence: request and actor identity, policy decision, versions, state transitions, outputs, tests, errors, timing, and accountable disposition.

5. Release factory produces signed artifacts, attestations, SBOMs, deployment plans, and rollback evidence.

   Evidence: request and actor identity, policy decision, versions, state transitions, outputs, tests, errors, timing, and accountable disposition.

6. Production signals create defects, tests, risk updates, and controlled future work.

   Evidence: request and actor identity, policy decision, versions, state transitions, outputs, tests, errors, timing, and accountable disposition.

# TRUST BOUNDARIES AND ENFORCEMENT

## EVERY CROSSING REQUIRES AN EXPLICIT CONTRACT

### TRUST BOUNDARY 01

**Model context versus confidential repository content**

Minimum contract: authenticated parties, authorized actions, data classification, protocol, encryption, validation, rate and resource limits, observability, failure behavior, revocation, and evidence.

### TRUST BOUNDARY 02

**Agent tool access versus developer workstation and network**

Minimum contract: authenticated parties, authorized actions, data classification, protocol, encryption, validation, rate and resource limits, observability, failure behavior, revocation, and evidence.

### TRUST BOUNDARY 03

**Generated change versus protected branch**

Minimum contract: authenticated parties, authorized actions, data classification, protocol, encryption, validation, rate and resource limits, observability, failure behavior, revocation, and evidence.

### TRUST BOUNDARY 04

**Build environment versus release signing authority**

Minimum contract: authenticated parties, authorized actions, data classification, protocol, encryption, validation, rate and resource limits, observability, failure behavior, revocation, and evidence.

### TRUST BOUNDARY 05

**AI review versus accountable human approval**

Minimum contract: authenticated parties, authorized actions, data classification, protocol, encryption, validation, rate and resource limits, observability, failure behavior, revocation, and evidence.

### TRUST BOUNDARY 06

**Deployment automation versus production control plane**

Minimum contract: authenticated parties, authorized actions, data classification, protocol, encryption, validation, rate and resource limits, observability, failure behavior, revocation, and evidence.

# PART V - DEPLOYMENT PROFILES

## PROFILE CHANGES SCALE, NOT ASSURANCE OUTCOMES

### PROFILE 01 / INDIVIDUAL DEVELOPER ASSISTANT

Use this profile to establish a bounded implementation with explicit scale, criticality, users, data, autonomy, external dependencies, recovery objectives, staffing, and validation depth. The profile may simplify technology or topology but must preserve identity, least authority, evidence, safe failure, recovery, lifecycle, and accountable risk acceptance.

### PROFILE 02 / TEAM ENGINEERING PIPELINE

Use this profile to establish a bounded implementation with explicit scale, criticality, users, data, autonomy, external dependencies, recovery objectives, staffing, and validation depth. The profile may simplify technology or topology but must preserve identity, least authority, evidence, safe failure, recovery, lifecycle, and accountable risk acceptance.

### PROFILE 03 / REGULATED PRODUCT RELEASE FACTORY

Use this profile to establish a bounded implementation with explicit scale, criticality, users, data, autonomy, external dependencies, recovery objectives, staffing, and validation depth. The profile may simplify technology or topology but must preserve identity, least authority, evidence, safe failure, recovery, lifecycle, and accountable risk acceptance.

### PROFILE 04 / LARGE-SCALE MULTI-AGENT SOFTWARE DELIVERY PLATFORM

Use this profile to establish a bounded implementation with explicit scale, criticality, users, data, autonomy, external dependencies, recovery objectives, staffing, and validation depth. The profile may simplify technology or topology but must preserve identity, least authority, evidence, safe failure, recovery, lifecycle, and accountable risk acceptance.

# PART VI - OPERATING LIFECYCLE

## FROM DISCOVERY THROUGH RETIREMENT

### 01 / Establish project dossier and repository policy

Entry and exit criteria, owner, evidence, exceptions, dependencies, rollback, and next decision are recorded. No phase is complete solely because components were purchased, configured, or demonstrated.

### 02 / Add isolated assistant and read-only analysis

Entry and exit criteria, owner, evidence, exceptions, dependencies, rollback, and next decision are recorded. No phase is complete solely because components were purchased, configured, or demonstrated.

### 03 / Enable bounded change generation

Entry and exit criteria, owner, evidence, exceptions, dependencies, rollback, and next decision are recorded. No phase is complete solely because components were purchased, configured, or demonstrated.

### 04 / Strengthen automated verification

Entry and exit criteria, owner, evidence, exceptions, dependencies, rollback, and next decision are recorded. No phase is complete solely because components were purchased, configured, or demonstrated.

### 05 / Introduce provenance and signed releases

Entry and exit criteria, owner, evidence, exceptions, dependencies, rollback, and next decision are recorded. No phase is complete solely because components were purchased, configured, or demonstrated.

### 06 / Integrate progressive delivery and telemetry

Entry and exit criteria, owner, evidence, exceptions, dependencies, rollback, and next decision are recorded. No phase is complete solely because components were purchased, configured, or demonstrated.

### 07 / Expand parallel agents under measured governance

Entry and exit criteria, owner, evidence, exceptions, dependencies, rollback, and next decision are recorded. No phase is complete solely because components were purchased, configured, or demonstrated.

# SECURITY, PRIVACY, SAFETY, AND RIGHTS

## CROSS-CUTTING ASSURANCE

### SECURITY

Threat model identities, trust boundaries, supply chain, control planes, data, protocols, privilege, abuse, detection, response, recovery, and residual risk.

### PRIVACY

Purpose limitation, minimization, transparency, consent where required, access, correction, retention, deletion, sovereignty, metadata, and third-party processing.

### SAFETY

Hazards, unsafe states, human intervention, physical consequence, dependency failure, automation limits, emergency stop, recovery sequence, and validation.

### RIGHTS AND USABILITY

Accessibility, contestability, explanation, operator workload, affected-user communication, grievance, misuse prevention, and equitable performance.

### SUPPLY CHAIN

Provenance, version, SBOM or component inventory, signature, vulnerability, support, update, provider, subcontractor, and exit.

### EVIDENCE

Design decisions, tests, telemetry, incidents, approvals, exceptions, recovery exercises, source currency, and review findings remain linked to the architecture version.

# FAILURE MODES AND RESILIENCE

## DESIGN FOR SAFE DEGRADATION AND TRUSTWORTHY RECOVERY

**FAILURE SCENARIO 01**

Generated code compiles but violates architecture, safety, privacy, or product intent.

Required response: detect, contain, preserve evidence, maintain or enter safe service, communicate, recover, verify, learn, and prevent recurrence.

**FAILURE SCENARIO 02**

An agent exposes secrets, proprietary code, or customer data to an unapproved model or tool.

Required response: detect, contain, preserve evidence, maintain or enter safe service, communicate, recover, verify, learn, and prevent recurrence.

**FAILURE SCENARIO 03**

Parallel changes conflict semantically despite clean merges.

Required response: detect, contain, preserve evidence, maintain or enter safe service, communicate, recover, verify, learn, and prevent recurrence.

**FAILURE SCENARIO 04**

AI-generated tests reproduce implementation assumptions rather than challenge them.

Required response: detect, contain, preserve evidence, maintain or enter safe service, communicate, recover, verify, learn, and prevent recurrence.

**FAILURE SCENARIO 05**

A compromised dependency, runner, or model contaminates the release.

Required response: detect, contain, preserve evidence, maintain or enter safe service, communicate, recover, verify, learn, and prevent recurrence.

**FAILURE SCENARIO 06**

Documentation and operational procedures diverge from shipped behavior.

Required response: detect, contain, preserve evidence, maintain or enter safe service, communicate, recover, verify, learn, and prevent recurrence.

# PART VII - INTEGRATION AND DEPENDENCY MODEL

## CONNECTED REFERENCE ARCHITECTURES AND STANDARDS

### REFERENCE ARCHITECTURE DEPENDENCIES

- RA-008 - Multi-Cloud Landing Zone and Internal Developer Platform
- RA-009 - Zero-Trust Identity, Policy, Secrets, and Access Fabric
- RA-010 - Local-First Knowledge, Memory, and Evidence Platform
- RA-013 - Secure Software Supply Chain and Release Factory
- RA-014 - Enterprise Data, API, Event, and Integration Fabric
- RA-019 - Sovereign Browser, Remote Access, and Web Automation
- RA-020 - Adaptive Learning, Cyber Range, and Workforce Assurance

### MAPPED MYTHOS STANDARDS

- MYTHOS-SWE-0001
- MYTHOS-SWE-0004
- MYTHOS-SWE-0008
- MYTHOS-PLT-0001
- MYTHOS-PLT-0002
- MYTHOS-AI-0007

**DEPENDENCY DOCTRINE**

A dependency is not a blanket trust grant. Each connection requires an interface, identity, policy, data, availability, version, failure, recovery, and evidence contract. Correlated failure and circular authority must be analyzed across the full architecture graph.

# PART VIII - IMPLEMENTATION AND ADOPTION ROADMAP

## PHASED, EVIDENCE-DRIVEN TRANSITION

| | |
|---|---|
| **0-30 DAYS** | Inventory purpose, owners, users, systems, data, dependencies, risks, current diagrams, incidents, recovery, and evidence gaps. |
| **31-90 DAYS** | Establish authoritative identity, configuration, policy, data, source, and architecture records; define target profile and acceptance tests. |
| **3-6 MONTHS** | Implement bounded foundations, isolation, telemetry, change control, backup, recovery, and representative pilot workflows. |
| **6-12 MONTHS** | Expand through standard patterns, automation, service ownership, SLOs, independent assessment, failure injection, and supplier-exit exercises. |
| **ONGOING** | Reconcile drift, review sources, reassess risk, measure outcomes, exercise recovery, renew exceptions, migrate dependencies, and retire obsolete paths. |

# ARCHITECTURE ACCEPTANCE CRITERIA - A

## CRITERIA 01-09

| | |
|---|---|
| **AC-01** | The architecture has a named service owner, technical owner, security owner, data owner, and recovery authority. |
| **AC-02** | The system boundary, external dependencies, retained responsibilities, and trust transitions are documented and reviewed. |
| **AC-03** | Identity, authorization, secrets, policy, and privileged operations are explicit and independently auditable. |
| **AC-04** | Desired state, runtime state, observed state, and evidence state are distinguishable and reconciled. |
| **AC-05** | Consequential changes require preconditions, approval policy, bounded execution, verification, and rollback. |
| **AC-06** | Telemetry supports service health, security detection, forensic reconstruction, and conformance evidence. |
| **AC-07** | Failure modes include safe degradation, dependency loss, corrupted state, operator error, and malicious action. |
| **AC-08** | Backup, restoration, reconstitution, and exit procedures are exercised against defined recovery objectives. |
| **AC-09** | Data classification, retention, privacy, sovereignty, and disposition controls follow the actual information flow. |

# ARCHITECTURE ACCEPTANCE CRITERIA - B

## CRITERIA 10-18

| | |
|---|---|
| **AC-10** | Supplier, model, protocol, platform, and software dependencies have provenance, version, support, and replacement records. |
| **AC-11** | Representative performance and capacity tests include saturation, contention, latency, and degraded conditions. |
| **AC-12** | Exceptions are time-bounded, visible, approved by accountable risk owners, and linked to compensating controls. |
| **AC-13** | Every generated change is attributable to a mission, model profile, context set, tool grant, branch, authorizing user, and verification record. |
| **AC-14** | Agents cannot access secrets, protected branches, production, or unrestricted networks by default. |
| **AC-15** | Acceptance tests originate from requirements and risk analysis, not solely from generated implementation. |
| **AC-16** | Builds are reproducible or explain why they are not; release artifacts include provenance, SBOM, signatures, and policy results. |
| **AC-17** | Independent review can reject, amend, or request evidence without being silently overridden by the generating agent. |
| **AC-18** | Production incidents and defects create regression tests and architecture updates before the same class of change is retried. |

# METRICS, EVIDENCE, AND DECISION GATES

## MEASURE SERVICE OUTCOMES AND ASSURANCE

### COVERAGE

Percent of components, interfaces, identities, data flows, dependencies, risks, and controls represented in current architecture evidence.

### CHANGE QUALITY

Plan success, rollback, verification pass, escaped defect, drift, and emergency-change rates.

### RELIABILITY

Availability, latency, saturation, dependency health, recovery time, recovery point, and safe-degradation success.

### SECURITY

Privilege exposure, policy denial, identity compromise, vulnerable dependency, detection coverage, response, and evidence completeness.

### HUMAN OUTCOME

Task success, operator workload, accessibility, override, contestability, training, and user-impact measures.

### LIFECYCLE

Version currency, unsupported component, exception age, migration readiness, restore exercise, and retirement completeness.

Release or expansion gate: mandatory acceptance criteria pass; failed safety, privacy, identity, recovery, evidence, or legal gates block promotion regardless of aggregate score.

# SOURCE AUTHORITY REGISTER

## CURRENT PRIMARY AND OFFICIAL REFERENCES

**SOURCE 01** **NIST SP 800-218, Secure Software Development Framework**

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

**SOURCE 02** **CISA Secure by Design**

https://www.cisa.gov/resources-tools/resources/secure-by-design

**SOURCE 03** **SLSA Specification 1.2**

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

**SOURCE 04** **NIST AI Risk Management Framework 1.0 (revision in progress as of publication date)**

https://www.nist.gov/itl/ai-risk-management-framework

**SOURCE 05** **NIST SP 800-53 Rev. 5**

https://doi.org/10.6028/NIST.SP.800-53r5

**SOURCE 06** **OpenTelemetry Specification**

https://opentelemetry.io/docs/specs/

Source currency note. Official frameworks, specifications, and implementation guidance evolve. Assessors must verify the current version, status, errata, applicability, and effective date at adoption and scheduled review. NIST AI RMF 1.0 was under revision at the publication date.

# PUBLICATION CONTROL AND REVISION

## PERMANENT IDENTITY / CONTROLLED EVOLUTION

| | |
|---|---|
| PERMANENT DOCUMENT IDENTITY | RA-007 |
| VERSION | 1.0.0-candidate |
| PUBLICATION DATE | 2026-07-24 |
| SCHEDULED REVIEW | 2027-01-24 |
| STATUS | Candidate Reference Architecture |
| ARCHITECTURE SET | Canonical Set |
| CLASSIFICATION | Public |
| PUBLISHER | Mythos Systems |

### REVISION RECORD

1.0.0-candidate (2026-07-24) - Permanent-publication candidate edition regenerated under the Mythos Systems architecture publication system. Production sequence metadata is excluded from public identity.