Receipts, Not Promises

Fail-Closed Governance for High-Stakes AI

KINGDOM CONFORMANCE RFC V1.3.6 (PUBLIC DRAFT)

Author: Yong Bok Lee

Yong Bok Lee (publishing as Scott Lee)

Organization: Meridian Verity Group

Contact: MeridianVerity@proton.me

Date: January 2026

Status: Public Draft (Non-Confidential / Public-Safe)

Keywords: governance; conformance; deterministic validation; fail-closed; receipts; auditability; procurement; privacy-preserving verification

DOI: 10.5281/zenodo.18236114

Copyright and License

This document may evolve; conformance claims MUST bind to the specific RFC version, CanonicalFormID(s), and the referenced Policy Pack and Validator versions.

It is not legal advice and does not establish obligations unless incorporated by reference into a program, consortium agreement, or procurement instrument.

This document is a public draft specification intended for implementation, procurement readiness, and independent audit.

Status of This Memo

This document is a public draft specification and may change without notice. It does not create obligations unless incorporated by reference into an agreement or procurement instrument. Feedback is welcome; conformance claims MUST bind to the specific RFC version, CanonicalFormID(s), and the referenced Policy Pack and Validator versions.

Contents

Summary

Receipts, not promises. Kingdom Conformance turns admissibility into portable, replayable evidence and fail-closed enforcement at real control points for the people these systems affect.

Reader map (pick one path)

- Builders: Sections 4–9 (architecture, control points, validator + receipt contracts, conformance pack, privacy model)
- Operators: Sections 8–12 (levels, lifecycle governance, procurement knobs, adoption plan)
- Buyers/Auditors: Sections 7, 11, Appendix B (Conformance Statement, acceptance tests, findings taxonomy, sampling)

Guarantees and non-guarantees (avoid misinterpretation)

- This RFC makes conformance claims independently replayable; it does not guarantee semantic ground-truth correctness unless explicitly claimed.
- PASS authorizes only a declared action class and scope under a specific policy version and freshness bounds; it is not a blanket approval.
- HOLD is a safety decision: it blocks sensitive side effects until evidence is refreshed or clarified.
- This RFC is vendor-neutral and avoids prescribing specific cryptography or operational secrets.
- This RFC is public-safe by design (no exploit guidance, no real keys/hashes, no confidential implementation details).
- IP/data-rights and research-security language is conceptual only (Appendix A) and is not legal advice.

Abstract

This RFC defines a vendor-neutral governance substrate for high-stakes AI work: Policy Packs (versioned, machine-readable admissibility rules), Deterministic Validators (PASS/FAIL/HOLD outcomes with reason codes), Fail-Closed Gates at real control points, and Replay-Verifiable Receipts that make conformance claims independently auditable. The operational claim is narrow and buildable: any governed "data -> train -> evaluate -> deploy -> operate" step can emit a portable Conformance Pack that a third party can replay to confirm admissibility. When evidence is missing, stale, inconsistent, or insufficient to decide deterministically, the correct outcome is HOLD, and sensitive downstream actions are fail-closed by default. This RFC is intended for environments such as scientific computing, healthcare, critical infrastructure, financial systems, and other high-stakes AI deployments where AI-assisted actions can significantly affect safety, rights, or large-scale systems.

The substrate is designed for heterogeneous environments (cloud, HPC, on-prem), multi-organization collaboration, and privacy-preserving workflows by relying on evidence handles and selective disclosure rather than broad payload disclosure. Conformance Levels (L0-L3) reduce adoption friction: teams can start with receipts, add replay verification, enforce gates at execution boundaries, and reserve higher-assurance privacy and sampling audits for controlled or restricted workflows.

The result is procurement-ready governance: buyers can require portable artifacts and acceptance tests instead of narrative assurances.

Key Points

- Replace policy PDFs with Policy Packs that machines can evaluate deterministically.
- Require validator outcomes PASS/FAIL/HOLD with machine-readable reason codes; treat HOLD as fail-closed for sensitive actions.
- Emit Replay-Verifiable Receipts for each governed stage (DATA, TRAIN, EVAL, DEPLOY, OPERATE, TRANSFER).
- Place Fail-Closed Gates at side-effect control points: compute admission, registry/promotion, transfer/egress, externalization, deployment changes.
- Mint short-lived Permits only after PASS; validate permits at gates to reduce time-of-check/time-of-use drift.
- Package artifacts into a Conformance Pack with a Conformance Statement declaring coverage and level claims.
- Use L0-L3 levels plus entry criteria to prevent badge inflation and self-attestation.
- Govern self-improving systems via candidate -> approved -> revoked, with drift triggers and revalidation.

1. Problem and why now

High-stakes AI work fails in a repeatable way: organizations cannot prove that execution occurred only under admissible conditions, and they cannot reliably prevent non-admissible actions before side effects occur. Post-hoc logs and narrative attestations do not satisfy independent audits, procurement, or multi-party collaboration.

Why now: AI pipelines are increasingly continuous (self-improving), multi-tenant, and multi-party across cloud and HPC. Governance must be portable, machine-checkable, and enforceable at runtime control points, not reconstructed after incidents.

What success looks like

- Sensitive side effects do not occur without a verifiable Permit issued after PASS for the declared scope.
- Audits become replay exercises over Conformance Packs, not interviews and screenshots.
- Conformance claims remain stable across tool/model churn because receipts bind decisions to policy versions, canonical forms, and freshness bounds.

2. Scope, non-goals, and design principles

Scope

A minimal governance substrate adopted across tools and organizations to (a) make admissibility replay-verifiable, (b) fail closed on missing/stale prerequisites, and (c) support procurement and third-party audits using portable artifacts.

Non-goals

- Not a model architecture proposal; does not prescribe training methods.
- Not an identity system; consumes identity assertions but does not define them.
- Not logging, but better; receipts are verification artifacts with deterministic replay semantics.
- Not legal advice; IP/data-rights and research-security language is conceptual only (Appendix A).
- Not a cryptography standard; does not prescribe constructions or operational secrets.

Design principles

- Determinism: same declared inputs -> same decision; ambiguity returns HOLD.
- Replayability: independent parties reproduce checks using Conformance Packs + evidence handles.
- Fail-closed enforcement: side effects blocked unless prerequisites validate under current Policy Packs.
- Portability: wrapper-style integration; avoid platform replacement.
- Privacy-preserving verification: verify predicates using evidence handles, not broad payload disclosure.
- Adoption-first: L0-L3 plus minimum viable integration points and phase gates.

3. Core concepts and definitions (testable, non-proprietary)

Normative terms: The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 (RFC 2119 and RFC 8174) when, and only when, they appear in all capitals, as shown here. This document uses normative keywords sparingly to define interoperable contracts; implementations may vary in internal design, provided they satisfy the normative requirements and preserve deterministic replay outcomes under declared inputs.

Policy Pack: Versioned, signed, machine-readable ruleset defining admissibility for a workflow stage, action class, and scope, with signed change history.

Deterministic Validator: Deterministic evaluator over (Receipt + referenced evidence handles + policy version + freshness bounds) returning PASS/FAIL/HOLD with reason codes.

PASS / FAIL / HOLD: PASS: admissible for a declared action class and scope under referenced policy version and freshness bounds. FAIL: non-admissible for declared action class and scope (policy violation). HOLD: missing/stale/inconsistent evidence or inability to decide deterministically. HOLD is fail-closed for sensitive side effects.

Reason Code: Machine-readable identifier explaining why a decision occurred (stable meaning per validator/policy version).

CanonicalFormID: Identifier for the canonicalization ruleset used to serialize/sign receipts and validator outputs for replay.

Replay-Verifiable Receipt: Signed, content-addressed record of a governed event such that independent verification can reproduce the same PASS/FAIL/HOLD decision using a Conformance Pack plus authorized access to evidence handles.

Evidence Handle: Stable reference/digest/redacted bundle/selective disclosure handle sufficient to verify predicates without broad payload disclosure.

Permit: Short-lived, audience-bound authorization minted only after PASS; validated at control points.

Fail-Closed Gate: Enforcement point placed where side effects happen; blocks execution unless a valid Permit is verified.

Conformance Pack: Minimal portable bundle sufficient for independent replay verification (auditor mode).

Conformance Statement: Small signed manifest included in a Conformance Pack declaring: claimed conformance level (L0-L3), covered receipt types and control points, applicable Policy Pack(s) and Validator(s), CanonicalFormID(s), registry snapshot reference(s) (if used), audit window (or selection rule), retention and replay expectations, declared scope-of-use, and explicit out-of-scope exclusions.

Side Effect: An externally observable action that changes state outside the validator (e.g., compute admission, artifact promotion, transfer/egress, externalization, deployment changes, or enabling external tools).

4. Architecture overview (artifacts, control points, bindings)

4.1 Control points (where enforcement applies)

Fail-Closed Gates are placed where side effects happen. Control points include: (1) compute admission, (2) artifact registry and promotion, (3) transfer and egress, (4) externalization boundaries, and (5) deployment change control.

For L2 claims, enforcement MUST cover at least compute admission and transfer/egress within the declared scope in the Conformance Statement. If registry/promotion, externalization, or deployment changes occur within the declared scope, those control points MUST be permit-gated and evidenced by receipts as well.

4.2 Artifact <-> control point bindings (minimum)

- Compute admission -> Permit validation evidence + TRAIN/OPERATE receipt(s).
- Registry/promotion -> integrity/provenance evidence handle(s) + DEPLOY receipt(s).
- Transfer/egress -> scope + policy + freshness evidence + TRANSFER receipt(s).

- Externalization -> explicit export policy + permit evidence + TRANSFER/OPERATE receipt(s).
- Deployment change -> permit evidence + DEPLOY/OPERATE receipt(s).

4.3 Receipt types and what they claim

- DATA: dataset admissibility under policy version + freshness bounds.
- TRAIN: compute/training admissibility for declared scope + inputs.
- EVAL: evaluation reproducibility for pinned inputs + scoring rules.
- DEPLOY: artifact promotion and deployment change admissibility.
- OPERATE: runtime operation admissibility (tool enablement, config changes, actions).
- TRANSFER: cross-boundary transfer/egress/externalization admissibility.

4.4 High-level flow (public-safe)

```
Policy Pack -> Validator

Receipt + Evidence Handles -> PASS/FAIL/HOLD (+ reason codes, scope, freshness
window)

If PASS -> Permit (scoped, time-bounded)

Control Point Gate validates Permit -> side effect allowed (FAIL/HOLD -> blocked)

Conformance Pack bundles artifacts + Conformance Statement -> auditor-mode replay
```

Implementation note (public-safe): start by emitting receipts (L0), add replay verification (L1), then enforce gates at compute admission and transfer/egress (L2). Avoid platform replacement.

5. Deterministic validator semantics

5.1 Output contract (minimum)

A validator output MUST include:

- Decision: PASS | FAIL | HOLD
- Reason codes (one or more)
- Policy Pack identifier/version
- Validator identifier/version
- Evaluated action class and scope
- Decision timestamp
- Declared freshness window (or freshness bounds) used for the decision

5.2 Determinism boundary (non-negotiable)

Determinism is defined over declared inputs referenced by the Receipt: policy version, validator version, CanonicalFormID (or equivalent), evidence handles (including registry snapshot references if used), and freshness bounds. If required inputs are unstable or unavailable, the validator returns HOLD rather than guessing. During replay, freshness predicates MUST be evaluated against the receipt's recorded decision timestamp (or an explicitly recorded evaluation_time field), not the verifier's wall-clock time.

5.3 Canonicalization (interoperability)

Receipts and validator outputs SHOULD be signed over a canonical form so equivalent-but-different encodings do not break determinism across implementations. Canonicalization rules MUST be versioned and referenced by policy/validator versions.

For L1+ conformance claims, receipts and validator outputs MUST carry a CanonicalFormID (or equivalent) that identifies the canonicalization ruleset used for signature verification and field normalization in replay.

5.4 Reason code stability

Reason codes SHOULD be stable identifiers with consistent meaning for a given policy/validator version to prevent rename-away failures. For L1+ claims, Conformance Packs SHOULD include a versioned snapshot reference (or equivalent) for the Reason Code Registry used to interpret reason codes; if the snapshot is missing or unverifiable, the safe outcome is HOLD with a distinct reason code (e.g., REGISTRY_SNAPSHOT_MISSING).

6. Replay-verifiable receipts

6.1 Receipt types (minimal enum)

Receipt type MUST be one of: DATA | TRAIN | EVAL | DEPLOY | OPERATE | TRANSFER.

6.2 Minimal receipt fields (public-safe)

At minimum, a receipt SHOULD commit to:

- Receipt type (DATA/TRAIN/EVAL/DEPLOY/OPERATE/TRANSFER)
- Declared action class and scope (what the claim applies to)
- Policy Pack id/version
- Validator id/version
- CanonicalFormID (canonicalization ruleset identifier)
- Decision + reason codes
- Evidence handles for inputs/outputs (stable references or digests)
- Timestamps + declared freshness bounds
- Actor/environment identifiers for attribution (pseudonymous allowed)
- Signature/integrity binding for the receipt itself

Replay verification establishes conformance to pinned requirements; it does not automatically assert semantic correctness unless explicitly claimed.

7. Conformance Pack (portable unit of procurement and audit)

A Conformance Pack MUST include a Conformance Statement declaring what is claimed and what is covered.

Minimum contents:

- Conformance Statement (level, coverage, policy/validator versions, CanonicalFormID(s), registry snapshot reference(s) if used, declared scope, audit window (or selection rule), retention/replay expectations, out-of-scope exclusions)
- Policy Pack id/version + signed change history
- Validator id/version + output contract + published test vectors covering PASS/FAIL/HOLD outcomes (including negative and freshness-boundary cases). For L1+ claims, an independent verifier SHOULD be able to run these vectors and reproduce expected decisions using only the Conformance Pack plus authorized evidence handles.
- Receipts in scope (DATA/TRAIN/EVAL/DEPLOY/OPERATE/TRANSFER as applicable)
- Evidence handles required for replay (digests/references/redactions/selective disclosure handles)
- For L2+ claims: gate/permit validation evidence (or side-effect reconciliation evidence handles) sufficient to show that in-scope side effects did not occur without a valid, in-scope Permit minted after PASS and validated at the relevant control point(s) within the audit window.
- For evaluation claims: reproducibility bundle (pinned inputs by handle + scoring rules + harness instructions)

Coverage declaration: the Conformance Statement MUST declare which stages, control points, and audit window (or selection rule) are covered so procurement and audit are unambiguous.

8. Conformance Levels (L0-L3) + entry criteria (anti-badge-inflation)

Default guidance: require L1 for contributors, L2 for shared execution paths, and L3 for controlled/restricted workflows.

L0 - Record - Signed Policy Pack identifiers + signed receipts for key stages. No runtime gating required.

- Entry criteria: receipts exist and signatures validate for declared receipt types.

L1 - Verify - L0 + deterministic validator outputs + Conformance Packs replayable by an independent verifier.

- Entry criteria: independent replay succeeds for declared receipt types, policy versions, and freshness bounds.

L2 - Enforce - L1 + fail-closed gates at compute admission and transfer/egress (minimum), permits minted only after PASS and validated at gates.

- Entry criteria: receipts demonstrate enforcement at compute admission and transfer/egress, and at any additional control points declared in the Conformance Statement for the claimed scope.

L3 - High assurance - L2 + privacy-preserving verification for controlled/restricted scope, independent sampling audits, continuous revalidation with expiry/revocation.

- Entry criteria: evidence handling supports restricted scope without broad payload disclosure, sampling audits run, and lifecycle revalidation is in place.

9. Privacy-preserving verification (buildable, non-prescriptive)

Principle: verify predicates, not payloads.

Predicate catalog (concept)

Policy Packs SHOULD define explicit predicates for controlled/restricted workflows (eligibility, approvals, residency/export constraints, integrity checks). Each predicate declares acceptable evidence handle types and freshness requirements.

Freshness (first-class)

Receipts MUST declare freshness bounds for prerequisites that can change. If evidence is missing or stale, the validator returns HOLD; sensitive actions fail closed until refreshed.

This RFC does not specify cryptographic constructions or operational secrets. Implementations may vary as long as replay outcomes remain deterministic under declared inputs.

10. Operational model for self-improving systems (candidate -> approved -> revoked)

States

- Candidate: produced by governed pipeline; not authorized for sensitive use.
- Approved: authorized for a defined action class and scope under a specific policy version.
- Revoked: approval removed; sensitive use blocked by gates.

Required receipts (minimum)

- Approval receipt: Candidate -> Approved transition (PASS prerequisites + declared scope).
- Revocation receipt: Approved -> Revoked transition (triggered by FAIL/HOLD revalidation, policy invalidation, or drift).

Expiry and revalidation triggers (categories)

- Time-based expiry (scheduled revalidation).
- Policy changes affecting declared scope.
- Evidence freshness boundary breach.
- Drift indicators (buyer/operator-defined).

Unresolved prerequisites yield HOLD and fail-closed behavior for sensitive actions.

Break-glass (if allowed) MUST be explicitly policy-scoped and receipted, and SHOULD trigger post-incident replay and audit sampling.

11. Procurement language (paste-ready) + optional SLAs + control-point acceptance

Illustrative; adapt to environment and risk tolerance.

R1 Artifact deliverables

- Requirement: Supplier SHALL deliver Policy Packs, validator outputs, receipts, permits, and Conformance Packs.
- Acceptance: independent verifier parses artifacts, validates signatures, confirms required fields.

R2 Deterministic validator contract

- Requirement: Supplier SHALL implement PASS/FAIL/HOLD with stable reason codes and declared scope.
- Acceptance: published vectors reproduce expected outcomes, including negative and freshness-boundary cases.

R3 Replay-verifiable receipts

- Requirement: Receipts SHALL be replayable under referenced policy/validator versions.
- Acceptance: independent replay reproduces decisions using only Conformance Pack + authorized evidence handles.

R4 Fail-closed gates at control points

- Requirement: Side-effect actions SHALL be blocked unless a valid Permit is verified at required control points (within declared coverage).
- Acceptance: Per control point: compute admission blocks on missing/stale prerequisites; registry/promotion blocks on missing integrity evidence; transfer/egress and externalization block on missing scope/export preconditions; deployment change blocks on missing permits. Demonstrate deterministic blocking on HOLD/FAIL.

R5 Permit discipline

- Requirement: Permits SHALL be minted only after PASS; permits SHALL be scope- and time-bounded.
- Acceptance: sampling audits show no side effects in scope without valid permits.

R6 Auditor-mode verification

- Requirement: Solution SHALL support auditor-mode replay without privileged access to raw restricted payloads.
- Acceptance: auditor completes replay using Conformance Packs + authorized handles.

R7 Retention & reproducibility

- Requirement: Supplier SHALL retain Conformance Packs for an agreed period and support replay of evaluation claims.
- Acceptance: buyer replays defined sample within retention window.

R8 Optional operational SLAs (buyer-selected knobs)

- Requirement: Buyer MAY specify targets for revocation propagation latency, permit validation availability, receipt retention duration, and audit cadence/sample size.
- Acceptance: periodic drills validate targets and fail-closed behavior under degraded conditions.

12. Adoption plan (phase gates + role map; no timeline guesses)

- Phase 0 - L0 Record: Exit criteria: receipts exist and signatures validate for declared receipt types.
- Phase 1 - L1 Verify: Exit criteria: independent replay succeeds for declared scope under policy/validator versions and freshness bounds.
- Phase 2 - L2 Enforce: Exit criteria: control points in declared scope are permit-gated; side effects fail closed on HOLD/FAIL.
- Phase 3 - L3 High assurance: Exit criteria: restricted workflows verify predicates without broad payload disclosure; sampling audits + lifecycle revalidation operate continuously.

Roles (minimum)

- Policy Author (policy versions + changelog).
- Validator Owner (deterministic behavior + vectors + reason code registry).
- Platform Owner (gates + permit validation at control points).
- Auditor/Verifier (sampling plan + replay process).

Minimum viable integration points (MVIPs)

- Compute admission hook.
- Registry/promotion hook.
- Transfer/egress gate.
- Deployment change gate.
- Policy Pack registry + receipt store.

13. FAQ (buyer/security/operator objections)

Is this just logging?

No. Receipts are replayable verification artifacts; logs alone are not.

Does PASS mean safe?

No. PASS means admissible for a declared action class/scope under specified policy and freshness bounds.

What if evidence is missing or stale?

HOLD; fail-closed for sensitive actions until evidence is refreshed.

Will this slow research?

Use levels. L0/L1 can be lightweight; enforce gates only where side effects justify L2/L3.

Multi-cloud/HPC portability?

Designed for wrapper-style hooks at common control points.

Cost/latency concerns?

Start at L0/L1; constrain L2/L3 to high-risk scopes; SLA knobs are buyer-selected.

Break-glass operations?

If allowed, it MUST be explicitly policy-scoped and receipted, and it SHOULD trigger post-incident replay and audit sampling.

Does this require sharing weights/data?

Default is evidence handles; raw disclosure only if policy requires.

Who owns policies?

The Policy Author role; this RFC requires versioning and replayable change history.

Is this legal advice?

No. Appendix A is conceptual only.

Implementation experience / evidence (informative): The Zenodo deposit alongside this RFC includes (S2) a reference verifier and published test vectors (PASS/FAIL/HOLD, negative + freshness-boundary cases, and registry-missing/control-point coverage probes), (S6) a one-page interoperability summary with reproducibility pointers and run-log hashes, (S8) an independent verifier used to cross-check deterministic replay, and (S9) registry stewardship/change control guidance. These supplements are non-normative; they exist to make the specification directly testable by reviewers, buyers, and auditors. Zenodo DOI: 10.5281/zenodo.18236114.

14. Interoperability and registry considerations

14.1 Interoperability objective

This section defines minimal registries and extension rules that preserve determinism and fail-closed behavior without prescribing implementation internals.

Interoperability is the point: a Conformance Pack SHOULD be replay-verifiable by an independent verifier across organizations, platforms, and vendors.

14.2 Registries (conceptual; program-managed)

- Fail-closed handling of unknowns: when a verifier cannot decide deterministically, it returns HOLD.
- Comparable procurement and audit results across suppliers via Conformance Statements, test vectors, and coverage metrics.
- Deterministic replay of PASS/FAIL/HOLD outcomes given the same declared inputs and freshness bounds.

14.3 Extension and compatibility rules (fail-closed by design)

- Policy Pack Profile Registry: defines reusable policy-pack profiles (domain or program-specific) that specify required predicates, receipt types, and control points for a scope.
- Evidence Handle Type Registry: defines acceptable evidence-handle forms (e.g., digest, reference, registry snapshot reference, redacted bundle, selective-disclosure handle) and how they satisfy predicates.
- Reason Code Registry: defines stable reason-code identifiers for each validator/policy version; meanings MUST NOT change without a version bump. For replay, the registry SHOULD be referenceable via a versioned snapshot handle.
- Action Class Registry: defines stable action-class identifiers used in receipts and permits (recommended: short, dot-delimited tokens).
- Control Point Registry: defines control points where permits are validated (e.g., compute admission, registry/promotion, transfer/egress, externalization, deployment change).
- Receipt Type Registry: defines allowed receipt types and their meaning (e.g., DATA, TRAIN, EVAL, DEPLOY, OPERATE, TRANSFER).

To keep semantics stable while allowing evolution, deployments SHOULD maintain signed registries (or equivalent) for the identifiers below. Registries MAY be maintained by a consortium, a program office, or a supplier, as long as they are versioned and replay-stable. For L1+ claims, Conformance Packs

SHOULD include registry snapshot reference(s) (as evidence handles) for any registry required to interpret required identifiers in scope.

14.4 Interoperability test vectors (minimum)

- Editorial clarifications that do not change semantics SHOULD be PATCH version bumps.
- Additive changes that preserve replay outcomes (e.g., new optional reason codes or new predicates that are not required for an existing profile) SHOULD be MINOR version bumps.
- Canonicalization rules MUST be versioned and referenced; a change that can alter replay outcomes for the same declared inputs requires a MAJOR version bump of the affected contract (policy and/or validator).
- Parsers SHOULD be forward-compatible: ignore unknown optional fields, but treat missing required fields as HOLD/FAIL per policy.
- If a verifier cannot interpret a required identifier (policy, validator, receipt type, predicate, or evidence-handle type) needed to decide, it MUST return HOLD (not PASS).

15. Security considerations

This RFC is designed to make admissibility enforceable and auditable. The considerations below help adopters preserve fail-closed behavior and replay integrity without prescribing implementation internals.

Fail-closed on unverifiable artifacts

If Policy Packs, Receipts, Permits, or evidence handles cannot be verified as authentic and complete under the declared versions and freshness bounds, the safe outcome is HOLD (or FAIL when policy requires).

Bind decisions to scope and time

PASS is scoped. Permits SHOULD be time-bounded, audience-bound, and scope-bound, and MUST be validated at control points to reduce time-of-check/time-of-use drift.

Protect against substitution and drift

Receipts SHOULD bind declared inputs and evidence handles strongly enough that a verifier can detect substitution (e.g., swapping an evidence handle or policy version). When required inputs change or expire, validators return HOLD and gates fail closed until refreshed.

Break-glass is not a bypass

If emergency override is allowed, it MUST be explicitly policy-scoped and receipted, and it SHOULD trigger post-incident replay and audit sampling.

Key and access hygiene (non-prescriptive)

This RFC does not prescribe cryptographic constructions or key management. Implementers should protect signing materials and access control systems such that unauthorized parties cannot mint permits or forge receipts.

Threat model snapshot: forged receipts/permits; stale/substituted evidence beyond freshness bounds; TOCTOU drift.

Appendix A: Conceptual IP/data-rights + research-security operationalization (non-legal)

Conceptual program-design language only; not legal advice.

- A1 Artifact classes: Open Governance Artifacts / Audit Artifacts / Background IP / Sensitive Program Data (rights apply separately).
- A2 Open Governance Artifacts should be broadly usable for interoperable conformance testing.
- A3 Audit Artifacts shareable in privacy-preserving forms without forced payload disclosure unless explicitly required.
- A4 Background IP retained; no compelled release of weights/secrets unless explicitly negotiated.
- A5 Research-security constraints (eligibility/residency-export/access scoping) expressible as machine-checkable preconditions where feasible; missing/stale evidence -> HOLD until refreshed.

Appendix B: Audit checklists, findings taxonomy, negative cases, sampling plan (public-safe)

B1 Minimum Conformance Pack checklist

- Conformance Statement (level, coverage, policy versions, scope, retention/replay expectations, out-of-scope exclusions).
- Policy Pack id/version + signed changelog.
- Validator id/version + output contract + test vectors (including negative + freshness-boundary).
- Receipts in scope (DATA/TRAIN/EVAL/DEPLOY/OPERATE/TRANSFER).
- Evidence handles required for replay.
- Evaluation reproducibility bundle (if evaluation claims included).

B2 Auditor-mode checklist (offline replay)

- Verify policy version resolution and changelog authenticity.
- Verify receipt signatures and completeness.
- Replay decisions under declared inputs/freshness bounds.
- Confirm decision matches recorded PASS/FAIL/HOLD + reason codes.
- Confirm required control points were permit-gated for declared scope.
- Record findings; missing/stale prerequisites treated as HOLD for sensitive actions.

B3 Audit finding taxonomy

- F1 Structural (missing fields, unverifiable signatures, malformed artifacts).
- F2 Replay mismatch (replay decision differs under declared inputs).
- F3 Coverage gap (claimed stages/control points not evidenced).
- F4 Freshness breach (stale beyond declared bounds).
- F5 Enforcement failure (side effect without valid permit in scope).
- F6 Reason-code instability (meaning changes without version bump).

B4 Minimum negative cases (public-safe categories)

- Missing evidence handle -> HOLD/FAIL per policy.
- Stale evidence beyond freshness bound -> HOLD.
- Policy version mismatch -> FAIL/HOLD.
- Integrity evidence missing at promotion -> HOLD/FAIL.
- Permit missing at gate -> blocked.
- Attempted side effect outside scope -> FAIL.

B5 Sampling plan (stage- and control-point-balanced)

- Sample across receipt types and control points; always include negative cases and freshness-boundary cases.
- Require remediation for F2/F5/F4 findings before sensitive operation continues in scope.
- Track coverage metrics (B7) by receipt type x control point x scope to drive adoption priorities.

B6 Conformance Statement template (public-safe)

Use this fill-in template to declare conformance claims unambiguously (coverage, scope, and exclusions) without disclosing sensitive payloads or implementation secrets.

Field	Value (fill-in; public-safe)
Conformance Statement ID	<identifier>
Issuer	<organization / team / service>
RFC version	Kingdom Conformance RFC v1.3.6
Claimed conformance level	L0 | L1 | L2 | L3
Declared action class	<action class label(s)>
Declared scope-of-use	<scope boundaries (tenant/project/env)>
Covered receipt types	List: DATA, TRAIN, EVAL, DEPLOY, OPERATE, TRANSFER
Covered control points	List: compute admission; registry/promotion; transfer/egress; externalization; deployment change
CanonicalFormID(s)	<canonical form ruleset ID(s)/version(s)>
Registry snapshot reference(s)	<evidence handle(s) for versioned registry snapshots (if used)>
Applicable Policy Pack(s)	<policy IDs/versions>
Applicable Validator(s)	<validator IDs/versions>
Freshness assumptions	<what changes + max age/bounds>
Audit window	<start/end or selection rule>
Retention window for pack	<duration; buyer-defined>
Replay expectation	Replay with Conformance Pack + authorized evidence handles.
Out-of-scope exclusions	<explicit exclusions>
Signatory	<name/role>
Signature reference	<verification reference (no secrets)>
Issued date	<YYYY-MM-DD>

Note: This template is intentionally implementation-agnostic. It should reference evidence handles and versions, not disclose raw sensitive payloads, operational secrets, or proprietary internals. The audit window defines the time range over which coverage evidence is asserted or sampled; it is distinct from the retention window and freshness bounds.

B7 Coverage metrics (receipt type x control point x scope)

- Coverage unit: (receipt type, control point, declared scope).
- Coverage claim: the Conformance Statement declares the set of covered units within scope.
- Coverage evidence: receipts and gate evidence exist for covered units during the declared audit window.
- Coverage rate (conceptual): |covered units with replay-verifiable evidence| / |covered units claimed|.
- For L2+ claims, enforcement coverage requires permit-gated control points for all covered side effects in scope.

- Minimum reporting: counts by receipt type and control point, plus negative-case and freshness-boundary case counts.
- Use: procurement milestones, adoption prioritization, and audit sampling targeting.

B8 Auditor Playbook: Aggressive Audit Gauntlet (informative; public-safe)

Also published as Supplement S1 (Auditor Playbook PDF) alongside this RFC's Zenodo deposit.
Zenodo DOI: 10.5281/zenodo.18236114
Purpose: a high-stringency, buyer/auditor checklist intended to falsify (not merely confirm) L1+ replay claims and L2+ enforcement claims within a declared audit window.

Inputs (minimum):

- Conformance Statement(s) for the claimed scope and audit window.
- A sample of Conformance Packs within scope, plus authorized access to referenced evidence handles.
- Validator test vectors for the claimed validator versions (including negative and freshness-boundary cases).
- For L2+ claims: a side-effect inventory for the audit window (compute admissions, promotions, transfer/egress, externalizations, deployment changes) sufficient to reconcile permits/receipts to observed actions.

Gauntlet checks (attempt to fail the claim):

- G0 Conformance Statement completeness and scope boundaries are unambiguous. (Failing this is F1 Structural.)
- G1 Coverage reconciliation: every claimed coverage unit (receipt type × control point × scope) has replay-verifiable evidence in the audit window. (F3 Coverage gap.)
- G2 Offline replay: an independent verifier reproduces recorded PASS/FAIL/HOLD and reason codes under the pinned policy/validator versions. (F2 Replay mismatch.)
- G3 Freshness boundary: stale/expired prerequisites deterministically yield HOLD (or FAIL if policy requires) and do not allow sensitive side effects. (F4 Freshness breach; F5 if enforcement fails.)
- G4 Side-effect reconciliation (L2+): no side effect in scope occurs without a valid, in-scope Permit minted after PASS and validated at the relevant control point. (F5 Enforcement failure.)
- G5 Fail-closed drills (L2+): at each covered control point, missing/unknown/unverifiable prerequisites block the action (no fail-open on errors or degraded conditions). (F5 Enforcement failure.)
- G6 Change-control integrity: policy/validator/CanonicalFormID version resolution and signed change history are replay-stable; required registry snapshot references are present and verifiable; ambiguous identifiers yield HOLD, not PASS. (F2/F6.)
- G7 Reason-code stability: reason-code meanings do not change without a policy/validator version bump, and the applicable Reason Code Registry snapshot reference (or equivalent) is available for audit. (F6 Reason-code instability.)
- G8 Revocation propagation (L3 or if claimed): revoked approvals become operationally effective at gates within the buyer-selected revocation latency objective. (F5.)

Notes: (1) This playbook is informative and implementation-agnostic; it does not prescribe cryptography or internal controls. (2) Treat unverifiable artifacts as HOLD/FAIL per policy; do not "paper over" missing evidence. (3) Map findings to F1–F6 for comparability across suppliers.

Appendix C: Example Conformance Pack (illustrative; public-safe)

This appendix is illustrative and non-normative. All identifiers and values are placeholders. It demonstrates what an auditor could receive without disclosing sensitive payloads or implementation secrets.

C.1 Illustrative pack layout (non-normative)

An illustrative Conformance Pack can be represented as a portable bundle with the following conceptual contents:

conformance-pack/

conformance-statement (signed manifest: level, scope, coverage, exclusions)

policy-packs/ (referenced Policy Pack versions + signed changelog)

validator/ (validator contract + published test vectors)

receipts/ (DATA/TRAIN/EVAL/DEPLOY/OPERATE/TRANSFER receipts in scope)

evidence-handles/ (handles and retrieval instructions; no raw restricted payloads by default)

C.2 Example Conformance Statement (filled; placeholders)

Example (values are placeholders):

```
Signature reference: <verification handle; no secrets>
Issued date: <YYYY-MM-DD>
```

Audit window: <YYYY-MM-DD to YYYY-MM-DD or selection rule>

```
Out-of-scope exclusions: <explicit exclusions>
Replay expectation: replay using this pack + authorized evidence handles
Retention window for pack: <buyer-defined>
Freshness assumptions: <what changes + max age/bounds>
```

CanonicalFormID(s): CF-EXAMPLE-01@1.0.0

Registry snapshot reference(s): EH:REGISTRY_SNAPSHOT:EXAMPLE

```
Applicable Validator(s): V-EXAMPLE-01@1.0.0
Applicable Policy Pack(s): PP-EXAMPLE-01@1.0.0
Covered control points: compute admission; transfer/egress
Covered receipt types: DATA, TRAIN, EVAL, DEPLOY, OPERATE, TRANSFER
Declared scope-of-use: <tenant/project/env boundaries>
```

```
Declared action class: compute.admission; transfer.egress

Claimed conformance level: L2
```

RFC version: Kingdom Conformance RFC v1.3.6

```
Issuer: Example Organization / Service

Conformance Statement ID: CS-EXAMPLE-0001
```

C.3 Example Receipt (skeleton; placeholders)

Example receipt skeleton (fields are illustrative; implementations may add more):

```
Receipt signature: <verification handle; no secrets>

Decision timestamp: <timestamp>
```

CanonicalFormID: CF-EXAMPLE-01@1.0.0

Evidence handles:

- registry.snapshot: EH:REGISTRY_SNAPSHOT:EXAMPLE

```
  - output.model:  EH:MODEL_ARTIFACT:EXAMPLE

  - input.code:    EH:SOURCE_SNAPSHOT:EXAMPLE

  - input.dataset: EH:DATASET_SNAPSHOT:EXAMPLE
```

Freshness bounds: <declared bounds>

```
Reason codes: POLICY_OK; EVIDENCE_FRESH

Decision: PASS

Validator: V-EXAMPLE-01@1.0.0

Policy Pack: PP-EXAMPLE-01@1.0.0

Declared scope: <scope boundary string>

Action class: compute.admission

Receipt type: TRAIN
```

C.4 Example test vectors (categories)

At minimum, publish test vectors that include:

- Negative cases (missing evidence handle, policy mismatch, permit missing at a gate).
- Freshness-boundary cases (evidence stale beyond declared bounds).
- Non-determinism detection cases (unstable or unavailable required inputs -> HOLD).

Appendix D: One-page adoption checklist (public-safe)

This checklist is intentionally platform- and vendor-agnostic. It is meant to be used as a quick readiness and auditability gate.

D.1 Builder/operator checklist (by conformance level)

L0 (Record)

- [ ] Policy Pack identifiers are versioned and recorded for governed stages.
- [ ] Receipts are emitted for declared receipt types; signatures validate.
- [ ] Receipt store exists with retention policy (buyer-defined).

L1 (Verify)

- [ ] Deterministic validator outputs PASS/FAIL/HOLD with reason codes, scope, and freshness bounds.
- [ ] Conformance Packs are produced for declared scope; independent replay succeeds.
- [ ] Validator test vectors exist (PASS/FAIL/HOLD), including negative and freshness-boundary cases.

L2 (Enforce)

- [ ] Fail-Closed Gates are deployed at compute admission and transfer/egress (minimum) for the declared scope.
- [ ] Permits are minted only after PASS and are time- and scope-bounded.
- [ ] Demonstration: side effects in scope do not occur without valid permits; HOLD blocks.

L3 (High assurance)

- [ ] Restricted workflows verify predicates via privacy-preserving evidence handles (no broad payload disclosure by default).
- [ ] Independent sampling audits run continuously; findings taxonomy (F1-F6) is used.
- [ ] Lifecycle governance is operational: candidate -> approved -> revoked with expiry/revalidation.

D.2 Buyer/auditor checklist (procurement-ready)

- [ ] Contract requirements include artifact deliverables (Policy Packs, Receipts, Permits, Conformance Packs).
- [ ] Acceptance tests require replay verification and fail-closed gating demonstrations at declared control points.
- [ ] Conformance Statements declare coverage units (receipt type × control point × scope) and explicit exclusions.
- [ ] Audit cadence and retention windows are specified; degraded-mode drills confirm fail-closed behavior.

Versioning and citation

Version: v1.3.6

DOI: 10.5281/zenodo.18236114

Citation: Yong Bok Lee. "Receipts, Not Promises: Fail-Closed Governance for High-Stakes AI (Kingdom Conformance RFC v1.3.6)." Meridian Verity Group, January 2026.
DOI: 10.5281/zenodo.18236114

Changelog

v1.3.6: Clarified BCP14 normative terms usage; anchored replay freshness evaluation to receipt timestamps; added audit window to Conformance Statement definition/template and examples; added explicit L2+ gate/permit validation evidence to minimum Conformance Pack guidance; removed two dangling editorial lines in Section 14; publish-quality license wording; editorial-only clarifications.

v1.3.5: Made CanonicalFormID a first-class replay binding; added CanonicalFormID and registry snapshot reference fields to Conformance Statement template; added registry snapshot hooks for reason-code stability; aligned one-page templates to RFC version; editorial-only.

v1.3.4: Added Appendix B8 (Auditor Playbook: Aggressive Audit Gauntlet) and published it as Supplement S1 (PDF) for Zenodo attachments; editorial polish.

v1.3.3: Editorial-only release. Minor grammar fixes; standardized appendix headings; clarified R5 permit wording; improved Appendix C formatting; updated template version references.

v1.3.2: Editorial-only release. Filled Status of This Memo; fixed Section 14 ordering; updated version references.

- v1.3.1: Added Security considerations section; updated RFC version references; editorial polish.
- v1.3.0: Standardization edition. Added Status of This Memo and Copyright/License; added interoperability and registry considerations; added Appendix C example Conformance Pack and Appendix D adoption checklist; version bump and editorial polish.
- v1.2.3: Added high-stakes context sentence to Abstract; clarified series naming; editorial polish.
- v1.2.2: Added Conformance Statement template and coverage metrics guidance (Appendix B); improved auditor comparability; editorial polish.
- v1.2.1: Clarified Reader map section references; clarified L2 minimum control points and coverage semantics; added explicit out-of-scope exclusions to the Conformance Statement definition; added externalization to acceptance tests; editorial tightening.

Appendix E: Schemas / Registries Roadmap

Published as Supplement S5 for Kingdom Conformance RFC v1.3.6. Implementer-facing interoperability roadmap (informative; adds no requirements). Aligned to Sections 5 (determinism), 7 (Conformance Packs), and 14 (registries).

Goal: Make cross-vendor replay practical by publishing a minimal, versioned set of machine-readable schemas and signed registries. An independent verifier should be able to parse a Conformance Pack, resolve required identifiers via registry snapshots, and deterministically reproduce PASS/FAIL/HOLD outcomes.

E1 Minimal artifact schemas (recommended publication set)

- Conformance Statement (kc.conformance_statement@1): level; declared scope and action class; covered receipt types and control points; audit window (or selection rule); bindings (Policy Pack / Validator / CanonicalFormID, plus optional registry snapshot refs); explicit exclusions; signature reference.
- Receipt (kc.receipt@1): receipt_type; action_class; scope; Policy Pack id/version; Validator id/version; CanonicalFormID; decision + reason_codes; evidence_handles; decision timestamp + declared freshness bounds; signature reference.
- Permit (L2+) (kc.permit@1): permit_id; minted_after (receipt/decision reference); scope + audience; issued_utc + expires_utc; signature reference.
- Policy Pack metadata (kc.policy_pack@1): policy_id; version; declared profile/predicates; signed change history reference; optional human-readable summary.
- Validator contract metadata (kc.validator_contract@1): validator_id; version; output contract; reason code registry handle (if published); declared determinism boundary; test vectors reference (including negative and freshness-boundary cases).
- Evidence-handle record (kc.evidence_handle@1, optional): digest/reference type; redaction/selective-disclosure descriptor; retrieval instructions (public-safe); access note; redaction and retention policy.

E2 Minimal registries (IANA-style: versioned, signed, replay-stable)

- Receipt Type Registry: allowed receipt types and meanings (DATA, TRAIN, EVAL, DEPLOY, OPERATE, TRANSFER).
- Control Point Registry: where permits are validated (compute admission; registry/promotion; transfer/egress; externalization; deployment change).
- Action Class Registry: stable action-class identifiers used in receipts and permits (recommended: short, dot-delimited tokens).
- CanonicalFormID Registry: canonicalization rulesets used for deterministic replay and signature verification.

E2 Minimal registries (continued)

- Evidence Handle Type Registry: acceptable evidence-handle forms and how they satisfy predicates.
- Reason Code Registry: stable reason-code identifiers for each policy/validator version; meanings must not change without a version bump.
- Policy Pack Profile Registry: program/domain profiles that declare required predicates, receipt types, and control points for a scope.

For L1+ claims, Conformance Packs should include registry snapshot references (as evidence handles) for any registry required to interpret identifiers in scope. If a verifier cannot resolve a required identifier, the safe outcome is HOLD (not PASS).

E3 Versioning and compatibility (recommended)

- Semantic versioning for registries/schemas: PATCH = editorial clarifications; MINOR = additive, backwards compatible; MAJOR = changes that can alter replay outcomes for the same declared inputs (including canonicalization).
- Never change the meaning of an existing identifier without a version bump. Prefer adding new identifiers and marking old ones deprecated.
- Parsers should ignore unknown optional fields, but treat missing required fields (per schema/profile) as HOLD or FAIL per policy.
- During replay, freshness predicates should be evaluated against the receipt's recorded decision timestamp (or explicitly recorded evaluation_time), not the verifier's wall-clock time.

E4 Reviewer fast checks (approval-route accelerators)

- Schemas: a verifier can validate Conformance Packs against the published schemas with no privileged payload access.
- Registries: required registry snapshots resolve; unknown required identifiers deterministically yield HOLD.
- Replay: offline replay reproduces recorded PASS/FAIL/HOLD and reason codes under pinned Policy Pack / Validator / CanonicalFormID.
- Negative vectors: missing snapshot, stale evidence beyond freshness bounds, and policy/version mismatch produce HOLD or FAIL as specified.
- Fail-closed posture: when evidence is missing, stale, inconsistent, unverifiable, or insufficient to decide deterministically, HOLD blocks sensitive side effects.

Contact: MeridianVerity@proton.me | Status: Public Draft (public-safe) |