LARRI
VERIFICATION PROTOCOL

Ledger of Attested Records and Receipts for Integrity

Volume IV of V

LARRI Conformance and Release Specification

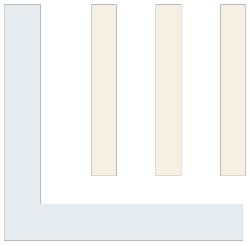
Version 0.1.0

Purpose. Defines black-box implementation conformance, exact evidence identity, frozen expected-byte comparison, objective publication gates, conformance reports, and release-envelope assembly for LARRI - Ledger of Attested Records and Receipts for Integrity.

Property	Value
Document type	Normative conformance and release specification
Publication set	LARRI 0.1.0, Volume IV of V
Protocol	LARRI - Ledger of Attested Records and Receipts for Integrity
Version	0.1.0
Status	Experimental
Use boundary	Synthetic laboratory and supervised evaluation only
Publication date	27 July 2026
Official website	https://larriverification.org

Assurance boundary. A Protocol Conformance PASS establishes only that one identified implementation reproduced the required public LARRI behavior for one exact reported scope. It does not establish artifact truth, human or organizational identity, authority, completeness, independently trusted time, substantive correctness, legal sufficiency, regulatory compliance, production readiness, or correctness of a private decision process.

Normative specification



Contents

Document relationship. Volume I owns assurance meaning, trust boundaries, verification targets, and the eleven-stage dependency order. Volume II owns exact artifact, trust, configuration, refusal, result, and canonical-byte contracts. Volume III owns optional profile and extension semantics. This volume owns conformance evidence, execution, claims, reports, publication gates, and release-envelope behavior. Volume V is informative.

1. Authority, scope, and conformance principle

LARRI conformance is determined through observable behavior against one exact semantic-contract identity and one exact conformance-evidence identity. An implementation conforms only for the targets, profile contexts, extensions, result contracts, implementation subject, platform scope, and vectors expressly identified by its report.

Internal engineering choices are outside ordinary conformance. A conforming implementation may be open or closed source and may be delivered as a library, binary, service, appliance, embedded component, or other evaluable form, provided the harness can control every protocol-visible input and compare every required output.

Independently checked means checked separately from the implementation under test. It does not require another organization, an outside reviewer, or a live service.

LARRI-010-CONF-SCOPE-001	A conformance claim MUST identify the exact scope tuple defined in Section 2 and MUST report PASS only for that tuple.
LARRI-010-CONF-SCOPE-002	A narrower target, profile, extension, result-contract, implementation-subject, platform, or evidence result MUST NOT be represented as satisfying a broader scope.
LARRI-010-CONF-SCOPE-003	A Protocol Conformance report MUST be FAIL when the selected normative sources prescribe conflicting behavior for the same input and scope.
LARRI-010-CONF-SCOPE-004	Every normative requirement definition in this volume MUST use one globally unique ASCII identifier beginning with LARRI-010-CONF-. A duplicate definition is a publication failure.

1.1 Observable behavior

Observable conformance includes input acceptance and refusal, target and profile context, exact identity availability, check status, primary-refusal selection, sibling-refusal order, limitations, actual trust use, canonical outputs, invocation-refusal bytes, exit classes, resource behavior, and the absence of host-dependent leakage. Diagnostic prose, source layout, internal modules, and proprietary decision methods are not compared.

LARRI-010-CONF-BEHAVIOR-001	An implementation MUST accept, refuse, classify, and normalize every applicable public protocol input exactly as required for its claimed scope.

1.2 Experimental status

A passing report permits an Experimental LARRI 0.1.0 Protocol Conformance statement within the synthetic laboratory and supervised evaluation boundary. It does not create a production approval, substantive authorization, or optional reproduction claim.

2. Claims and exact conformance scope

Property	Required evidence	Permitted statement
Artifact verification PASS or REFUSE	One protocol execution under a selected target and public inputs.	The selected assurance was or was not established for that artifact or subject.
Protocol Conformance PASS or FAIL	One implementation executes every applicable vector and gate for the exact scope.	The identified implementation conforms to LARRI 0.1.0 for the reported scope.
Clean-Room Reproduction	A separately engineered implementation reproduces every applicable canonical result without shared LARRI-specific verifier logic.	The stated behavior was reproduced under the reported scope.
Independent Interoperability	At least two organizationally independent implementations exchange applicable inputs and reproduce required outputs.	Independent implementations interoperate for the tested scope.
Reproducible Release	One implementation artifact is rebuilt under a declared recipe and its identity is reproduced.	The identified implementation release is reproducibly constructed under the stated recipe.

Property	Required evidence	Permitted statement
Release-envelope authentication	A registered signature verifies over exact release-manifest bytes.	The identified key authenticated the exact release-envelope bytes.

Only Protocol Conformance is required for an ordinary implementation claim. Every other property is optional and independent. Reproducible Release concerns an implementation artifact only. Reproducibility of a bundle carrier belongs only to a separately enabled carrier extension.

LARRI-010-CONF-CLAIM-001	A claim MUST use one registered claim label and MUST NOT imply Clean-Room Reproduction, Independent Interoperability, Reproducible Release, or release-envelope authentication without the additional evidence required for that property.
LARRI-010-CONF-CLAIM-002	Protocol Conformance MUST NOT require publication of proprietary source code, decision logic, private evidence, software architecture, models, prompts, thresholds, policies, or product workflow.

2.1 Closed scope tuple

Member	Scope rule
protocol_version	Exactly 0.1.0.
contracts_sha256	Identity of the exact semantic-contract manifest established under Volume II.
vector_set_sha256	Identity of the exact aggregate conformance-evidence index.
target_contexts	Non-empty duplicate-free contexts, each binding target, profiles, combination where required, extensions, semantic-record digests, and result_contract_id.
implementation_subject	Stable non-sensitive subject identifier, subject form, and exact version or deployment identifier where mutable.
platform_scope	Non-empty stable descriptors for every claimed execution environment or service deployment identifier.

SCHEMATIC - NOT MACHINE-VALID. The example is incomplete and shows member relationships only. It is not a fixture or expected output.

```
{
  "schema_id": "larri/conformance-scope",
  "schema_version": "0.1.0",
  "protocol_version": "0.1.0",
  "contracts_sha256": "<64 lowercase hex>",
  "vector_set_sha256": "<64 lowercase hex>",
  "target_contexts": [{
    "verification_target": "receipt",
    "profiles": [],
    "profile_combination": null,
    "extensions": [],
    "result_contract_id": "larri/verifier-result"
  }],
  "implementation_subject": {"...":"closed subject identity"},
  "platform_scope": ["declared-environment"]
}
```

LARRI-010-CONF-SCOPE-010	Every conformance claim MUST be limited to the exact scope tuple reported in its conformance evidence.
LARRI-010-CONF-SCOPE-011	Receipt and bundle contexts MUST use an empty profiles array. A profiled-bundle context MUST bind each selected profile identifier and exact semantic profile-record digest.
LARRI-010-CONF-SCOPE-012	A context with more than one selected profile MUST bind the exact applicable profile-combination identifier and semantic record digest. A context with zero or one selected profile MUST use null.

LARRI-010-CONF-SCOPE-013	A conformance scope MUST contain at least one target context, one applicable executable vector, one implementation subject, and one claimed platform descriptor. An empty scope cannot PASS.
LARRI-010-CONF-SCOPE-014	Each target context MUST bind the exact result_contract_id used for expected-byte comparison and every required extension identifier with its semantic contract digest.

3. Identity layers, traceability, and package model

Layer	Identity	Permitted dependency
Semantic contracts	contracts_sha256	Canonical semantic-contract manifest and its closed semantic files only.
Conformance evidence	vector_set_sha256	May depend on contracts_sha256 and applicable semantic-record digests.
Conformance report	Digest of canonical report bytes	May depend on exact scope, contracts_sha256, vector_set_sha256, detailed report, and gate report.
Release envelope	Digest of final canonical release manifest	May inventory all preceding identities, reports, gates, normative PDFs, metadata, and release files.

LARRI-010-CONF-IDENTITY-001	contracts_sha256 MUST retain exactly the semantic-contract meaning defined by Volume II and MUST NOT depend on vectors, fixtures, expected results, reports, gate results, publication metadata, or a release envelope.
LARRI-010-CONF-IDENTITY-002	vector_set_sha256 MUST be SHA-256 over one canonical closed aggregate conformance-evidence index that binds contracts_sha256, applicable semantic-record digests, vectors, fixtures, frozen expected outputs, result contracts, exit classes, and applicability.
LARRI-010-CONF-IDENTITY-003	Dependencies MUST flow only from semantic contracts to conformance evidence to reports to the release envelope. No earlier identity may include a later-layer digest.

3.1 Semantic-contract package

Component	Required content
schemas/	Every enabled protocol and supporting semantic schema.
registries/	One canonical authored source for every semantic registry class.
profiles/ and extensions/	Exact semantic records and their record digests; no vector-set or release identity.
requirements.json	Every normative definition, owning volume, source section, normalized text digest, and assertion class.
publication_gates.json	Objective gate definitions with no release-specific result digest.
semantic-contract manifest	Closed inventory whose canonical bytes are identified by contracts_sha256.

LARRI-010-CONF-PACKAGE-001	A conformance harness MUST use the exact semantic-contract package whose canonical manifest digest equals the reported contracts_sha256.
LARRI-010-CONF-PACKAGE-002	The semantic-contract package MUST use closed scope and MUST include every semantic schema, registry, requirement source, profile or extension record, and non-release-specific gate definition used for the claim.
LARRI-010-CONF-PACKAGE-003	No vector may establish trust by carrying the trusted key or trust object only inside the artifact under test.

3.2 Requirement identity and traceability

requirements.json distinguishes one requirement definition from any number of references. Every definition records its owning volume and source section. Vector records, coverage records, examples, reports, and gates may reference that identifier.

LARRI-010-CONF-TRACE-001	Every normative requirement definition across Volumes I through IV MUST appear exactly once in requirements.json under its exact ASCII identifier.
LARRI-010-CONF-TRACE-002	Every requirement reference MUST resolve to exactly one definition. A dangling or multiply resolved reference is a publication failure.
LARRI-010-CONF-TRACE-003	Every applicable MUST and MUST NOT MUST map to at least one executable schema constraint, registry assertion, vector, harness assertion, or objectively executable publication gate.
LARRI-010-CONF-TRACE-004	Coverage MUST include every active refusal, enabled stage branch, check-status transition, boundary, nullability branch, profile context, and supported identifier combination capable of changing public output.

3.3 Aggregate evidence index

Member	Rule
schema and protocol context	Closed 0.1.0 constants.
contracts_sha256	Exact semantic-contract identity.
semantic_record_digests	Every profile, combination, extension, and result-contract digest needed by indexed vectors.
vectors	Path, byte length, SHA-256, vector_id, target context, result contract, expected exit class, and applicability.
fixtures and expected outputs	Normalized paths, byte lengths, SHA-256 digests, and media roles.
applicability	Exact target, profile, extension, result-contract, platform, and configuration conditions.

LARRI-010-CONF-PACKAGE-010	The aggregate evidence index MUST be canonical and closed. A report MUST NOT combine results from different vector_set_sha256 values.

4. Vector contract, evidence index, and coverage

A vector is an immutable public test case that binds one exact protocol context, every input fixture, one expected exit class, and exact expected public bytes when the exit class emits a protocol object.

SCHEMATIC - NOT MACHINE-VALID. The example is incomplete and shows member relationships only. It is not a fixture or expected output.

```
{
  "schema_id": "larri/conformance-vector",
  "schema_version": "0.1.0",
  "vector_id": "receipt-id-mismatch-001",
  "requirement_ids": ["LARRI-010-ART-RID-001"],
  "case_class": "negative-mutation",
  "verification_target": "receipt",
  "profiles": [],
  "profile_combination": null,
  "required_extensions": [],
  "result_contract_id": "larri/verifier-result",
  "artifact_inputs": [{"path":"receipt.json","sha256":"<64 lowercase hex>"}],
  "external_trust_fixture": {"path":"trust/key.json","sha256":"<64 lowercase hex>"},
  "configuration_fixture": {"path":"config/baseline.json","sha256":"<64 lowercase hex>"},
  "expected_output": {"path":"expected/result.json","sha256":"<64 lowercase hex>"},
  "expected_exit_class": "verification_refuse",
  "fault": "Replace receipt_id and regenerate the outer attestation.",
  "applicability": {"platforms":["all-claimed"]}
}
```

LARRI
VERIFICATION PROTOCOL

Deterministic ordering and duplicate rejection

Canonical JSON preserves array order; it does not sort arrays. Every set-like Volume IV array therefore rejects duplicates before canonicalization and uses the exact order below. These rules apply to the conformance scope, aggregate evidence index, vector records, detailed execution records, gate report, conformance report, and release manifest. They are part of the canonical-and-closed requirement in LARRI-010-CONF-PACKAGE-010.

Set-like array	Duplicate comparison and required order
requirement_ids	Reject duplicate exact ASCII identifiers; sort by exact ASCII identifier.
profiles and extensions	Reject duplicate identifiers; sort by applicable registry rank, then exact ASCII identifier.
target_contexts	Reject duplicate canonical contexts; sort by verification-target registry rank, then canonical context bytes.
platform_scope	Reject duplicate exact descriptors; sort by exact ASCII descriptor.
vectors	Reject duplicate vector_id values; sort by exact ASCII vector_id.
semantic_record_digests	Reject duplicate semantic identities; sort by semantic class, applicable registry rank, then exact ASCII identifier.
artifact_inputs, fixtures, expected_outputs, and release payload files	Reject duplicate normalized paths; sort by normalized_path.
detailed executions	Reject duplicate vector executions for one evidence position; emit in aggregate evidence-index vector order.
gate results	Reject duplicate gate identifiers; emit exactly G01 through G13.

Exactness. A duplicate is rejected rather than silently deduplicated. Input order never changes a canonical hash, report byte sequence, or conformance meaning.

Scope of G14. G14 is not a member of the canonical G01-G13 gate-results array because it executes only after the reports and release envelope have been assembled.

LARRI-010-CONF-VECTOR-001	Every vector MUST bind requirement references, case class, target context, profiles and exact combination where applicable, required extensions, result contract, every artifact-input digest, independently supplied trust fixture, exact verification configuration, expected output when applicable, expected exit class, fault description, and applicability.
LARRI-010-CONF-VECTOR-002	Vector and fixture paths MUST satisfy the Volume II normalized-path contract and MUST resolve within the closed evidence root without links, traversal, aliases, or duplicate normalized paths.
LARRI-010-CONF-VECTOR-003	A negative-mutation vector MUST describe its intended defect and SHOULD change only the bytes required to exercise that branch unless expressly classified as multi-fault.
LARRI-010-CONF-VECTOR-004	A vector, every fixture, and every expected output MUST be immutable under the reported vector_set_sha256.
LARRI-010-CONF-VECTOR-005	An expected-output file MUST contain the exact frozen canonical bytes required by its result contract or invocation-refusal contract. A digest or prose summary alone is insufficient.

4.1 Closed case classes

Case class	Required purpose
positive	Valid minimal or representative input and exact verification_pass output.
negative-mutation	One defined defect and deterministic verification_refuse output.
boundary	Behavior at and immediately beyond an encoding, count, size, threshold, graph, or lexical boundary.
multi-fault	Primary-refusal precedence, safe sibling collection, and downstream SKIP.
unsupported-feature	Unknown, disabled, inapplicable, unsupported, or unsupported-schema-version behavior under 0.1.0 recognition rules.
resource	Deterministic verification_refuse or permitted execution_limit without unsafe processing.
trust	External-trust identity, permission, fingerprint, key-use, and phase-boundary behavior.
normalized-output	Exact identities, nullability, ordering, statuses, limitations, and canonical bytes.
claim-safety	Exact target, profile, result contract, claim, limitation, and no-overclaim behavior.
platform	Host independence across environments actually claimed.
security	Canonicalization, signature, downgrade, path, and parser adversarial behavior.

4.2 Branch-based coverage

Vector counts are not a measure of completeness. Coverage is complete only when every applicable public branch is exercised directly or assigned to an objective gate that can mechanically determine compliance. Reference implementation output is not an oracle; expected outputs are independently checked and frozen before ordinary execution.

LARRI-010-CONF-COVERAGE-001	Every enabled base target for which at least one enabled semantic context capable of PASS exists MUST have applicable positive, negative, boundary, multi-fault, trust, resource, normalized-output, security, and platform coverage. While no optional profile is enabled, profiled-bundle coverage MUST consist of PROFILE_SELECTION_REQUIRED and every applicable Stage 3 contract-resolution branch and MUST NOT require or report a positive profiled-bundle PASS.
LARRI-010-CONF-COVERAGE-002	Every enabled profile, extension, profile combination, and result contract MUST have direct coverage for required evidence, trust inputs, claims, limitations, refusals, status transitions, downgrade behavior, and interaction with prerequisite base checks.
LARRI-010-CONF-COVERAGE-003	A broad regression result or aggregate pass rate MUST NOT substitute for a missing branch, refusal, status, boundary, nullability condition, applicability condition, or gate assertion.

LARRI-010-CONF-COVERAGE-004	Coverage MUST exercise every specified null and non-null branch of artifact_sha256, trust_input_sha256, and configuration_sha256 without fabricated, zero, placeholder, or empty-input identities.

5. Harness execution and exit classes

A conformance harness may use any interface supported by the implementation. LARRI does not mandate a command line, process model, container, repository layout, programming language, API, or service architecture. The harness controls all protocol-visible inputs and retains enough evidence to reproduce each execution.

- Supply semantic contracts, artifacts or subjects, external trust fixtures, profile selection, target, and verification configuration through distinct harness-controlled channels.
- Use the exact contracts_sha256 and vector_set_sha256 reported by the claim.
- Disable network access for base conformance execution and supply every required input explicitly.
- Treat contracts, vectors, fixtures, configurations, and expected outputs as read-only and detect modification.
- Use an isolated mutable working scope for each vector and reset state between executions.
- Run every applicable vector at least twice and run the full suite in at least one independently shuffled order.
- Vary non-protocol environment details such as temporary paths, working directory, locale, time zone, process identifier, and concurrency schedule.
- Retain protocol outputs, exit classes, implementation-subject identity, platform descriptor, and non-sensitive diagnostics separately.

LARRI-010-CONF-HARNESS-001	Repeated executions with identical protocol inputs MUST produce byte-identical canonical public output bytes.
LARRI-010-CONF-HARNESS-002	An artifact-caused or target-scoped refusal MUST be represented as protocol data and MUST NOT be classified as a harness crash, uncontrolled exception, or successful early termination.
LARRI-010-CONF-HARNESS-003	The harness MUST fail a vector if the implementation modifies a read-only input, observes vector-order dependence, uses undeclared network input, or leaks host-dependent data into public output bytes.
LARRI-010-CONF-HARNESS-004	The harness MUST record the exact implementation subject, platform descriptor, contracts_sha256, vector_set_sha256, input fixture digests, trust-fixture digest, and configuration-fixture digest for every execution.

5.1 Exit classes

Exit class	Meaning and treatment
verification_pass	PASS under the selected result contract; bytes exactly match. Successful only when expected.
verification_refuse	REFUSE under the selected result contract; bytes exactly match. Successful only when expected.
invocation_refuse	Exact closed larri/invocation-refusal bytes. Successful only for two permitted invocation codes.
execution_limit	Safe termination before a protocol object could be emitted. Successful only where expressly permitted.
harness_error	Harness or implementation could not execute as specified. Always a failure.
comparison_failure	Exit class or expected bytes differ. Always a failure.
publication_gate_failure	One or more required gates failed. Always prevents a passing report.

LARRI-010-CONF-HARNESS-010	Each vector MUST name exactly one expected exit class. A harness MUST NOT treat materially different outputs or exit classes as equivalent success for the same vector.

LARRI-010-CONF-EXIT-001	invocation_refuse MUST apply only to SEMANTIC_CONTRACTS_NOT_ESTABLISHED and UNSUPPORTED_VERIFICATION_TARGET and MUST compare the exact closed larri/invocation-refusal bytes.
LARRI-010-CONF-EXIT-002	Missing, malformed, schema-invalid, or non-stricter verification configuration after an enabled target is established MUST remain target-scoped verification_refuse behavior. Harness errors, comparison failures, and gate failures MUST NOT be accepted as alternate expected outcomes.

6. Expected-byte comparison and stage behavior

The harness compares the exact bytes selected by the vector result contract. For target-scoped verification this is normally canonical larri/verifier-result. For invocation refusal it is canonical larri/invocation-refusal. Diagnostic logs, process output, free-form messages, and exception details are never substitutes.

LARRI-010-CONF-COMPARE-001	A difference in target context, profiles, artifact or subject identity, contracts identity, trust-input identity, trust fingerprints, limits context, outcome, primary refusal, refusal order, check identity, check order, check status, detail code, limitation, result contract, exit class, or canonical output bytes MUST fail the vector.
LARRI-010-CONF-COMPARE-002	Absolute paths, hostnames, process identifiers, local timestamps, execution duration, temporary directories, stack traces, stdout, stderr, and free-form exception text MUST be absent from normalized comparison.
LARRI-010-CONF-COMPARE-003	Expected-output files MUST contain exact canonical bytes. Normal conformance execution MUST NOT regenerate expected outputs from the implementation under test.

6.1 Eleven-stage dependency order

Stage	Check domain	Required precedence
1	Input and subject safety	Establish safe handling and applicable limits before unsafe processing.
2	Strict decoding and schema recognition	Apply strict decoding and recognize the intended top-level schema.
3	Contract and feature resolution	Resolve protocol features, selected profiles, combinations, required extensions, artifact types, checks, trust contracts, refusal codes, and result contract.
4	External trust	Validate the complete external-trust object and key eligibility for a direct receipt or manifest control.
5	Manifest canonical form and identity	Validate canonical form, reject manifest-declared duplicate normalized paths, and recompute manifest identity and digest.
6	Detached manifest attestation	Verify the manifest attestation after Stage 5 succeeds.
7	Closed bundle scope and file integrity	Validate observed paths, closed scope, artifact_type, content role, exact length, and digest.
8	Receipt identities and signed bindings	Safely decode contained receipts and recompute identity and signed duplicate relationships.
9	Receipt attestations	Resolve contained-receipt trust entries only after safe file integrity and verify required receipt signatures.
10	Commitments and selected profiles	Evaluate only conditions whose contracts resolved successfully at Stage 3.
11	Normalized result	Select deterministic primary and sibling refusals and emit exact result-contract bytes.

LARRI-010-CONF-STAGE-001	Vectors MUST prove the exact eleven-stage order, including Stage 3 selection and required-extension failures, Stage 5 manifest identity before Stage 6 signature acceptance, Stage 6 before Stage 7 file integrity, and Stage 10 evaluation only after successful resolution.

LARRI-010-CONF-STAGE-002	No unverified bundle file may be opened to establish Stage 4 trust. Contained-receipt key lookup and actual key use MUST be exercised at Stage 9 after Stage 7 file integrity and Stage 8 receipt processing.

6.2 Multi-fault behavior

Relationship	Expected behavior
Configuration or contract resolution before artifact processing	A blocking configuration or Stage 3 resolution refusal prevents dependent interpretation.
Manifest identity before manifest signature	Canonical-form, duplicate-path, manifest_id, or digest mismatch is primary before signature acceptance.
Manifest signature before file integrity	A failed manifest signature blocks dependent scope and contained-artifact assurances.
File integrity before contained trust use	A failed path, length, digest, role, or artifact_type check blocks dependent receipt attestation.
Receipt identity before receipt attestation	Identity or signed-binding mismatch is primary when signature verification depends on that content.
Resolved profile prerequisites before Stage 10	A failed base prerequisite produces SKIP for dependent profile checks.

LARRI-010-CONF-MULTI-001	Every verification stage, refusal rank, safe sibling relationship, and dependency-blocked check path MUST have direct multi-fault coverage.
LARRI-010-CONF-MULTI-002	A verifier MUST NOT execute an unsafe or dependency-blocked downstream check merely to increase the refusal list.

6.3 Receipt-identity sole-fault vector

Required construction. Replace receipt_id with a non-derived value, then regenerate the outer attestation over the resulting final signed content. The only remaining defect is the receipt identity mismatch. The expected refusal is RECEIPT_IDENTITY_REFUSED.

LARRI-010-CONF-RID-001	The receipt-identity mismatch vector MUST preserve a valid outer attestation over the mutated signed content so that RECEIPT_IDENTITY_REFUSED is the sole intended defect.

6.4 Differential comparison

Clean-Room Reproduction and Independent Interoperability compare canonical outputs for identical protocol inputs. Agreement only at the high-level PASS or REFUSE outcome is insufficient.

LARRI-010-CONF-COMPARE-010	Implementations participating in a reproduction or interoperability claim MUST produce byte-identical canonical outputs for every applicable vector and exchanged input in the reported scope.

7. Required base conformance suites

7.1 Result identity and nullability

Field and context	Required coverage
contracts_sha256	Always non-null in target-scoped results; no target result exists before semantic contracts are established.
artifact_sha256 - receipt	Complete byte stream, incomplete stream, malformed JSON after complete receipt, schema refusal, limit refusal, and later signature refusal.
artifact_sha256 - prepared directory	Missing or malformed manifest, noncomputable canonical form, established logical manifest subject followed by path, file, receipt, or profile refusal.
trust_input_sha256	Missing, malformed, noncanonicalizable, canonicalizable but schema-invalid, duplicate labels, fingerprint mismatch, forbidden target, unused entries, and later refusal.

Field and context	Required coverage
configuration_sha256	Exact baseline, valid but unused stricter configuration, missing, malformed, schema-invalid, non-stricter, and later refusal after valid nonbaseline selection.
trust_fingerprints	Only fingerprints whose keys actually satisfied required signatures.

LARRI-010-CONF-NULL-001	Identity and nullability vectors MUST compare every Volume II branch exactly and MUST NOT substitute zero digests, empty-input digests, placeholders, synthetic identities, or invented directory hashes.

7.2 Prepared-directory bundles and artifact classification

Base bundle vectors use the prepared-directory subject defined by Volume II. They cover normalized relative paths, absolute and traversal rejection, case and NFC collisions, prohibited links and non-regular entries, manifest-declared duplicate paths, observed filesystem collisions, closed scope, missing and extra files, length and digest mismatch, unsupported carrier input, and deterministic logical-subject identity.

Contained receipts are discovered only through artifact_type == "larri/decision-receipt". Path, filename, suffix, prefix, media type, and content sniffing have no classification effect. Protocol-owned unknown types refuse. Application-namespaced opaque types and null types receive only generic integrity treatment where permitted.

LARRI-010-CONF-SAFETY-001	A resource-exhaustion or unsafe-subject vector MUST produce the exact declared verification_refuse or permitted execution_limit outcome without uncontrolled allocation, code execution, root escape, unsafe file access, or partial PASS.
LARRI-010-CONF-SAFETY-002	A parser, path, canonicalization, trust, signature, artifact-classification, or check-bypass defect that permits unsafe acceptance MUST fail the applicable gate and prevent a Protocol Conformance PASS.
LARRI-010-CONF-SAFETY-003	Vectors, fixtures, expected outputs, reports, gate results, and release files MUST NOT contain private keys, signing seeds, live credentials, raw sensitive matches, prohibited host details, or unintended proprietary information.

7.3 Exactly six baseline resource limits

Limit	Required vectors
protocol_json_max_bytes	Exactly at 5,242,880 and immediately greater.
json_max_depth	Exactly at 64 and immediately greater.
json_max_object_members	Exactly at 200,000 and immediately greater.
json_max_array_length	Exactly at 200,000 and immediately greater.
bundle_max_entries	Exactly at 10,000 and immediately greater.
bundle_max_total_bytes	Exactly at 524,288,000 and immediately greater.

LARRI-010-CONF-RESOURCE-001	A baseline-limit vector MUST use the exact enabled larri/core-baseline-limits-v1 record and MUST NOT substitute an implementation default.
LARRI-010-CONF-RESOURCE-002	A valid nonbaseline configuration vector MUST bind the exact configuration bytes and compare the required non-null configuration_sha256 whether or not its stricter threshold changes the outcome.
LARRI-010-CONF-RESOURCE-003	Base conformance MUST provide at-limit and immediately-over coverage for exactly the six published baseline limits and MUST NOT require archive extraction, decompression, ratio, or expansion behavior.

7.4 External trust

Trust coverage includes missing trust, malformed trust JSON, canonicalizable but schema-invalid trust, duplicate key labels, malformed key bytes, fingerprint mismatch, target-forbidden trust, unused entries, direct receipt or manifest key failure at Stage 4, contained-receipt trust entry missing or unusable at Stage 9, trust_input_sha256 retention after later refusal, and trust_fingerprints containing only keys that actually satisfied required signatures.

Trust fixtures remain independent of the artifact under test, as required by LARRI-010-CONF-PACKAGE-003.

7.5 Canonicalization and cryptography

Area	Required coverage
Strict JSON and UTF-8	BOM rejection, malformed encodings, surrogate rejection, duplicate decoded names, control handling, and full-input consumption.
Unicode and ordering	Unicode 16.0.0 NFC, post-NFC collisions, scalar member ordering, escaping, and trailing-newline rejection.
Numbers	Signed 64-bit boundaries, fractions, exponents, negative zero, leading zeros, and non-integer rejection.
Base64	Exact standard alphabet, padding, decoded lengths, zero unused pad bits, whitespace rejection, and full-string matching.
Signature and policy	Strict key, signature, point, and scalar rejection; wrong key, wrong message, missing or extra attestation; policy counting and deterministic ordering.
Downgrade	Unknown, disabled, inapplicable, or unsupported required suites and policies refuse without fallback.

The baseline suite is tested exactly as published: larri/ed25519-sha256-v1 applies Ed25519 to the raw 32-byte SHA-256 digest produced by the registered message construction. It is not direct-message Ed25519 and is not RFC 8032 Ed25519ph.

7.6 Profiles and extensions

The base semantic registries contain no enabled optional profile, profile combination, or optional extension. No positive profiled-bundle PASS vector is manufactured. The profiled-bundle target is exercised through PROFILE_SELECTION_REQUIRED and every applicable Stage 3 contract-resolution branch.

Claim-safety vectors compare the exact selected result contract, retain every mandatory limitation unless passing public evidence directly resolves it, bind the identity and actual-use facts of every additional trust input, and prove that privacy screening is never represented as universal absence of sensitive information.

LARRI-010-CONF-PROFILE-001	A profile or extension is conforming only when every prerequisite base target and semantic contract also conforms for the claimed platform scope.
LARRI-010-CONF-PROFILE-002	A profile context MUST bind every selected profile-record digest, applicable combination-record digest, required extension semantic digest, exact result_contract_id, and the aggregate vector_set_sha256. No separate vector-set digest may be attached to a semantic profile or extension record.
LARRI-010-CONF-PROFILE-003	Opaque preservation of unknown or unselected extension bytes MUST NOT be counted as execution of extension behavior, satisfaction of a profile, semantic assurance, or extension conformance.
LARRI-010-CONF-PROFILE-004	When no enabled optional profile exists, conformance MUST NOT require or report a positive profiled-bundle PASS. Positive profile coverage becomes applicable only when a real enabled companion profile and its closed result contract exist.

Before any optional time, witness, transparency, privacy, lineage, alternative carrier, multisignature, post-quantum, hybrid, or domain scheme can be enabled, it requires complete positive, negative, downgrade, boundary, trust, and multi-fault evidence. No vectors are required for a disabled or reserved scheme except recognition and non-fallback behavior where applicable.

7.7 Platform scope

LARRI-010-CONF-PLATFORM-001	A path, encoding, key, signature, result, or artifact accepted on one claimed platform and refused on another is a conformance failure unless the contracts disable that case for the entire reported scope.
LARRI-010-CONF-PLATFORM-002	Locale, time zone, filesystem case behavior, temporary path, process identity, concurrency scheduling, and host metadata MUST NOT alter gating public output bytes.

8. Implementation subject and optional claims

Protocol Conformance identifies one implementation subject through a stable non-sensitive identifier. A subject may be an executable, service deployment, appliance, library, or other evaluable implementation. Source disclosure and reproducible construction are not required for ordinary Protocol Conformance.

Evidence	Protocol Conformance	Reproducible Release
Subject identifier	Required.	Required.
Subject version or deployment identifier	Required for mutable or updateable subjects.	Required.
Subject artifact digest	Required when content-addressable; otherwise optional when stable deployment identity is available.	Required for reproduced implementation artifact.
Source identifier	Optional.	Required when the recipe depends on identified source.
Dependency or toolchain record	Optional.	Required to the extent needed to reproduce the implementation artifact.
Build recipe	Optional.	Required.
Public source code	Not required.	Not inherently required when an independently executable reproduction method is supplied.

LARRI-010-CONF-RELEASE-001	Implementation release metadata MUST NOT affect ordinary receipt, bundle, or profiled-bundle PASS unless a selected profile expressly requires and verifies that metadata.
LARRI-010-CONF-RELEASE-002	A mutable service claim MUST identify the exact deployment identifier or equivalent stable subject identity tested; a persistent product name alone is insufficient.

8.1 Clean-Room Reproduction

A clean-room claimant independently implements LARRI-specific parsing, canonicalization, identity, policy, refusal selection, and output normalization. General-purpose cryptographic libraries may be shared, but published vector-generation logic must not be used as verifier logic.

LARRI-010-CONF-INDEPENDENCE-001	A Clean-Room Reproduction claim MUST document the independent implementation boundary and MUST disclose shared LARRI-specific code, generated parser, canonicalizer, verifier component, or test oracle that would invalidate the claim.

8.2 Independent Interoperability

Organizational independence requires separate maintainers and release authority. Exchange tests include valid inputs and adversarial cases and bind exact artifacts, trust, configurations, outputs, and differential results.

LARRI-010-CONF-INDEPENDENCE-010	An Independent Interoperability claim MUST identify at least two organizationally independent implementation subjects and MUST bind the exact exchanged inputs, outputs, and differential-test results.
LARRI-010-CONF-INDEPENDENCE-011	Independent Interoperability MUST NOT be claimed from two wrappers around the same LARRI-specific verifier logic.

LARRI
VERIFICATION PROTOCOL

9. Publication gates

G01 through G13 execute over the established semantic contracts, frozen conformance evidence, vector executions, and report inputs. Their results form one canonical gate report ordered exactly G01 through G13. The conformance report binds that gate report. After the detailed report, G01-G13 gate report, and conformance report are complete, the release envelope is assembled. G14 then executes last over the exact final package. G14 failure blocks publication, and the G14 outcome is not embedded in any report or gate artifact that G14 audits.

Gate	Pass condition
G01 - Semantic and evidence closure	Semantic contracts, evidence, report inputs, and one-way identities are closed and coherent before report generation.
G02 - Schema, fixture, and example validation	Every schema and every fixture validates under its intended schema. Every specification example designated MACHINE-VALID validates; every incomplete or placeholder-bearing example is labeled SCHEMATIC - NOT MACHINE-VALID.
G03 - Registry parity	Every generated registry view and prose table matches one canonical authored source.
G04 - Requirement identity	Every definition uses one globally unique ASCII identifier, every text digest matches, and every reference resolves exactly once.
G05 - Coverage completeness	Every applicable requirement, branch, refusal, status, boundary, nullability, applicability condition, and ordering rule has direct evidence or an objective gate.
G06 - Identifier cross-reference	Every closed-registry value resolves exactly once with applicable status.
G07 - Target coherence	receipt, bundle, and profiled-bundle are enabled; positive PASS coverage is required only where an enabled semantic context capable of PASS exists.
G08 - Baseline-limit coherence	Exactly six Volume II baseline limits resolve to larri/core-baseline-limits-v1 and every boundary vector uses the same record.
G09 - Evidence validation	Every index entry, vector, fixture digest, frozen expected bytes, exit class, result contract, applicability rule, array order, and vector_set_sha256 validates and executes as declared.
G10 - Cryptographic and multi-fault security	Canonicalization, signature, trust, downgrade, message construction, refusal precedence, and blocked-check vectors pass.
G11 - Prepared-directory safety	Path, closed scope, links, duplicate handling, artifact_type classification, exact file integrity, and resource vectors pass with no base archive requirement.
G12 - Determinism and platform parity	Repeated and shuffled runs, host variation, and every environment actually claimed produce exact required bytes.
G13 - Secrets and privacy scan	No prohibited key material, credential, raw sensitive match, host detail, or unintended proprietary information appears.
G14 - Deterministic final-package audit	Executed last over the exact final release controls and payload. It verifies closure, identities, reports, signatures where present, and payload bytes. Its result is external to the audited package and failure blocks publication.

LARRI-010-CONF-PUB-001	G01 through G13 MUST produce the canonical gate report and MUST PASS before a conformance report may PASS. The conformance report MUST bind that G01-G13 gate report. After the detailed, gate, and conformance reports and the complete release envelope are assembled, G14 MUST execute last on the exact final package. G14 output MUST remain outside every artifact G14 audits, and a G14 failure MUST block publication.
LARRI-010-CONF-PUB-002	Requirement-identity, identifier-cross-reference, and one-way-identity checks are standing gates and MUST run on every candidate package.
LARRI-010-CONF-PUB-003	G02 MUST validate every published schema and every fixture. The MACHINE-VALID designation governs which specification examples also undergo schema and registry validation; every incomplete or placeholder-bearing specification example MUST carry the exact SCHEMATIC - NOT MACHINE-VALID designation.
LARRI-010-CONF-PUB-004	The evidence set MUST include a generic unsupported-schema-version vector derived solely from the 0.1.0 version-recognition rules and MUST NOT use fallback aliases or heuristic parsing.
LARRI-010-CONF-PUB-005	G14 MUST be executable entirely offline and internally and MUST execute only after the release manifest and any detached release signature are final. Its outcome MUST NOT be written into a conformance report, gate report, release manifest, detached signature, or other package artifact that G14 audits. Optional independent review MAY be performed but MUST NOT be a prerequisite.

LARRI
VERIFICATION PROTOCOL

10. Conformance report

SCHEMATIC - NOT MACHINE-VALID. The example is incomplete and shows member relationships only. It is not a fixture or expected output.

```
{
  "schema_id": "larri/conformance-report",
  "schema_version": "0.1.0",
  "protocol_version": "0.1.0",
  "scope": {"...":"closed conformance scope"},
  "applicable_vector_count": 0,
  "verification_pass_count": 0,
  "verification_refuse_count": 0,
  "invocation_refuse_count": 0,
  "execution_limit_count": 0,
  "comparison_failure_count": 0,
  "harness_error_count": 0,
  "publication_gate_failure_count": 0,
  "detailed_report_sha256": "<64 lowercase hex>",
  "gate_report_sha256": "<64 lowercase hex>",
  "outcome": "PASS"
}
```

The closed scope is the sole authoritative location for implementation_subject, contracts_sha256, and vector_set_sha256. A report schema that retains any duplicate top-level copy of one of those values must require exact equality with scope. The detailed report is a deterministic closed execution record whose executions appear in aggregate evidence-index vector order. The gate report is the deterministic closed G01-G13 record, with results ordered exactly G01 through G13. Neither report contains a G14 outcome.

LARRI-010-CONF-REPORT-001	A conformance report MUST bind one exact closed scope, every required count, detailed_report_sha256, the canonical G01-G13 gate_report_sha256, and outcome PASS or FAIL. implementation_subject, contracts_sha256, and vector_set_sha256 MUST be authoritative in scope; any retained duplicate top-level member MUST exactly equal scope.
LARRI-010-CONF-REPORT-002	PASS requires a non-empty applicable scope, every applicable vector accounted for exactly once, exact expected exit classes and bytes, zero comparison failures, zero harness errors, zero G01-G13 publication-gate failures, and no scope, identity, ordering, or count inconsistency.
LARRI-010-CONF-REPORT-003	A report MUST be FAIL when any applicable vector is missing, duplicated, skipped without contract authorization, mismatched, or executed under a different scope identity, target context, result contract, platform, trust fixture, or configuration fixture.
LARRI-010-CONF-REPORT-004	A report signature or release signature authenticates exact bytes only and MUST NOT change FAIL to PASS, repair a failed gate, enlarge scope, authorize relying-party action, establish artifact truth, or imply regulatory compliance.
LARRI-010-CONF-REPORT-005	Human-readable summaries MUST be derived from the canonical report and MUST NOT omit a failure, limitation, target context, or identity necessary to interpret the claim.

LARRI
VERIFICATION PROTOCOL

10.1 Report derivation and count equation

The detailed execution array is emitted in aggregate evidence-index vector order. Every applicable vector occupies exactly one evidence position and contributes to exactly one execution-class count. The canonical G01-G13 gate-results array is separate, duplicate-free, and ordered G01 through G13.

Equation component	Treatment
applicable_vector_count	Total number of applicable evidence-index vector positions.
Successful expected classes	verification_pass_count + verification_refuse_count + invocation_refuse_count + execution_limit_count
Vector-execution failures	comparison_failure_count + harness_error_count
Required equation	applicable_vector_count equals the sum of the six counts above.
Gate failures	publication_gate_failure_count is separate and never enters applicable_vector_count.

LARRI-010-CONF-REPORT-010	Each applicable vector MUST contribute to exactly one of verification_pass_count, verification_refuse_count, invocation_refuse_count, execution_limit_count, comparison_failure_count, or harness_error_count. Their sum MUST equal applicable_vector_count. Gate failures are not vector executions and MUST NOT enter applicable_vector_count.
LARRI-010-CONF-REPORT-011	A report MUST NOT combine executions from different implementation subjects, platform descriptors, semantic-contract identities, evidence identities, or scope tuples.
LARRI-010-CONF-REPORT-012	Successful-class counts MUST include only vectors whose expected class and exact expected bytes, where applicable, matched. comparison_failure_count and harness_error_count MUST be explicit vector-execution counts. publication_gate_failure_count MUST count only failed G01-G13 gate results and MUST NOT be treated as a vector-execution count.

LARRI
VERIFICATION PROTOCOL

11. Release envelope and authentication

The fixed release control files are release.larri.json and, when supplied, release.larri.sig.json. Neither control file appears in the release manifest files array. The release envelope is assembled only after the detailed report, canonical G01-G13 gate report, and conformance report are complete. It may inventory semantic contracts, conformance evidence, reports, normative PDFs, publication metadata, and other declared release payload files.

SCHEMATIC - NOT MACHINE-VALID. The example is incomplete and shows member relationships only. It is not a fixture or expected output.

```
{
  "schema_id": "larri/release-manifest",
  "schema_version": "0.1.0",
  "protocol_version": "0.1.0",
  "release_id": "<64 lowercase hex>",
  "contracts_sha256": "<64 lowercase hex>",
  "vector_set_sha256": "<64 lowercase hex>",
  "conformance_report_sha256": "<64 lowercase hex>",
  "gate_report_sha256": "<64 lowercase hex>",
  "files": [{
    "path": "<normalized payload path>",
    "role": "<registered release role>",
    "byte_length": 0,
    "sha256": "<64 lowercase hex>"
  }]
}
```

Control-file rule. files inventories every release payload other than release.larri.json and release.larri.sig.json exactly once in normalized_path order. Duplicate normalized paths refuse. release_id remains non-recursive: canonicalize the closed release manifest with release_id omitted, compute SHA-256, encode lowercase hexadecimal, insert the value, and canonicalize the final manifest.

Optional detached signature. release.larri.sig.json, when present, authenticates the exact final canonical release.larri.json bytes under one registered public signature contract. It does not authenticate an inferred archive or directory serialization.

LARRI-010-CONF-ENVELOPE-001	The fixed top-level release controls MUST be release.larri.json and optional release.larri.sig.json and MUST be excluded from files. files MUST reject duplicates and inventory every other payload exactly once by normalized path, byte length, role, and SHA-256 in normalized-path order, while binding contracts_sha256, vector_set_sha256, the conformance-report digest, and the G01-G13 gate-report digest.
LARRI-010-CONF-ENVELOPE-002	Release-envelope identities and signatures MUST NOT appear inside contracts_sha256 or vector_set_sha256. Authentication MUST NOT repair failed evidence, change report outcome, enlarge a claim, or embed a G14 result. G14 MUST execute only after the optional detached release signature has been assembled.

The release envelope may be unsigned. When authenticated, the permitted statement is limited to the exact key and exact canonical release-manifest bytes verified. Key identity, organizational authority, and legal effect remain outside this volume unless separately established.

LARRI
VERIFICATION PROTOCOL

12. Failure, withdrawal, correction, and reissue

A Protocol Conformance claim remains valid only while its exact report, scope, identities, and package evidence remain available and no known defect invalidates the claim. Handling is objective and may be performed entirely within one organization.

LARRI-010-CONF-WITHDRAW-001	A claim owner MUST withdraw a Protocol Conformance PASS when an applicable vector, comparison, harness, traceability, security, privacy, identity, G01-G13 gate, or G14 final-package defect is discovered that would block the bound claim or publication.
LARRI-010-CONF-CORRECT-001	Correction MUST preserve authenticated report and package bytes, correct the authoritative source material, regenerate the semantic and evidence layers as applicable, rerun the complete vector suite and G01-G13, generate new detailed, gate, and conformance reports, assemble a new release envelope and optional signature, and execute G14 last. G14 failure MUST block reissue.
LARRI-010-CONF-REISSUE-001	A corrected report or release package MUST be issued as a new immutable byte sequence with a new digest. Authenticated bytes MUST NOT be edited in place, and a partial rerun MUST NOT restore PASS.

Operational boundary. Withdrawal, correction, and reissue are internal evidence-control actions. They do not require an external reviewer, regulator, service, or organization.

A. Minimum coverage catalogue

Coverage family	Minimum direct cases
Invocation	Semantic contracts not established; unsupported selected target; exact invocation-refusal bytes and codes.
Configuration	Baseline; valid unused stricter; missing; malformed; schema-invalid; non-stricter; later refusal with valid nonbaseline digest retained.
Receipt intake	Complete bytes with malformed JSON; incomplete bytes; schema defect; exact artifact identity branches.
Receipt identity	Positive identity; sole-fault mismatch with regenerated attestation; non-recursive construction.
Manifest identity	Canonical positive; malformed; duplicate declared normalized paths; manifest_id mismatch; manifest digest mismatch.
Manifest attestation	Missing, extra, key mismatch, wrong message, invalid signature, target-forbidden key.
Prepared directory	Absolute/traversal paths, case/NFC collisions, links, non-regular entries, observed collisions, missing/extra files, length/digest mismatch, unsupported carrier.
artifact_type	All Volume II top-level types, arbitrary safe receipt path, misleading path, missing type, null opaque type, application namespace, unknown protocol type, role mismatch.
Trust	Missing, malformed, schema-invalid, duplicate labels, fingerprint mismatch, target permission, unused entry, Stage 4 direct/manifest key, Stage 9 contained key.
Canonicalization	UTF-8/BOM, surrogates, duplicate names, NFC, ordering, escaping, number boundaries, negative zero, full-string lexical checks, Base64.
Signature policy	Strict key/point/scalar, wrong key/message, missing/extra attestation, counting, ordering, downgrade.
Limits	At and immediately over each of six base limits; every selected nonbaseline configuration identity.
Profiles	No-profile Stage 3 refusal; unknown/disabled/inapplicable profile; missing/ambiguous combination; required-extension resolution; Stage 10 SKIP after base failure.
Output	Every refusal order, check status, limitation, identity nullability branch, exact result contract, and host-leakage prohibition.
Platform	Repeated, shuffled, locale, time-zone, path, process, filesystem-case, and concurrency variation for claimed environments only.

Applicability. This catalogue is a minimum branch catalogue, not a fixed vector count. Disabled or reserved optional schemes require recognition and non-fallback evidence where applicable, but do not require manufactured positive vectors.

B. Permitted claim templates

B.1 Protocol Conformance

Permitted statement. The identified implementation conforms to LARRI 0.1.0 for the exact scope, contracts_sha256, vector_set_sha256, implementation subject, and platform descriptors identified by the passing conformance report.

B.2 Clean-Room Reproduction

Permitted statement. The stated LARRI behavior was reproduced by a separately engineered implementation for the exact reported scope.

B.3 Independent Interoperability

Permitted statement. The identified independent implementations interoperated for the exact exchanged inputs and tested scope.

B.4 Reproducible Release

Permitted statement. The identified implementation artifact was reproducibly constructed under the stated recipe and reproduced identity.

B.5 Release-envelope authentication

Permitted statement. The identified key authenticated the exact canonical release-manifest bytes. This statement does not assert Protocol Conformance unless the enclosed report independently PASSes.

End of specification.