# Stable Authority Boundary

**Conformance Profile for Legitimate Execution**

By David B. Forbes

Published by BLOCK VECTOR Technologies, L.L.C.
Canonical citation record: *Zenodo*. Supporting materials: *blockvectortech.com*.



.

**Foreword (informative)**

Modern autonomous and distributed systems increasingly operate in conditions where authority, coordination, and information quality cannot be assumed. In these environments, failure is not limited to incorrect outputs or degraded performance; a more fundamental risk emerges: a system may **proceed as if authorized** when authority is absent, out of scope, expired, conflicted, or unverifiable at the moment of decision.

The Stable Authority Boundary (SAB) addresses this problem by treating **execution legitimacy** as a first-class engineering constraint. SAB does not attempt to prove that a system "works" or that a decision was optimal. Instead, it constrains what a system is **permitted to do** under uncertainty and degraded coordination, and it requires that non-progress outcomes are governed rather than implied. In SAB, **Act**, **Defer/Hold**, **Refuse**, and (where applicable) **Silence** are legitimate execution outcomes when they are selected through an explicit authority gate and supported by evidence at a declared quality level.

This document is written as a conformance profile: it defines a minimum set of SHALL-level requirements, evidence artifacts, and a repeatable test method so that implementers can make auditable conformance claims. The intent is to enable independent evaluation of whether a system's decision boundaries are authority-bounded and whether the system preserves legitimacy when coordination is degraded, signals conflict, or required information is missing.

Engineering Seams (SEAMS) is included as the supporting audit lens used by the test method. SEAMS is used to identify where authority is missing, implied, contradicted, or bypassable—particularly at decision boundaries that tend to be undocumented or treated as "implementation detail." The results of the SEAMS audit are captured in the required evidence artifacts and used to substantiate or deny conformance claims.

This profile is designed to be used within broader assurance efforts (safety cases, security reviews, mission assurance, and operational governance) as a focused tool for one specific question: *did the system remain inside legitimate execution when it chose to act—or chose not to?*

**1. SCOPE** 7

1.1 Purpose 7
1.2 In-scope systems 8
1.3 Out of scope 10
1.4 Intended users 11
1.5 Conformance claim statement 11

**CHAPTER 2 — BACKGROUND AND LITERATURE** 15

2.1 Why autonomy and distribution amplify "authority drift" 15
2.2 Where systems fail in practice: implicit authority and responsibility diffusion 15
2.3 Degraded coordination as the normal adversary, not the edge case 16
2.4 What standards and assurance practice typically prove—and what they don't 16
2.6 Governance, non-progress, and the "silence problem" 18
2.7 Positioning SAB relative to existing approaches 18
2.8 Where existing standards "fall short" in SAB terms 19
2.9 Summary 20

**3. TERMS, DEFINITIONS, AND ABBREVIATIONS** 20

3.1 Terms and definitions 20
*3.1.1 Authority* 20
*3.1.2 Permission to execute* 21
*3.1.3 Legitimacy / legitimate execution* 22
*3.1.4 Illegitimate execution* 22
*3.1.5 Stable Authority Boundary (SAB)* 23
*3.1.6 Boundary condition* 24
*3.1.7 Decision boundary* 24
*3.1.8 Degraded coordination* 24
*3.1.9 Uncertainty / ambiguous input* 25
*3.1.10 Conflict (signal conflict, authority conflict)* 25
*3.1.11 Act* 26
*3.1.12 Defer / Hold* 26
*3.1.13 Refuse* 27
*3.1.14 Silence (intentional non-output)* 28
*3.1.15 Conformance evidence / audit evidence* 28
*3.1.16 Engineering seam* 29
*3.1.17 Ability to execute (A2E)* 29
*3.1.18 Time-to-live (TTL)* 29
*3.1.19 Legitimate execution (LTE)* 30
*3.1.20 Illegitimate execution (ITE)* 30
3.2 Abbreviations 30

**4.0 CONFORMANCE MODEL** 30

4.1 Conformance targets 30
4.2 Conformance levels 31
4.3 Conformance statement format 31
4.4 Non-conformance 32

**5. AUTHORITY MODEL REQUIREMENTS** 32

5.1 AUTHORITY DECLARATION 32
5.2 AUTHORITY BOUNDS 32
5.3 VALIDITY AND EXPIRY 33
5.4 DECISION PLACEMENT 33
5.5 CONFLICT RULES 33
5.6 ABILITY TO EXECUTE (A2E) 33

**6. EXECUTION-STATE REQUIREMENTS (ACT / DEFER / REFUSE)** 34

6.1 PERMITTED ACTION (ACT) 34
6.2 DEFER / HOLD STATE 34
6.3 REFUSAL STATE 34
6.4 SILENCE AS A GOVERNED STATE 34
6.5 DEGRADED COORDINATION CONSTRAINTS 35

**7. EVIDENCE AND TEST METHOD** 35

7.1 REQUIRED EVIDENCE ARTIFACTS 35
7.2 EVIDENCE QUALITY LEVELS 35
7.3 TEST METHOD OVERVIEW 36
7.4 ENGINEERING SEAMS AUDIT METHOD (NORMATIVE REFERENCE) 36

**8. VALIDATION APPROACH** 36

8.1 VALIDATION GOALS 36
8.2 AVOIDING CIRCULAR VALIDATION 36
8.3 RED-TEAM VALIDATION PROTOCOL 36
8.4 DISPOSITION RULES 37
8.5 REPEATABILITY AND VARIANCE CONTROLS 37

**9. GOVERNANCE AND MAINTENANCE** 37

9.1 CHANGE CONTROL 37
9.2 PROFILE EVOLUTION 37
9.3 PUBLICATION AND REVIEW CYCLE 37

**APPENDIX** 37

***Appendix A: SAB Core Conformance Requirements Table*** *37*
***Appendix B: Engineering Seams Method Spec*** *37*
***Appendix C: Evidence Templates*** *37*
***Appendix D: Worked examples*** *38*
***Appendix E: Mapping to existing assurance frameworks*** *38*
***Appendix F: Design patterns and anti-patterns*** *38*

**APPENDIX G: RESEARCH FRAMING AND OPEN QUESTIONS (INFORMATIVE)** 38

G.1 PURPOSE OF THIS APPENDIX 38
G.2 RESEARCH FRAMING 38
G.3 EVALUATION QUESTION FOR REVIEWERS (INFORMATIVE) 39
G.4 OPEN QUESTIONS FOR RESEARCH AND STANDARDIZATION (INFORMATIVE) 39

*G.4.1 Authority semantics across domains* ____ *39*
*G.4.2 Evidence sufficiency and quality levels* ____ *39*
*G.4.3 Degraded coordination and contested control* ____ *39*
*G.4.4 Non-progress outcomes as legitimate behavior* ____ *39*
*G.4.5 Practical test method repeatability* ____ *40*
*G.4.6 Composition and inheritance* ____ *40*
G.5 CANDIDATE RESEARCH-TO-STANDARDIZATION PATH (INFORMATIVE) ____ 40
G.6 NON-NORMATIVE STATEMENT ____ 40

# 1. Scope

## 1.1 Purpose

This document defines a conformance profile for the *Stable Authority Boundary* (SAB): the minimum, testable requirements that determine when an autonomous or distributed system (or declared subsystem/component) is **permitted to proceed**, is **required to defer**, or is **required to refuse** an in-scope action. This profile is standards-forward: it expresses requirements using SHALL-level language, pairs those requirements with mandatory evidence artifacts, and defines a repeatable test method so that conformance claims can be independently audited.

SAB does not attempt to demonstrate that a system "works," achieves mission objectives, or produces correct outputs. SAB constrains **execution legitimacy**. It requires that any attempt to act is preceded by an explicit, runtime-evaluated authority gate; that authority sources and bounds are declared; that conditions for action are bounded and testable; and that behavior under uncertainty and degraded coordination is governed rather than implied. In SAB, "not acting" is not a failure mode—it is an explicitly supported execution outcome when legitimacy cannot be established.

The SAB invariant enforced by this profile is:

**If valid authority for a proposed action cannot be demonstrated for the present context at execution time, the system SHALL NOT proceed as if it were authorized.**

Accordingly, this profile requires that systems implement and evidence governed non-progress states, including **defer** (intentional non-progression pending required inputs or authority), **refuse** (terminal non-progression because authority is invalid or constraints are violated), and—where applicable—**silence** (intentional suppression of output or action to prevent illegitimate influence, escalation, or implied authorization). These outcomes are treated as legitimate system behaviors and SHALL be testable, attributable to declared conditions, and captured in evidence artifacts.

Conformance to this profile is scoped: a system SHALL NOT claim conformance except for declared action classes and declared components. A valid conformance claim SHALL identify the conformance target, the scope of actions covered, the conformance level, the profile version, and any exclusions with rationale and compensating controls. Evidence for conformance SHALL be sufficient to reconstruct: (1) what action was proposed, (2) what authority was asserted, (3) what bounds and conditions applied, (4) what the gate evaluated at execution time, and (5) why the system proceeded, deferred, refused, or remained silent.

## 1.2 In-scope systems

This profile applies to engineered systems (or declared subsystems/components) in which actions are initiated, selected, or executed by software agents, automated workflows, or distributed components, where correctness of action selection may be impacted by partial information, delayed coordination, ambiguous authority, or contested control.

This profile applies when **any** of the following are true:

- The system can change state outside itself (e.g., actuate, deploy, route, grant access, mutate data, open/close valves, start/stop services).
- The system may act while inputs are incomplete, stale, conflicting, or unavailable.
- The system's authority is delegated (human, policy, credential, mission rule, supervisory controller) and may be time-bounded, scope-bounded, or revocable.
- The system can encounter degraded coordination conditions (partition, quorum loss, latency spikes, intermittent links, loss of supervisor/operator, contested command channels).

### *In-scope system classes include:*

- **Autonomous and semi-autonomous systems**
  - autonomous agents and multi-agent systems
  - human-in-the-loop, human-on-the-loop, and supervisory control arrangements
  - systems with fallback autonomy modes during communications loss
- **Distributed and cloud-native control systems**
  - distributed services that execute changes (deploys, configuration, routing, failover, scaling, access control)
  - data mutation workflows (writes, deletes, merges, state transitions)
  - control planes, schedulers, orchestrators, and admission controllers
- **Cyber-physical and operational automation**
  - robotics and industrial automation (where actions affect physical processes)
  - mission operations automation and ground systems
  - safety interlocks and operational limit enforcement components (where they gate action)
- **Socio-technical execution loops**
  - systems where policy, operators, and automation jointly determine action (runbooks, approvals, change windows)
  - operator-intent systems where a human proposes an action but automation selects timing/extent under constraints
- **Degraded coordination contexts**
  - partition, latency, intermittent links
  - quorum loss, split-brain risk, supervisor loss
  - contested control, conflicting commands, or ambiguous command authority

**Note (normative intent):** SAB requirements apply to in-scope actions and the decision boundaries that permit, defer, refuse, or suppress those actions. Systems may claim conformance for a declared subset of actions and components.

**Legitimacy vs correctness lock:** This profile evaluates whether an action was legitimate to execute under declared authority and conditions; it does not evaluate whether the action was correct or optimal.

**Mechanism-agnostic lock:** This profile is mechanism-agnostic: it requires that authority can be demonstrated as valid at execution time, but it does not standardize the mechanism by which authority is represented or verified.

**Action taxonomy:** For the purposes of this profile, an *action* is any system-initiated or system-authorized operation that produces an externally meaningful state change or influence. In-scope actions typically fall into one or more of the following classes.

- **REQ-SCOPE-001 (Action Class Declaration):** A conformance claim SHALL declare which action classes from the Action Taxonomy are in-scope.

### *Action classes:*

- **State mutation:** writes/deletes/updates to durable state (databases, ledgers, configuration stores, distributed KV, etc.)
- **Access and privilege change:** granting/revoking permissions, role changes, credential issuance/rotation, policy rule changes, ACL edits
- **Routing / traffic / connectivity change:** route advertisement/withdrawal, failover, DNS/service discovery changes, firewall rule updates, traffic shifting
- **Deployment / configuration change:** release promotion, rollback, feature-flag changes, config reload, infrastructure-as-code apply, orchestration rescheduling
- **Resource control:** starting/stopping services, scaling up/down, allocating quotas, power-state changes, storage attach/detach
- **External actuation:** commands to physical devices or cyber-physical systems (robots, valves, motors, relays, spacecraft subsystems)
- **External signaling / commitments:** emitting commands/events/messages that downstream systems treat as authoritative (orders, approvals, triggers, irreversible commits)
- **Safety/guardrail overrides:** bypassing interlocks, suppressing alarms, changing thresholds, switching to degraded autonomy modes

## 1.3 Out of scope

This profile does not define, require, or standardize the following, except where necessary to **declare authority sources and test conformance**:

- **Performance and optimization**
  - performance targets, optimization strategies, scheduling goals, or control-law design
  - "best possible" decisions under uncertainty (SAB governs legitimacy, not optimality)
- **Functional correctness and mission success**
  - proof that the system "works," meets requirements, or achieves objectives
  - business logic correctness, model accuracy, or planning optimality
- **Complete safety/certification regimes**
  - full safety case construction, hazard analysis, certification programs, or regulatory compliance (SAB may be used *within* them)
  - domain-specific safety standards compliance requirements (aviation, medical, nuclear, automotive), except via mapping
- **Ethics / values / policy content**
  - ethical policy frameworks, value alignment policies, or normative "what should be done" decision rules
  - organizational policies beyond what is required to define authority sources, bounds, and escalation paths
- **Security mechanism standardization**
  - cryptographic identity standards, specific security protocols, key management, or authentication schemes
  - detailed threat models (SAB requires verifiable authority, not a particular mechanism)
- **System architecture mandates**
  - prescribed architectures (centralized vs distributed), consensus choices, quorum algorithms, deployment topologies
  - programming languages, tooling, vendor products, or runtime platforms
- **Human factors and training**
  - operator training, staffing levels, or procedural compliance programs (except where operator authority sources must be declared)
- **Data governance and privacy frameworks**
  - privacy policy compliance, data retention regimes, or regulatory data controls beyond required evidence integrity/retention for conformance artifacts
- **Fault tolerance as an objective**
  - SAB does not guarantee availability, uptime, or resilience outcomes; it constrains *what the system is permitted to do* when resilience mechanisms fail or coordination degrades

## 1.4 Intended users

Intended users include systems engineers, platform architects, safety/assurance engineers, mission operators, SRE/operations teams, security reviewers, and auditors. This profile is also written to be usable by program reviewers who must evaluate whether automation is constrained to legitimate execution under uncertainty, and by operators evaluating analogous constraints in human-led procedures that delegate authority under uncertainty.

**Domain applicability note:** The Stable Authority Boundary (SAB) and legitimate-execution requirements in this profile are expressed in system-agnostic terms (authority, delegation, decision boundary, refusal, deferment, silence, auditability). While the primary target is engineered automation (software, autonomous, distributed, and cyber-physical systems), the same constraints can be applied to human-led operational procedures and other execution domains where actions are taken under uncertainty and delegated authority. Domain-specific profiles may be derived from this document by mapping SAB terms to local roles, controls, and acceptable outcomes.

## 1.5 Conformance claim statement

A system (or declared subsystem/component) **MAY** claim conformance to this profile only when all profile requirements applicable to the declared scope are satisfied. At minimum, this includes: (a) authority sources are explicitly declared and bounded for all in-scope actions; (b) an execution-time gate evaluates authority validity prior to action; (c) governed non-progress outcomes are implemented (defer, refuse, and where applicable silence); (d) conformance evidence artifacts are produced at the required quality level; and (e) the conformance test method is applied and recorded.

A conformance claim is valid only when it is **explicit, reviewable, and traceable** to supporting evidence. Accordingly, a conformance claim **SHALL** state: (1) the conformance target, (2) the conformance level, (3) the version of this profile, (4) the scope of actions covered, and (5) known exclusions and their rationale.

The following **Conformance Claim Requirements (CC-001 through CC-015)** define the mandatory content, traceability, integrity, and maintenance conditions for any conformance claim under this profile.

- CC-001 — Written claim
  A conformance claim **SHALL** be made in a durable written form (document, system manifest, or signed report) that can be referenced by identifier.
- CC-002 — Unique identification
  A conformance claim **SHALL** include a unique identifier for the conformance target (system/subsystem/component build ID, version, or immutable artifact hash).

- CC-003 — Operational conditions
  A conformance claim **SHALL** state the operational context in which conformance is asserted, including at minimum: deployment environment(s), execution mode(s), and any safety/security posture assumptions that affect authority validation.
- CC-004 — Authority source enumeration
  A conformance claim **SHALL** enumerate the authority source types used for in-scope actions (e.g., human operator authorization, policy engine, cryptographic credential, mission ruleset, supervisory controller) and **SHALL** identify where each is validated at runtime.
- CC-005 — Gate placement and coverage
  A conformance claim **SHALL** identify the execution gate location(s) in the control flow (or equivalent decision boundary) and **SHALL** state whether the gate covers all in-scope actions or only a declared subset.
- CC-006 — Non-progress state mapping
  A conformance claim **SHALL** map each in-scope action class to its governed non-progress outcomes (defer/refuse/silence if applicable) and **SHALL** state the conditions that trigger each non-progress outcome category.
- CC-007 — Evidence traceability
  A conformance claim **SHALL** provide traceability from each applicable requirement in this profile to evidence artifacts that demonstrate satisfaction (requirements-to-evidence mapping).
- CC-008 — Evidence integrity
  Evidence artifacts used to support a conformance claim **SHALL** be protected against undetected modification (e.g., cryptographic hash, signed log, immutable storage, or equivalent integrity mechanism) at the declared conformance level.
- CC-009 — Reproducibility
  A conformance claim **SHALL** include sufficient information to reproduce the conformance test method application, including tool versions, configuration parameters, datasets or scenarios used (where applicable), and pass/fail criteria.
- CC-010 — Time-bounding
  A conformance claim **SHALL** include the date of assessment and the validity window (or event) that invalidates the claim (e.g., software update, policy update, authority-source change, retraining, configuration change).
- CC-011 — Change control
  Any change that affects authority sources, authority validation logic, the execution gate, or non-progress behaviors **SHALL** trigger re-application of the conformance test method before the conformance claim remains valid.
- CC-012 — Exclusions discipline
  Any exclusion **SHALL** be explicitly scoped to action types and conditions, **SHALL** include rationale, and **SHALL** state compensating controls (or explicitly state “none”).
- CC-013 — No implied conformance
  A system **SHALL NOT** claim partial conformance by implication. If the claim is partial, the claim **SHALL** state “partial conformance” and enumerate which requirement IDs are not satisfied.

- CC-014 — Level-specific minimums
  A conformance claim **SHALL** state the minimum evidence quality level achieved and **SHALL NOT** claim a higher conformance level unless all level-specific SHALL requirements are met.
- CC-015 — Independent review (optional but strong)
  For systems operating in safety- or mission-critical contexts, the conformance claim **SHALL** state whether an independent reviewer assessed the evidence, and if not, **SHALL** state why.

# Chapter 2 — Background and Literature

## 2.1 Why autonomy and distribution amplify "authority drift"

Modern engineered systems rarely execute as a single, centralized decision-maker. They execute as **distributed components**, often with **partial information**, **delayed coordination**, and **delegated authority**. In these conditions, failure is not only "wrong output" or "performance degradation." A deeper class of failure appears: *the system proceeds as if authorized when authority is absent, expired, out of scope, conflicted, or unverifiable at the moment of execution.*

### *Stable Authority Boundary (SAB)…*

This problem emerges because autonomy and distribution create structural incentives for *implicit authority*:

- Autonomy pushes decisions closer to the edge (local controllers, agents, admission controllers, orchestrators), where global intent is often incomplete.
- Distribution fragments the "source of truth" for state and permission (policy, leases, quorums, supervisors, runbooks), especially during partitions and recovery.
- Delegation separates "who intended" from "what executed," which can create accountability diffusion and responsibility gaps in human–automation handoffs.

SAB frames this as an engineering constraint: **legitimacy is not implied by capability**. A system may be able to act while lacking authority to act, and the risk surface grows when coordination degrades and verification signals go stale.

## 2.2 Where systems fail in practice: implicit authority and responsibility diffusion

A recurring theme across autonomy and governance literature is that automation can produce responsibility diffusion: outcomes occur, but it becomes unclear who actually held authority at the decision moment and whether the system's action was "permitted" versus merely "possible." This appears in discussions of algorithmic governance and human–AI transitions, where accountability is transitional and often poorly instrumented at the boundaries where authority changes hands.

In engineering terms, this diffusion becomes concrete at **decision boundaries**:

- A controller "fails over" because it detects degraded service, but the authority to reroute was contingent on quorum or a supervisor signal that is now missing.
- An orchestrator "rolls forward" because it sees drift, but the change window or approval lease has expired.

- An agent “continues operating” during lost comms, but its delegated autonomy mode was never explicitly bounded for this context.

The pattern is consistent: the system treats “lack of disproof” as permission. SAB rejects this default. It makes legitimacy **a positive condition that must be demonstrated** at execution time for the proposed action in the present context.

## 2.3 Degraded coordination as the normal adversary, not the edge case

Distributed systems research and production experience emphasize that **partitions**, quorum loss, and leader ambiguity are not exotic.

They’re *inevitable* operational states. Split-brain risk and quorum failure scenarios illustrate a key point: components can each act “correctly” locally while the system-level outcome becomes inconsistent or unsafe. The usual mitigation is quorum/leader discipline—only the majority partition may commit changes—because without coordination, **authority to commit** becomes ambiguous.

However, even when distributed algorithms prevent inconsistent writes, they do not automatically guarantee **legitimacy of execution** under the broader authority model (policy, mission rules, human approvals, operational modes). They protect *consistency*, not *permission*. A system can remain consistent and still execute an action that was **out of bounds**, **expired**, or **unapproved** in the relevant governance context.

SAB treats degraded coordination as a first-class condition that must *reduce* action permissions or raise evidence/authority thresholds, not silently preserve “normal mode” behavior. This aligns with conservative operational practice (reduce blast radius when the system cannot verify), but SAB makes it testable and auditable through explicit gate behavior and evidence artifacts.

## 2.4 What standards and assurance practice typically prove—and what they don’t

Safety and assurance standards (and the safety-case tradition) excel at a particular job: **arguing that a system is acceptably suspect to hazards, requirements, and verification activities**. In many safety-critical domains, the focus is on correctness of the software and adequacy of the lifecycle assurance activities. NASA’s assessment of safety-critical standards highlights that standards often emphasize software correctness verification and process assurance, while leaving broader system-level safety questions dependent on architecture, integration, and argumentation outside the software verification scope.

That distinction matters here:

- Standards can show that a component *behaves correctly* under specified assumptions.
- They often do not force the system to show that it **remained legitimate** when those assumptions fail (stale authority, missing coordinator, conflicting command channels, unverifiable approval).

Safety cases can, in principle, argue legitimacy. But in practice they frequently rely on *narrative argumentation* and domain-specific evidence. SAB takes a narrower, standards-forward slice: it defines **minimum SHALL-level requirements** and **evidence artifacts** that focus on one question:

Did the system remain inside legitimate execution when it chose to act—or chose not to?

Stable Authority Boundary (SAB)…

SAB is therefore complementary: it is not a full safety regime, and it does not replace safety cases. Instead, it supplies a **repeatable conformance profile** for legitimacy gating that can be inserted into broader assurance frameworks as a constrained, auditable claim.

Stable Authority Boundary (SAB)…

Stable Authority Boundary (SAB)…

Access control and authorization: close neighbor, different target

Security standards and control catalogs (e.g., NIST SP 800-53) define robust requirements for access control, authorization policy, and enforcement (e.g., “access enforcement” and “approved authorizations”

Stable Authority Boundary (SAB)…

These frameworks are essential, but they typically focus on *who may access resources* and *what operations are permitted* in an information security sense.

SAB overlaps conceptually but diverges in emphasis:

- Access control focuses on enforcing policy for resource access and information flows.
- SAB focuses on the *legitimacy of execution outcomes* in autonomy/distributed contexts where:
    - authority can be time-bounded (leases/TTL),
    - the relevant authority source may be unreachable,
    - signals conflict,
    - and “non-action” must be treated as a legitimate governed outcome rather than an error state.

In other words: access control tells you whether an identity is allowed to do X under policy; SAB tells you whether the system is permitted to **proceed at this decision boundary, now**, given authority validity, bounds, degraded coordination, and evidence quality.

Stable Authority Boundary (SAB)…

## 2.6 Governance, non-progress, and the "silence problem"

A subtle but critical gap in common engineering practice is how systems treat **non-progress**. Many systems encode "do nothing" as an implicit failure mode (timeouts, retries, suppressed outputs), rather than a governed, testable execution outcome. Yet under uncertainty, degraded coordination, or contest

Stable Authority Boundary (SAB)…

ng" can be the *correct* legitimacy-preserving behavior.

SAB formalizes non-progress outcomes as legitimate execution states:

- **Defer/Hold** when legitimacy might be established later (missing evidence, temporary dependency loss).
- **Refuse** when legitimacy cannot be established within bounds (invalid authority, out-of-bounds context).
- **Silence** where producing output would itself create illegitimate influence or escalation—*but only when silence is explicitly governed and auditable.*

  Stable Authority Boundary (SAB)…

This is not merely philosophical. In distributed and autonomous systems, "silent success" is dangerous: downstream components may interpret absence of an explicit refusal as implied permission, and operators may misread non-response as "system still running as intended." SAB therefore treats silence as something that must be **distinguishable from failure** and bac

Stable Authority Boundary (SAB)…

e declared conformance level.

Stable Authority Boundary (SAB)…

## 2.7 Positioning SAB relative to existing approaches

SAB can be positioned relative to four adjacent bodies of practice:

1. **Safety cases / assurance cases**
   SAB contributes a bounded, testable claim: “this system does not Act without demonstrable authority validity at execution time; otherwise it Defers/Refuses/Silences in governed ways.” It strengthens

   Stable Authority Boundary (SAB)…

   s by providing explicit evidence artifacts (authority map, seam register, conformance delta, reproducibility packet) that are repeatable and auditable.

   Stable Authority Boundary (SAB)…

2. **Access control and policy enforcement**
   SAB is compatible with access control mechanisms, but addresses broader authority semantics (scope, bounds, degraded-mode rules, conflict defaults) and treats action legitimacy—not resource access—as the primary invariant.
3. **Distributed consensus / coordination theory**
   Con

   Stable Authority Boundary (SAB)…

   tocols solve consistency and leadership ambiguity at the data/control-plane level. SAB operates one layer up: even if the cluster can commit consistently, SAB asks whether the commit was *legitimate under declared authority bounds* (e.g., still within approval TTL, still within mode constraints, still within mission governance).

4. **Human–automation governance and accountability**
   Literature on handoffs and governance emphasizes transitional accountability and diffusion risks. SAB operationalizes that insight as an engineering requirement: the authority basis must be explicit and verifiable at the point of execution, and the system must emit evidence sufficient for reconstruction.

## 2.8 Where existing standards “fall short” in SAB terms

From the SAB perspective, typical gaps are not that standards are “bad,” but that they optimize for different questions:

- **Correctness**: Does the system behave as specified?
- **Safety**: Are hazards identified and mitigated to acceptable risk?
- **Security**: Are access and flows controlled and audited?

SAB’s question is narrower and more ruthless:

- **Legitimacy**: Was the system permitted to proceed *at the decision boundary* given authority validity, bounds, conflict rules, degraded coordination constraints, and evidence sufficiency?

  Stable Authority Boundary (SAB)…

The difference is most visible under degraded coordination. Many regimes assume coordination dependencies (policy availability, supervisor reachability, quorum signals) as environmental preconditions. SAB treats loss of these dependencies as a **governance event** that must force a constrained outcome unless explicitly authorized otherwise.

## 2.9 Summary

This chapter established the terrain SAB is meant to operate in:

- Autonomy and distribution make implicit authority tempting and responsibility diffusion common.
- Degraded coordination is not rare; it is the adversary systems must be

  Stable Authority Boundary (SAB)…

  ithout drifting into illegitimate execution.

- Existing standards and assurance methods primarily verify correctness/safety/security, but often do not provide a minimal, repeatable, testable profile for legitimacy gating at execution time.
- SAB complements these approaches by focusing on one invariant and making it auditable: **no acting "as if authorized" when authority cannot be demonstrated as valid for the present context at execution time**.

  Stable Authority Boundary (SAB)…

# 3. Terms, definitions, and abbreviations

## 3.1 Terms and definitions

### 3.1.1 Authority

**Authority**: the **explicit, bounded, and verifiable** basis by which an actor (human or system component) is **permitted to execute** a defined action within a defined scope.

Authority is valid only when all of the following are satisfied:

- **Source** — the granting entity is identified (e.g., role, policy, controller, operator, quorum, contract)
- **Scope** — the action(s) authorized are specified.
- **Bounds** — the conditions under which the authority applies are specified (e.g., context, mode, dependencies).
- **Validity** — the authority has a defined validity condition (e.g., time-to-live, lease, revocation rule).
- **Verifiability** — the system can determine, at decision time, whether the authority is present and applicable.

**Note 1:** Authority is not "capability." A component may be capable of acting while lacking authority to act.
**Note 2:** Authority may be delegated, but delegation SHALL preserve source traceability and bounds.
**Note 3:** In this standard, authority that is implicit, assumed, or unverifiable at decision time is treated as **absent** for conformance purposes.
**Note 4:** For conformance purposes, if an item cannot be evidenced at the declared evidence level, it SHALL be treated as absent.

### 3.1.2 Permission to execute

**Permission to execute** is produced by evaluating the applicable **authority** against the current context and any required bounds.

Permission to execute exists only when:

- **Applicable authority is present** — an authority source covering the action is identified.
- **Authority is valid** — the authority satisfies its validity conditions (e.g., not expired or revoked).
- **Authority is within bounds** — the current context satisfies the authority's declared bounds (scope, mode, dependencies).
- **Conflicts are resolved** — any conflicting authority or signals are resolved according to the system's conflict rule.
- **Evidence is producible** — the system can produce evidence that the evaluation occurred and the permission outcome was selected.

**Note 1:** Permission to execute is **not** a static attribute. It is evaluated at the decision boundary and may change over time.
**Note 2:** If authority is absent, ambiguous, conflicting without a defined resolution, or unverifiable, permission to execute is treated as **not granted**.

### 3.1.3 Legitimacy / legitimate execution

**Legitimacy**: the property that an execution outcome (Act, Defer/Hold, or Refuse) is **authorized, bounded, and justified** under this standard.

**Legitimate execution**: an execution outcome is legitimate when, at the time of decision:

- **Authority basis exists** — the relevant authority source(s) and applicable scope are identified.
- **Permission rule is satisfied** — permission to execute is either **granted** (for Act) or **not granted** (for Defer/Hold or Refuse), consistent with the defined conflict/default rules.
- **Bounds are respected** — the outcome complies with declared bounds (context, mode, dependencies, validity).
- **Non-progress is governed** — if the outcome is Defer/Hold or Refuse, the non-progress state is treated as a valid state with defined triggers and (where applicable) exit/termination conditions.
- **Evidence exists** — the system can provide evidence sufficient to demonstrate the authority evaluation and outcome selection (per the conformance evidence level in scope).

**Note 1:** Legitimate execution includes **non-action** (Defer/Hold or Refuse) when authority is absent, ambiguous, expired, or conflicting.
**Note 2:** A technically correct or successful action may still be illegitimate if performed without valid authority.

### 3.1.4 Illegitimate execution

- **Illegitimate execution**: an action or outcome is illegitimate when the system **Acts** (or transitions into an action-producing state) **without legitimacy**, including any of the following conditions at decision time:
- **Authority absent** — no applicable authority source is identified for the action.
- **Authority invalid** — authority is expired, revoked, out of date, or otherwise not valid.
- **Out of bounds** — authority exists but the current context violates declared bounds (scope, mode, dependencies, constraints).
- **Unresolved conflict** — authority or signals conflict and the conflict-resolution rule is missing, ambiguous, or not applied.
- **Unverifiable authority** — the system cannot determine whether authority applies, yet proceeds to act.
- **Ungoverned transition** — the system moves from uncertainty/ambiguity directly to action without a governed Defer/Hold or Refuse pathway when permission is not clearly granted.

- **No evidence** — the system cannot produce required evidence that the authority evaluation occurred and justified the action (per the selected conformance evidence level).

**Note 1:** Illegitimate execution is a governance failure, not merely an operational or functional failure.
**Note 2:** A system may be "correct" in output and still be illegitimate if it acted without valid authority.

### 3.1.5 Stable Authority Boundary (SAB)

**Stable Authority Boundary (SAB)**: a **design invariant** and governance constraint requiring that an autonomous or distributed system **shall not Act** unless authority is **explicit, bounded, valid, and verifiable** at the decision boundary, and requiring that the system **shall Defer/Hold or Refuse** when permission to execute is not legitimately granted.

### *An SAB is* stable *when:*

- **Ability is evaluated and bounded** — the system defines how it determines whether it is **able to execute** the candidate action at the decision boundary (e.g., resources, dependencies, actuator availability, system health, connectivity), and it defines the legitimacy-preserving outcome when ability is insufficient (typically **Defer/Hold** or **Refuse**) with an evidenceable reason.
- **Authority is declared** — authority sources, scope, and bounds are explicitly stated.
- **Authority evaluation is defined** — the system defines where and how **permission to execute** is evaluated.
- **Non-progress states are governed** — **Defer/Hold** and **Refuse** are first-class outcomes with defined triggers and (where applicable) exit/termination conditions.
- **Degraded coordination is addressed** — the SAB remains enforceable when coordination is degraded (e.g., partial information, partition, conflicting signals).
- **Evidence is producible** — the system can produce conformance evidence that the SAB decision boundary evaluated **ability and authority**, and that the selected outcome was legitimacy-preserving.

**Note:** Authority does not compel action. If ability is insufficient, the system SHALL not be forced into execution; it SHALL enter a governed non-progress state with recorded reason, consistent with this standard.

**Implementation note:** SAB does not require central control. It requires that authority bounds and evaluation rules remain enforceable even when control is distributed.

**Reader note:** This standard defines the conformance profile, evidence, and validation method for SAB. The conceptual and motivating basis for SAB is developed in the supporting work listed

in the Normative References / Supporting Documents section (e.g., authority contraction/refusal, silence/failure modes, missing boundary, and engineering seams).

### 3.1.6 Boundary condition

**Boundary condition**: a declared condition that **constrains** whether an action may proceed, and that determines which legitimacy-preserving outcome applies (**Act**, **Defer/Hold**, or **Refuse**).

Boundary conditions SHALL be expressible in terms of:

a) **Context bounds** — system mode, operating environment, coordination state, dependency availability.
b) **Authority bounds** — scope, roles/policies, delegation limits, validity/expiry/revocation.
c) **Ability bounds** — feasibility constraints such as health, resources, actuator readiness, connectivity, safety interlocks.
d) **Evidence bounds** — minimum evidence required to justify action under the selected conformance level.

**Note:** A boundary condition is not merely a "precondition" for success; it is a legitimacy constraint. If the boundary condition is not met, the correct outcome may be non-progress.

### 3.1.7 Decision boundary

**Decision boundary**: the point in a system's behavior where it commits to an outcome that could change system state or the external world, and where SAB evaluation SHALL occur.

A decision boundary includes (but is not limited to):

a) transition from observation to action,
b) transition from plan to execution,
c) escalation from defer to act,
d) override, recovery, or reroute actions, and
e) any automatic action taken under degraded coordination or uncertainty.

**Note:** The SAB gate is evaluated at the decision boundary, not only at design time. A system may re-evaluate at multiple boundaries as conditions change.

### 3.1.8 Degraded coordination

**Degraded coordination**: a system condition in which the mechanisms normally used to coordinate authority, state, or action across actors are reduced, delayed, partitioned, inconsistent, or unavailable, such that a correct global view cannot be assumed at the decision boundary.

Degraded coordination includes (but is not limited to):

- **Partition** — actors cannot reliably communicate or reach required participants.
- **Delay / staleness** — coordination signals arrive too late to be trusted for the current decision.
- **Inconsistency** — different actors hold conflicting views of authority, state, or intent.
- **Loss of quorum / supervisor** — required coordination roles or approval sources are unreachable.
- **Partial information** — required evidence or state needed for permission evaluation is missing.

**Note:** Under degraded coordination, this standard treats authority and evidence that cannot be verified at the decision boundary as **not available** for conformance purposes, unless an explicit, bounded degraded-mode rule is declared.

### 3.1.9 Uncertainty / ambiguous input

**Uncertainty / ambiguous input**: a condition in which the system cannot determine, at the decision boundary, a single reliable interpretation of required inputs or signals needed to evaluate ability, authority, or permission to execute.

Uncertainty / ambiguous input includes (but is not limited to):

- **Missing input** — required input is absent or incomplete.
- **Low confidence** — input exists but fails an evidence/quality threshold (e.g., confidence score, checksum, provenance).
- **Conflict** — two or more inputs or signals relevant to the same decision are inconsistent.
- **Non-uniqueness** — multiple plausible interpretations exist and the system cannot disambiguate within defined bounds.
- **Staleness** — inputs are present but outdated beyond an allowed window for the decision.

**Note:** Under this standard, uncertainty and ambiguity are treated as first-class decision conditions that require a governed outcome (Act only if permission remains legitimately granted; otherwise Defer/Hold or Refuse).

### 3.1.10 Conflict (signal conflict, authority conflict)

- **Conflict**: a condition in which two or more relevant assertions required to determine the outcome at a decision boundary are mutually incompatible, such that a single legitimate outcome cannot be selected without an explicit resolution rule.

- **Signal conflict**: conflict between observations, measurements, state reports, or inferred intent used to determine ability to execute or the appropriate action (e.g., inconsistent sensor readings, inconsistent state estimates, competing control objectives).
- **Authority conflict**: conflict between authority sources, grants, constraints, or delegation statements relevant to permission to execute (e.g., two authorities grant incompatible actions; one grants while another forbids; overlapping scopes with different bounds; conflicting revocation/expiry status).

**Conflict resolution rule**: a declared rule that specifies how conflicts are detected, prioritized, and resolved, including the default legitimacy-preserving outcome when conflict cannot be resolved within bounds.

**Note:** If conflict exists and no applicable conflict resolution rule is declared, permission to execute is treated as **not granted** for conformance purposes.

### 3.1.11 Act

**Act**: the execution outcome in which the system performs the candidate action, producing an externally observable effect or committing the system to an action-producing state change.

An **Act** outcome is legitimate only when, at the decision boundary:

- **Ability to execute is satisfied** — the system determines it is able to perform the action within declared bounds.
- **Permission to execute is granted** — applicable authority is present, valid, in-bounds, verifiable, and any conflicts are resolved per the declared rule.
- **Evidence is producible** — the system can produce the required evidence that the above evaluations occurred and justified the Act outcome (per the selected conformance evidence level).

### 3.1.12 Defer / Hold

**Defer / Hold**: a governed non-progress execution outcome in which the system does not perform the candidate action now, and instead enters a bounded waiting/holding state while required conditions for legitimate execution are unresolved.

A **Defer / Hold** outcome SHALL define:

- **Entry trigger(s)** — the condition(s) causing defer (e.g., uncertainty, missing evidence, temporary inability, unresolved conflict where deferral is permitted).
- **Hold behavior** — what the system does while holding (e.g., maintain safe state, continue monitoring, request evidence/authority, retry strategy).
- **Exit condition(s)** — the condition(s) for transition from Defer/Hold to Act or Refuse.

- **Bound(s)** — one or more limits on deferral (e.g., TTL, lease duration, retry limit, escalation rule).
- **Evidence and notification** — the system SHALL record (or otherwise emit) evidence of (1) entry into Defer/Hold, (2) the reason/trigger, (3) the active bound (TTL/lease/retry), and (4) the exit decision (Act or Refuse) including the condition that caused the exit.
- **Authority handling while deferred** — the system SHALL declare what authority-dependent behavior is permitted during Defer/Hold (e.g., requesting additional authority/evidence, escalating to a human/quorum, waiting for coordination recovery) and SHALL NOT perform the deferred candidate action until permission to execute is legitimately granted.

**Note:** Defer/Hold does not grant or transfer authority. It transfers **responsibility for progress** to a declared resolution mechanism (e.g., additional evidence acquisition, coordination recovery, escalation to a human/quorum/controller, or expiry/timeout policy). Until that mechanism produces a legitimate permission-to-execute outcome, the deferred candidate action SHALL NOT be executed.

### 3.1.13 Refuse

**Refuse**: a governed non-progress execution outcome in which the system does not perform the candidate action and explicitly records refusal as the selected outcome with a reason.

A **Refuse** outcome SHALL include:

- **Refusal trigger** — the condition(s) that require refusal (e.g., authority absent/invalid/out of bounds/unverifiable; ability insufficient beyond allowable bounds; unresolved conflict that cannot be resolved within the defer policy).
- **Recorded reason** — an evidentiary reason code or statement sufficient to support audit and post-analysis.
- **Evidence and notification** — the system SHALL record (or otherwise emit) evidence that refusal was selected, including the decision boundary, time, and the authority/ability basis for refusal (per the selected conformance evidence level).
- **Post-refusal handling** — the system SHALL define what happens after refusal (e.g., safe-state maintenance, escalation path, retry prohibition/conditions, manual override path if permitted).

**Note:** Refuse is a legitimacy-preserving outcome. It is not a failure state by default; it is an explicit governance decision to not execute.

### 3.1.14 Silence (intentional non-output)

**Silence (intentional non-output)**: a governed execution outcome in which the system produces no external response or output for a decision boundary, and where that non-output is **intentional**, **bounded**, and **justified** under this standard.

Silence SHALL be treated as an explicit outcome and SHALL define:

- **When silence is permitted** — the conditions under which silence is allowed (e.g., to avoid escalation, to prevent unsafe action, to comply with authority constraints).
- **When silence is prohibited** — conditions requiring explicit Defer/Hold or Refuse instead of silence (e.g., when an operator or dependent system requires an explicit outcome).
- **Bound(s)** — maximum duration and/or retry/heartbeat rules, if silence can persist.
- **Auditability** — the system SHALL record (or otherwise emit) evidence that silence was selected as the outcome, including the decision boundary and reason, unless explicitly out of scope for the selected conformance evidence level.

**Note:** Silence is not "missing behavior." If silence is used, it SHALL be governed and evidentiary at the declared conformance level.

### 3.1.15 Conformance evidence / audit evidence

**Conformance evidence / audit evidence**: the minimum set of artifacts required to justify a conformance judgement under this standard, sufficient for an independent reviewer to determine whether the SAB gate and its outcomes were evaluated and selected legitimately.

Conformance evidence SHALL include:

- **Decision record** — evidence that the decision boundary was evaluated and what outcome was selected (Act, Defer/Hold, Refuse, or governed Silence).
- **Basis** — evidence of the authority/ability/constraint basis used to select the outcome (including conflict handling where applicable).
- **Traceability** — references that map evidence to applicable requirement IDs and to the audited scope and parameter set.
- **Integrity** — enough information to detect missing, substituted, or stale evidence within the declared evidence level.

**Note:** Evidence may be documentary, model-based, or runtime-generated depending on the declared conformance evidence level; if the required evidence cannot be produced, conformance SHALL NOT be claimed.

### 3.1.16 Engineering seam

**Engineering seam**: a detectable gap, ambiguity, contradiction, or implied decision boundary in a system's design, documentation, implementation, or operation where authority, ability, outcome selection, or evidence requirements are missing, inconsistent, or not governable.

Engineering seams include (but are not limited to):

- **Missing boundary** — no explicit rule exists for Act/Defer/Refuse/Silence under a relevant condition.
- **Implied authority** — authority is assumed or inferred but not declared or verifiable.
- **Contradicted behavior** — stated rules conflict with observed/documented behavior or with other rules.
- **Unguarded transitions** — movement from uncertainty/conflict/degraded coordination directly into Act without governed non-progress handling.
- **Non-auditable outcomes** — outcomes occur without evidence sufficient to justify legitimacy.

**Note:** Engineering seams are the primary findings produced by the SEAMS audit method and are used to drive conformance delta and corrective action.

### 3.1.17 Ability to execute (A2E)

**Ability to execute (A2E)**: the system's current operational feasibility to perform the candidate action at the decision boundary, given required resources, dependencies, and safety constraints.

A2E SHALL be determined using declared criteria, which may include:

- system health and integrity,
- actuator availability and readiness,
- required inputs and dependencies,
- communications/connectivity where required, and
- applicable safety interlocks.

**Note:** A2E may be false even when authority is valid; in such cases the legitimacy-preserving outcome is Defer/Hold or Refuse with recorded reason.

### 3.1.18 Time-to-live (TTL)

**Time-to-live (TTL)**: a declared bound on the validity or duration of a state, authorization, deferral, or evidence item, after which it SHALL be treated as expired unless explicitly renewed under the applicable rules.

**Note:** TTL may apply to authority validity, Defer/Hold duration, cached evidence staleness, or coordination assumptions.

### 3.1.19 Legitimate execution (LTE)

**Legitimate execution (LTE)**: execution outcomes (Act, Defer/Hold, Refuse, or governed Silence) that satisfy the legitimacy criteria defined in this standard, including authority/permission evaluation, ability constraints, and evidence requirements.

### 3.1.20 Illegitimate execution (ITE)

**Illegitimate execution (ITE)**: execution in which the system Acts (or commits toward action) without meeting the legitimacy criteria of this standard, including cases of absent/invalid/unverifiable authority, out-of-bounds operation, unresolved conflict, or insufficient evidence at the declared conformance level.

## 3.2 Abbreviations

- SAB — Stable Authority Boundary
- SEAMS — Engineering Seams (audit method)
- LTE — Legitimate execution
- ITE — Illegitimate execution
- P2E — Permission to execute
- A2E — Ability to execute
- DC — Degraded coordination
- TTL — Time-to-live4. Conformance model

# 4.0 Conformance model

## 4.1 Conformance targets

A conformance target **SHALL** be explicitly declared as one of the following:

- **System** (the complete integrated system within a declared operational context)
- **Subsystem** (a bounded portion of the system with defined interfaces and responsibilities)
- **Component** (a discrete software/hardware element that can be independently evaluated)
- **Interface / API boundary** (a decision boundary where actions are requested, authorized, or committed)
- **Operator workflow** (a human-led procedure that delegates authority to automation or triggers in-scope actions)

A conformance claim **SHALL** define the target boundary, including:
(a) external interfaces in-scope, (b) action classes in-scope (REQ-SCOPE-001), and (c) authority sources in-scope.

## 4.2 Conformance levels

This profile defines three conformance levels:

- **SAB-Core**: Minimum requirements for legitimate execution gating, governed non-progress states, and evidence sufficient for independent review.
- **SAB-Extended**: Adds stricter degraded-coordination constraints, explicit conflict handling, and stronger evidence traceability expectations.
- **SAB-Critical**: Intended for safety/mission/security critical contexts; requires runtime evidence and representative scenario execution.

A conformance claim **SHALL** state the level claimed. A claim at a given level **SHALL** satisfy all SHALL requirements designated for that level (see Appendix A).

## 4.3 Conformance statement format

A conformance claim **SHALL** include the fields required by Section 1.5 and CC-001 through CC-015, and **SHALL** be expressed in the following structured form (or an equivalent form with identical fields):

**Conformance Claim Statement (template)**

- Conformance target:
- Target identifier (build/version/hash):
- Declared conformance level (Core / Extended / Critical):
- Profile version:
- Declared scope:
    - Components/interfaces in-scope:
    - Action taxonomy classes in-scope:
    - Authority sources in-scope:
- Operational context assumptions:
- Known exclusions + rationale + compensating controls:
- Evidence package identifier + location:
- Test method execution record identifier:
- Assessment date + validity window / invalidation triggers:

## 4.4 Non-conformance

Non-conformance exists when any SHALL requirement applicable to the declared scope is not satisfied. A conformance claim **SHALL NOT** be made (or **SHALL** be withdrawn) if any of the following are true for the declared scope:

- The authority gate can be bypassed (REQ-SAB-009 violated).
- Authority is implicit/unverifiable at decision time yet the system Acts (LTE criteria violated; ITE condition).
- Governed non-progress states are not implemented or are not evidentiary (Defer/Refuse/Silence not testable).
- Required evidence artifacts are missing, not traceable to requirements, or fail integrity expectations at the declared level.
- The test method was not applied, is not repeatable, or is not recorded at the declared level.

When non-conformance is identified, the evaluation record **SHALL** include: affected requirement IDs, observed failure condition, impacted action classes, and disposition (fix / scope reduction / exclusion / claim withdrawal).

# 5. Authority model requirements

This section defines SHALL-level requirements for the authority model. An implementation conforms only if it can demonstrate explicit authority declaration, bounded applicability, validity/expiry semantics, explicit placement of the gate decision, and defined behavior under conflict.

## 5.1 Authority declaration

• REQ-SAB-001 (Authority Source): The system SHALL identify the authority source for each in-scope action (role, policy, controller, quorum, contract, or operator).

• REQ-SAB-002 (Action Scope): The system SHALL enumerate the action(s) each authority source permits, at a granularity sufficient for audit.

• REQ-SAB-003 (Subject/Actor Binding): The system SHALL bind each permitted action to the acting entity (component, service, agent, or operator identity).

What must be declared (sources, roles, scope)

## 5.2 Authority bounds

• REQ-SAB-004 (Context Bounds): The system SHALL declare the contextual conditions under which authority is valid (mode, environment, dependencies, invariants, and required signals).

• REQ-SAB-005 (Operating Mode Bounds): The system SHALL declare which operating modes permit execution and which force defer/refuse.

• REQ-SAB-006 (Dependency Bounds): The system SHALL declare required dependencies (e.g., supervisor availability, quorum, sensor health, policy reachability) that must be satisfied before action.

- Context bounds, operating mode bounds, dependency bounds

## 5.3 Validity and expiry

• REQ-SAB-007 (Validity Semantics): Authority SHALL include explicit validity semantics (lease/TTL, revocation rules, or equivalent).
• REQ-SAB-008 (Expiry Handling): When authority expires or is revoked, the system SHALL transition to a governed non-progress state (defer or refuse) for affected actions.

- Time semantics, lease/TTL, revocation

## 5.4 Decision placement

• REQ-SAB-009 (Gate Placement): The authority gate SHALL be placed at, or logically upstream of, the execution point such that actions cannot bypass the gate.
• REQ-SAB-010 (Gate Ownership): The system SHALL identify who/what owns the gate decision (component, controller, operator role) and SHALL make this auditable.

- Where the gate decision is made; who/what owns it

## 5.5 Conflict rules

• REQ-SAB-011 (Conflict Default): Under conflicting authority or signals, the default SHALL be non-progress (defer) unless an explicit higher-priority authority permits execution.
• REQ-SAB-012 (Priority Rules): The system SHALL define deterministic priority/precedence rules for authority conflicts, including tie-breaking behavior.

Default behavior under conflicting authority or signals

## 5.6 Ability to execute (A2E)

 REQ-SAB-013 (A2E Criteria Declaration): The system SHALL declare the criteria used to determine Ability to Execute for each in-scope action class.

- REQ-SAB-014 (A2E Evaluation Evidence): The system SHALL produce evidence that A2E was evaluated at the decision boundary for each Act outcome and for any Defer/Refuse outcome where A2E is cited as the basis.”

# 6. Execution-state requirements (Act / Defer / Refuse)

## 6.1 Permitted action (Act)

• REQ-EXE-001 (Act Preconditions): The system SHALL only act when authority validity is proven for the action in the current context and all required bounds are satisfied.
• REQ-EXE-002 (Act Traceability): Each act decision SHALL produce evidence linking (a) action, (b) authority source, (c) evaluated bounds, and (d) timestamp/version of policy.

- Preconditions for action

## 6.2 Defer / hold state

• REQ-EXE-003 (Defer Triggers): The system SHALL enter defer when authority is ambiguous, incomplete, expired, or when required dependencies are unsatisfied.
• REQ-EXE-004 (Defer Behavior): While deferred, the system SHALL preserve safety/mission state and SHALL attempt only bounded recovery actions explicitly permitted in defer mode.
• REQ-EXE-005 (Defer Exit Conditions): The system SHALL define explicit conditions for transition from defer to act or refuse, including timeouts and escalation rules.

- Required triggers
- Required behaviors while deferred
- Exit conditions (what allows transition to act/refuse)

## 6.3 Refusal state

• REQ-EXE-006 (Refuse Triggers): The system SHALL refuse when authority is invalid, explicitly denied, or when execution would violate declared bounds or invariants.
• REQ-EXE-007 (Refuse Behavior): On refusal, the system SHALL (a) not execute the action, (b) record the refusal reason, and (c) follow the defined escalation/notification path when required.

- Required triggers
- Required behaviors on refusal (safe state, escalation, etc.)

## 6.4 Silence as a governed state

• REQ-EXE-008 (Silence Governance): If silence (intentional non-output) is used, the system SHALL define when silence is allowed, required, or prohibited, and SHALL make silence distinguishable from failure to execute.
• REQ-EXE-009 (Silence Evidence): When silence is an outcome, the system SHALL record the decision and its justification as conformance evidence.

- When silence is allowed, required, or prohibited

## 6.5 Degraded coordination constraints

- REQ-EXE-010: Under degraded coordination, the system SHALL apply a *declared degraded-mode* rule set that reduces permitted action scope and/or increases required authority/evidence thresholds (e.g., shorter TTL, stronger proof, reduced action set). The system SHALL NOT broaden action permissions in degraded mode unless explicitly authorized and bounded.
  • REQ-EXE-011 (Coordination Loss Response): When coordination dependencies are lost, the system SHALL transition to defer or refuse for actions requiring coordination, unless explicitly authorized to continue autonomously within declared bounds.

# 7. Evidence and test method

This section defines the evidence artifacts and test method required to make conformance claims auditable. Evidence is not optional: conformance is demonstrated through artifacts that can be inspected by an independent reviewer.

## 7.1 Required evidence artifacts

• REQ-EVID-001 (Authority Map): The conformance package SHALL include an Authority Map that enumerates authority sources, scopes, bounds, validity semantics, and acting entities.
• REQ-EVID-002 (Seam Register): The conformance package SHALL include a Seam Register listing identified decision seams, their authority status (declared/implicit/missing), and associated risks.
• REQ-EVID-003 (Conformance Delta): The conformance package SHALL include a Conformance Delta showing each requirement and its satisfaction status with evidence pointers.
• REQ-EVID-004 (Reproducibility Packet): The conformance package SHALL include a reproducibility packet (inputs, configuration, versions, test scripts/procedures) sufficient to rerun the evaluation.

- Authority Map
- Seam Register
- Conformance Delta
- Reproducibility Packet
- (Optional) runtime logs/telemetry requirements

## 7.2 Evidence quality levels

• REQ-EVID-005 (Evidence Level Declaration): The conformance claim SHALL declare the evidence quality level used (document-only, traceable artifacts, runtime evidence).
• REQ-EVID-006 (Runtime Evidence for Critical): For SAB-Critical conformance, the evidence SHALL include runtime traces or logs demonstrating gate decisions in representative scenarios.

- Document-only vs traceable artifacts vs runtime evidence

## 7.3 Test method overview

• REQ-TEST-001 (Repeatable Evaluation): The evaluation SHALL be repeatable by an independent reviewer using the provided evidence and procedures.
• REQ-TEST-002 (Scenario Coverage): The evaluation SHALL include scenarios for uncertainty, degraded coordination, and conflict sufficient to exercise act/defer/refuse behavior.

- How conformance is evaluated at a high level

## 7.4 Engineering Seams audit method (normative reference)

• REQ-TEST-003 (SEAMS Procedure): The conformance evaluation SHALL apply the Engineering Seams audit procedure and record its parameters, findings, and disposition for each seam.
• REQ-TEST-004 (Parameter Recording): The evaluation SHALL record audit parameters (system boundary, action set, authority sources, coordination assumptions, and time semantics) used to produce findings.

- Point to the SEAMS Method Spec as the test procedure
- Define required parameter recording

---

# 8. Validation approach

## 8.1 Validation goals

- What “validated” means here (prevent illegitimate execution, preserve legitimacy under uncertainty)

## 8.2 Avoiding circular validation

- Principle statement (separation of forward path vs challenge path)

## 8.3 Red-team validation protocol

- Timing requirements (early + late)
- Independence requirements (who can/can’t be on the team)
- Minimum review rounds (e.g., 2 passes)

### 8.4 Disposition rules

- Every critique results in: revision / boundary condition / limitation / rejection with justification

### 8.5 Repeatability and variance controls

- How two auditors should converge; how disagreements are recorded

---

## 9. Governance and maintenance

### 9.1 Change control

- Versioning, change log requirements

### 9.2 Profile evolution

- How extensions/Appendix es are added without breaking core

### 9.3 Publication and review cycle

- How updates are proposed and validated

---

## Appendix

**Appendix A: SAB Core Conformance Requirements Table**

- Req IDs, SHALL statements, evidence required, pass/fail criteria

**Appendix B: Engineering Seams Method Spec**

- Full procedure, parameters, seam taxonomy, output formats

**Appendix C: Evidence Templates**

- Seam Register template
- Authority Map template
- Conformance Delta template
- Reproducibility Packet template

**Appendix D: Worked examples**

- Case 1: (documentary audit)
- Case 2: (degraded coordination scenario)
- Case 3: (conflict/silence scenario)

**Appendix E: Mapping to existing assurance frameworks**

- How SAB complements (not replaces) safety cases, STAMP, etc.

**Appendix F: Design patterns and anti-patterns**

- Common seam patterns, common fixes

# Appendix G: Research Framing and Open Questions (Informative)

## G.1 Purpose of this appendix

This appendix preserves the research framing that motivated the Stable Authority Boundary (SAB) profile without making the profile itself a research document. The normative portions of this profile define conformance requirements; this appendix provides context for readers evaluating SAB as a candidate basis for broader assurance practice, standardization, or future derived profiles.

## G.2 Research framing

SAB treats **execution legitimacy** as a first-class constraint: a system must not proceed as if authorized when authority cannot be demonstrated as valid for the proposed action in the present context at execution time. This profile operationalizes that constraint via SHALL-level requirements, evidence artifacts, and a repeatable test method.

The research contribution implied by SAB is not "how to optimize decisions," but **how to constrain decisions** under uncertainty, degraded coordination, and contested control—especially when non-progress outcomes (defer/refuse/silence) are the correct behavior.

## G.3 Evaluation question for reviewers (informative)

**EQ-F-001:** When presented with a proposed in-scope action under uncertainty, can the system:

1. demonstrate valid authority for the present context at execution time, and
2. if not, reliably select a governed non-progress outcome (defer/refuse/silence where applicable), and
3. produce evidence sufficient for independent reconstruction of the decision boundary and outcome?

This is an evaluation prompt, not a normative requirement.

## G.4 Open questions for research and standardization (informative)

### G.4.1 Authority semantics across domains

- How should "authority validity" be parameterized across domains (e.g., safety-critical vs enterprise IT vs mission operations) while preserving the same legitimacy invariant?
- What minimal authority metadata (scope, time bounds, revocation signals, escalation paths) is necessary to make authority verifiable in practice?

### G.4.2 Evidence sufficiency and quality levels

- What evidence quality levels are sufficient for different assurance contexts (internal review, external audit, certification support)?
- How should evidence integrity and retention requirements scale with criticality without turning the profile into a security standard?

### G.4.3 Degraded coordination and contested control

- What is the minimal "degraded-mode rule set" that prevents authority drift during partitions, quorum loss, or supervisory loss?
- How should conflicting commands or contested authority sources be represented and adjudicated at the decision boundary?

### G.4.4 Non-progress outcomes as legitimate behavior

- How should systems differentiate *defer* (waiting for legitimacy) from *refuse* (legitimacy cannot be established) in ways that are testable and non-ambiguous?
- When is "silence" a required legitimacy-preserving outcome rather than a failure or an implementation gap?

### G.4.5 Practical test method repeatability

- What minimum set of scenarios constitutes a representative test suite for SAB conformance in distributed systems (e.g., partial partitions, clock skew, stale policy, revoked credentials, conflicting operator intent)?
- How should SEAMS-style seam identification be standardized to reduce variance between auditors while remaining tool-agnostic?

### G.4.6 Composition and inheritance

- If a component is SAB-conformant, what can be inferred (or not inferred) about the conformance of the system that composes it?
- How should conformance be expressed for layered systems where different subsystems hold different authority scopes?

## G.5 Candidate research-to-standardization path (informative)

This profile can support further work in three directions:

1. **Derived domain profiles** (e.g., “SAB for mission operations,” “SAB for cloud control planes”) that constrain the same invariant with domain-specific authority sources and evidence expectations.
2. **Conformance tooling** (e.g., evidence pack schemas, seam register formats, automated gate verification checks) that improves repeatability and audit efficiency.
3. **Metrics and assurance integration** (e.g., mapping SAB evidence to safety case arguments or security control frameworks) without redefining those frameworks.

## G.6 Non-normative statement

Nothing in this appendix adds to, modifies, or overrides the normative requirements of this profile. It is provided solely for context, reviewer orientation, and future work planning.