Meridian Verity Group

Evidence-first • Fail-closed • Audit-grade

Kingdom Conformance Standard (KCS)

Procurement-grade, offline-verifiable conformance artifacts for AI safety and governance.

Version	v1.0.0
Revision	R3 (Registry taxonomy + Supplier form + Practical vectors)
Release date	2026-02-12
Canonical URL	https://meridianverity.com/standards/
Zenodo DOI	10.5281/zenodo.18623859
Status	Public-safe draft; normative language for interoperability (not a certification).

Boundary: This document defines interoperability and conformance semantics for evidence packs, registries, and verifier behavior. It is not a compliance certification.

Table of contents

0. Status of this document

This is the **public-safe** release of KCS v1.0.0. The canonical web surface is https://meridianverity.com/standards/.

This Zenodo deposit provides the stable, mirrorable artifacts referenced by the specification (registries, schemas, templates, and test vectors). DOI: **10.5281/zenodo.18623859**.

KCS is designed for procurement and security diligence. It provides **deterministic PASS/FAIL/HOLD** outcomes, stable reason codes, and portable packs that buyers can verify **offline** (no uploads, no network).

Non-binding notice: KCS provides procurement-grade evidence artifacts. It is not a compliance certification or warranty.

1. Scope

KCS defines:

- A portable **Evidence Pack** directory layout and hashing rules.
- A **Conformance Pack** contract: tests, expected outcomes, and replayable semantics.
- **Registry snapshots** (trust anchors, reason codes, control points) pinned by content-addressed IDs.
- Verifier requirements for offline, buyer-run verification and **Auditor Mode** fail-closed checks.
- Supplier-facing **Conformance Statements** suitable for procurement submission.

KCS does **not** define model training, model architecture, or product UX. It defines the evidence and verification interface between suppliers and buyers.

2. Design goals

- **Offline verifiable**: buyers can verify packs without vendor network access.
- **Deterministic**: given pinned baselines and inputs, verifiers reproduce the same outcomes.
- **Fail-closed**: uncertainty yields HOLD (never silent PASS).
- **Procurement-shaped**: artifacts attach cleanly to tickets, audits, and acceptance criteria.
- **Mirrorable**: packs can be mirrored; integrity is protected by hashes and signatures.

3. Normative language and profiles

The key words **MUST**, **SHOULD**, and **MAY** are to be interpreted as described in RFC 2119 and RFC 8174.

Profiles constrain otherwise-ambiguous choices (crypto, canonicalization, privacy). Profiles MUST be declared and pinned for offline verification.

4. Terminology

Term	Definition
Evidence Pack	A portable directory (or ZIP) containing artifacts and a manifest of SHA-256 digests.
Conformance Pack	An Evidence Pack plus test vectors and expected outcomes that a verifier can replay.
Verifier Kit	The offline toolchain used by buyers/auditors to validate packs and reproduce outcomes.
Registry Snapshot	A content-addressed snapshot of a registry (e.g., reason codes) pinned by digest.
PASS / FAIL / HOLD	Deterministic outcomes. HOLD is the fail-closed result for missing, stale, unsupported, or unverifiable prerequisites.

5. Conformance levels (L0–L3)

Level	Meaning
L0	Receipts + manifest integrity + stable reason codes + pinned versions.
L1	Buyer-run offline verification + conformance packs (tests + expected outcomes).
L2	Permit-before-commit gates at declared control points; enforcement evidence in packs.
L3	Anti-equivocation signals (witness/transparency) + sampling + privacy/redaction profiles.

A supplier MAY claim multiple levels for different scopes, but each claim MUST be accompanied by a Conformance Statement declaring scope, profiles, and pins.

6. Evidence Pack format

An Evidence Pack is a directory (or ZIP) that MUST contain:

- **manifest.json** listing relative file paths and SHA-256 digests (excluding itself).
- **registry-snapshots/** containing pinned registry snapshots required to interpret artifacts (at minimum: trust anchors and reason codes).
- Artifacts under **artifacts/** including receipts, permits (if used), and audit evidence.

Packs SHOULD include **SHA256SUMS** for mirroring and human verification, but verifiers MUST treat **manifest.json** as the primary integrity source.

All verification MUST be possible offline. Any reference to external URLs MUST be treated as informational only and MUST NOT be required to determine PASS/FAIL/HOLD.

6.1 Manifest rules

For each file declared in the manifest, the verifier MUST recompute SHA-256 over the file bytes and compare to the declared digest.

If any digest mismatches, the verifier MUST output **FAIL** with reason code **DIGEST_MISMATCH**.

If the manifest is missing or invalid, or required files are missing, the verifier MUST output **HOLD** (fail-closed) with the appropriate reason codes (e.g., MANIFEST_MISSING, ARTIFACT_MISSING).

7. Conformance Pack contract (tests, outcomes)

A Conformance Pack is an Evidence Pack that includes a verifier contract plus test vectors and expected outcomes.

At L1+, a pack MUST include:

- **verifier-contract/verifier_contract.json** describing the abstract checks and control mappings.
- **verifier-contract/test_vectors.json** listing fixtures and expected PASS/FAIL/HOLD outcomes.
- **verifier-contract/expected_outcomes.json** providing a deterministic expected result map.

Test vectors MUST be replayable offline and MUST result in deterministic outputs under pinned baselines.

7.1 Practical verifier vectors (public-safe)

This Zenodo deposit includes a practical set of fixture packs and expected outcomes in **verifier_contract/**.

Example test vector entry:

```
{
  "vector_id": "TV-002",
  "description": "Tampered policy bundle without manifest update; digest mismatch is tamper-evident.",
  "fixture": {
    "path": "verifier_contract/fixtures/KCS_SAMPLE_PACK_TV-002_FAIL_DIGEST_MISMATCH_v1.0.0.zip",
    "sha256": "<sha256 of the fixture zip>"
  },
  "expected": {
    "outcome": "FAIL",
    "reason_codes": ["DIGEST_MISMATCH"]
  }
}
```

Verifiers MUST implement fail-closed semantics: when prerequisites are missing, stale, unsupported, or unverifiable, output HOLD (never silent PASS).

8. Verifier Kit (offline) requirements

- MUST run offline (no network required; no uploads).
- MUST support pinned registry snapshots and pinned schema bundles by digest.

- MUST provide Auditor Mode that treats missing/unsupported/stale prerequisites as HOLD.
- MUST emit deterministic PASS/FAIL/HOLD with stable reason codes.
- SHOULD support batch verification for mirrored packs (ZIP or directory).

9. Permits and fail-closed gates

At L2+, suppliers claim that certain control points are enforced by **permit-before-commit** gates.

A permit is an offline-verifiable authorization artifact. For in-scope actions at a declared control point:

- If no permit is present, the gate MUST deny and the verifier MUST output FAIL with PERMIT_MISSING.
- If permit is expired or not yet valid, output FAIL with PERMIT_EXPIRED (or equivalent).
- If permit scope does not match the action/control point, output FAIL with PERMIT_SCOPE_MISMATCH.
- If permit authenticity cannot be established offline, output HOLD (fail-closed).

Permits and enforcement evidence MUST be included in conformance packs such that buyers can reproduce enforcement decisions offline.

10. Commitments, witnesses, and anti-equivocation (optional)

At L3, suppliers MAY provide anti-equivocation signals so buyers can detect inconsistent claims over time.

Mechanisms include witness signatures (threshold) and/or transparency receipts. If claimed, required signals MUST be present and verifiable offline; otherwise HOLD.

11. Key management (keyrings, rotation, deprecation)

Trust anchors used for signing receipts and permits MUST be declared in a pinned trust anchor registry snapshot.

Registries MUST support key rotation and deprecation without silent breakage:

- Keys MUST carry validity windows (not_before/not_after).
- Revocation MUST be expressible in snapshots (revoked=true).
- Verifiers MUST treat unknown or revoked keys as non-authentic (HOLD or FAIL as appropriate).

12. Auditability, retention, and sampling

Packs MUST be self-contained enough to support offline audits within a declared retention window.

If sampling is used (L3), the sampling policy MUST be declared and reproducible offline.

13. Security considerations

- Fail-closed semantics (HOLD on uncertainty) reduces unsafe false PASS.
- Digest mismatch and signature invalidity are treated as tamper-evident FAIL.
- Registry snapshot pinning prevents downgrade and semantics drift.

14. Privacy considerations

KCS is compatible with public-safe publication and with NDA/limited disclosure modes.

If a privacy profile claims redaction/selective disclosure, redaction rules MUST be declared and verifiable offline; otherwise HOLD.

15. Registries

Registries standardize cross-vendor interpretation. Registry snapshots are pinned by digest and are treated as part of the conformance contract.

15.1 Registry snapshot requirements

- Each snapshot MUST include a content-addressed snapshot_id (e.g., nvrs:sha256:...).
- Receipts and conformance statements MUST reference snapshots by snapshot_id (not mutable URLs).
- Registry governance MUST define change control, deprecation policy, and reason-code lifecycle.

15.8 Reason Code Registry (standards-body profile)

Reason codes are a stable API. They MUST be machine-readable, versioned, and pinned for offline procurement interpretation.

This release upgrades the Reason Code Registry to include:

- **Code token naming rules** (collision-safe, no silent renames).
- **Category taxonomy** for consistent reporting.
- **Priority classes** (P0..P4) for primary-cause selection.
- **Normalization policy** for deterministic reason_codes ordering.

See: **KCS_reason_code_registry_v1.0.0_PUBLIC_SAFE.json** and **policies/KCS_REASON_CODE_REGISTRY_POLICY_v1.0.0.md** in this deposit.

Vendor-specific extensions MUST use the prefix **VENDOR_<ORG>_** and MUST NOT replace standard codes.

15.9 Conformance Statement (supplier submission form)

A Conformance Statement is a supplier declaration of scope, claimed level, profiles, and pins. It is designed to be attached to procurement artifacts.

This release upgrades the statement into a supplier submission schema with required fields and L-level conditionals.

See: **schemas/kcs_conformance_statement.schema.json** and the templates/samples under **conformance_statement/**.

16. Specification governance (this document)

The specification is governed like a standards-track document: changes are reviewed, versioned, and published with explicit deprecation policy.

Registry governance and change proposals are included in this deposit (see KCS_REGISTRY_GOVERNANCE.md and KCS_RCP_TEMPLATE.md).

Appendix A. Reference directory layout (non-normative)

```
kcs-pack/
  manifest.json
  SHA256SUMS
  registry-snapshots/
    trust_anchor_registry.json
    reason_code_registry.json
    control_point_registry.json        (L2+)
  policy-bundle/
    policy_bundle.json                 (optional)
  verifier-contract/
    verifier_contract.json             (L1+)
    test_vectors.json                  (L1+)
    expected_outcomes.json             (L1+)
  artifacts/
    receipts/
      receipts.jsonl
      receipts.dsse.json               (example signature envelope)
    permits/                           (L2+)
    audit-evidence/                    (optional)
```

Exact paths are implementation-defined, but packs MUST be self-contained for offline verification, and MUST include a manifest of digests.

Appendix B. Minimal procurement acceptance checklist

- Supplier provides a Conformance Statement with scope + pins (digest identifiers).
- Buyer can verify pack integrity offline using manifest.json and SHA-256 recomputation.
- Buyer can authenticate receipts offline using pinned trust anchors.
- Reason codes are stable and interpretable offline under pinned registry snapshot.
- For L2+: permit-before-commit enforcement evidence is present and replayable.
- For L3: anti-equivocation signals + sampling policy are present and reproducible.

Appendix C. Example outcome / reason codes

Outcomes are PASS / FAIL / HOLD. Reason codes MUST be stable tokens pinned by the reason code registry snapshot.

```
PASS: ["PASS_CONFORMANT"]

FAIL (tamper-evident): ["DIGEST_MISMATCH"]
FAIL (signature invalid): ["SIGNATURE_INVALID"]
FAIL (baseline mismatch): ["BASELINE_MISMATCH"]

HOLD (fail-closed): ["MANIFEST_MISSING"]
HOLD (missing registry snapshot): ["REGISTRY_SNAPSHOT_MISSING"]
HOLD (unsupported profile): ["CRYPTO_PROFILE_UNSUPPORTED"]
```

Appendix D. Registry Change Proposal (RCP) template (non-normative)

See KCS_RCP_TEMPLATE.md in this deposit. RCPs are used to add or deprecate reason codes, update control points, or rotate trust anchors.

Appendix E. Reason Code Registry taxonomy and normalization (normative)

Receipts and reports MUST normalize reason codes for determinism.

Normalization:

- Deduplicate codes.
- Order by priority (P0..P4), then lexicographic order.
- Primary reason code is the first after normalization.

Category taxonomy and mandatory interoperability sets are declared in the registry policy and snapshot included in this deposit.

Appendix F. Supplier Conformance Statement form (normative)

Suppliers MUST submit a statement containing identity, scope, level, profiles, pins, verifier kit details, retention, and security contact.

Validate against schemas/kcs_conformance_statement.schema.json. See completed examples in conformance_statement/.

Appendix G. Practical verifier test vectors (non-normative)

This deposit includes portable fixture packs for replayable verification (verifier_contract/fixtures).

Vector	Expected	Fixture
TV-001	PASS — PASS_CONFORMANT	KCS_SAMPLE_PACK_TV-001_PASS_v1.0.0.zip
TV-002	FAIL — DIGEST_MISMATCH	KCS_SAMPLE_PACK_TV-002_FAIL_DIGEST_MISMATCH_v
TV-003	HOLD — MANIFEST_MISSING	KCS_SAMPLE_PACK_TV-003_HOLD_MANIFEST_MISSING_
TV-004	FAIL — SIGNATURE_INVALID	KCS_SAMPLE_PACK_TV-004_FAIL_SIGNATURE_INVALID_
TV-005	HOLD — REGISTRY_SNAPSHOT_MISSING	KCS_SAMPLE_PACK_TV-005_HOLD_REGISTRY_SNAPSHO
TV-006	FAIL — BASELINE_MISMATCH	KCS_SAMPLE_PACK_TV-006_FAIL_BASELINE_MISMATCH_
TV-007	HOLD — CRYPTO_PROFILE_UNSUPPORTED	KCS_SAMPLE_PACK_TV-007_HOLD_CRYPTO_PROFILE_U