Universal Manifest v0.5 — Conformance

Implementation requirements

v0.5.0. Published as tag v0.5.0 of UniversalManifest/universal-manifest, with the same wire files as release candidate 1 (v0.5.0-rc.1). It is a mechanism release, not a completed implementation or conformance qualification. Immutable versions in UniversalManifest/universal-manifest are normative; main is the working version. Earlier versions were reviewed in the OMA3 Technical Working Group, and publication asserts no OMA3 ratification.

8 November 2026

This document: /spec/v0.5/conformance/

Document set: Universal Manifest v0.5.0 · What changed in v0.5.0

Artifacts: JSON Schema · JSON-LD context

History: https://universalmanifest.net/spec/ (version index and changelog)

Author: Grig Bilham — inventor, architect, and author of the Universal Manifest.

Copyright © 2026 SumSet Tech LLC and the Universal Manifest contributors. This specification is available under the Apache License, Version 2.0. It is free and remains free. Earlier published revisions retain their original licence notices.

Universal Manifest v0.5 - Conformance

> **v0.5.0.** Published as tag `v0.5.0` of `UniversalManifest/universal-manifest`, with the same wire files as release candidate 1 (`v0.5.0-rc.1`). Wire version `0.5`; namespace `https://universalmanifest.net/ns/v0.5`; reading routes `/spec/v0.5/`. It is a mechanism release, not a completed implementation or conformance qualification. Feature-specific PREVIEW and conditional requirements remain in force.

v0.5.0 inherits the accepted v0.4 editorial contract, adds a distinct wire/namespace identity and selects the stricter Profile A predicate. It is published and remains unqualified as a complete implementation. Migration and release notes distinguish old public bytes, inherited features and candidate changes.

Conformance in v0.5 is defined against the **abstract data model**, not against any single serialization. An implementation conforms by correctly handling the abstract properties and semantics of a manifest after consuming it through at least one production rule (Base Section 4.5).

Primary references:

- Spec text: the immutable revision of UniversalManifest/universal-manifest identified by a specification release is its normative source; `/spec/v0.5/` is the generated reading view, starting with the Base (Tier-0 / Tier-1 Core).
- `spec/v0.5/README.md`
- Signature Profile A: Base §1.6, including the v0.5 acceptance boundary. Earlier profile history: `spec/v0.2/SIGNATURE-PROFILE.md`.

1) Conformance targets

v0.5 defines conformance for:

- **Evaluator (Consumer).** Consumes a v0.5 manifest through a production rule and processes it through the six-stage evaluation sequence, producing a structured receipt.
- **Holder.** Assembles and presents a v0.5 manifest, including (at Tier 1 or above) the holder-binding and presentation-proof material the relying party will verify.
- **Bilateral Participant.** Implements both Evaluator and Holder behaviour. In a bilateral exchange (Base Section 6.4.6), both parties independently evaluate the other's manifest; the interaction tier floor is the maximum of both parties' `requiredTrustTier` — a negotiated floor, distinct from the `effectiveTrustTier` each party verifies and records in its receipt.

A conformance claim names the **tiers** and **optional features** it covers. The tier stack is additive: a Tier-1 claim implies Tier-0; Base + EXT-T1 is the Tier-1 surface; Base + EXT-T1 + EXT-T2 is the Tier-2 surface; and so on. Optional features (statusRef, bilateral sessions, federation, CBOR-LD, scope registry, and the rest of EXT-OPT) are claimed independently of tier.

Claims and reports MUST identify the implemented, partial, or not-implemented state of each claimed obligation, its exact source section, actual executed checks/evidence, finite limits, and the mandatory baseline for every unmet check, following EXT-OPT O11.1. A partial declaration does not waive a requirement or raise verification, assurance, or trust tier. Normative checks, structural PREVIEW checks, and unexecuted design-stage candidates remain distinct; a matching fixture verdict does not prove an unexecuted cryptographic primitive. This reporting rule creates no manifest member or wire capability carrier.

Bounded selective minimum disclosure is part of the v0.5 Base behavior. In plain language, each viewer, verifier, system, or use case receives only the minimum information it is allowed or needs to see. A v0.5 Base conformance claim covers this bounded profile through Stage 3 Project, Stage

4 Consent, sealed-entry handling, and structured receipts. It does not claim unlinkability, full or partial, cover-slot indistinguishability, resolver-operator privacy, or complete residual-fingerprint resistance. No v0.5 claim establishes them.

2) Required behaviour (evaluator)

For the bounded selective-minimum-disclosure profile, the Project and Consent stages work together: the evaluator processes only the facets, claims, pointers, and device entries relevant to the viewer/verifier/system/use case and authorized by the interaction context; withheld facets are not projected and are not evidence of absence; and the receipt records the resulting per-facet outcome honestly.

An evaluator claiming v0.5 conformance MUST:

Deliberately offline/no-resolver evaluation records `keyRefResolution: "not-attempted"`; failed/unavailable attempted resolution records `"unresolved"`; authorized matching resolution records `"resolved"`. Unattempted/unresolved keyRef-derived identity remains Tier0, and embedded-key mathematics is not identity authority. Present but unexecuted presentation proof verification records `"not-evaluated"`; actual absence records `"absent"`. Required interactive proofs refuse on absence, failure or non-evaluation. These are honest diagnostics, not adoption of a signed-receipt mandate or additional evidence carrier.

1. Execute all six stages of the evaluation sequence in order (Arrive, Verify, Project, Consent, Compose, Receipt). See spec text Section 3.1.
2. Consume the manifest through at least one conformant production rule and operate on the resulting abstract data model. JSON-LD is the reference encoding; an evaluator MAY additionally support CBOR-LD or another conformant encoding. See spec text Section 1.7 and Section 4.5.
3. Verify the signature over the canonical signing bytes defined by the production rule, retaining required envelope fields, TTL rejection and the `keyRef` ↔ `publicKeySpkiB64` consistency check. For wire `0.5` Profile A, MUST enforce all six requirements of Base §1.6.1.1; earlier-version compatibility paths retain their own profile rules.
4. Pin the content-hashed v0.5 context document (SHA-256 `27ab5a3cc491cee7a5947b73f550c07ecbff285a81a6ba801aa98f60046dcea4`) and verify its hash before trusting term definitions; an evaluator MUST NOT dereference an arbitrary remote context at evaluation time. The expected hash is the sha2-256 multihash `zQmR1W9vsPrdbXmhH38jdNhdEhBB1C7JUL1Z9kZMi6u16i7`, computed over the exact bytes of

the versioned context document (verbatim body and computation rule: Base Appendix B, EXT-OPT Section O1.7). See spec text Section 6.10.

5. Produce a structured receipt for every processed manifest. The receipt MUST include `@type` (with `um:Receipt`), `manifestId`, `outcome`, `signatureCheck`, `freshnessCheck`, and `facetStatuses` (empty array when the manifest contains zero facets). See spec text Section 3.3.
6. Record each per-facet status using the v0.5 status set: `processed`, `opaque`, `consent-denied`, `consent-missing`, `trustTierUnsupported`, `assuranceInsufficient`, `not-projected`, or `written` (`assuranceInsufficient` and `written` arise from the assurance-floor and write-authorization PREVIEW features). See spec text Section 3.3.1.
7. Treat encrypted facets as sealed entries when the evaluator lacks a decryption key, and MUST NOT reject a manifest solely because it contains undecryptable encrypted facets. See spec text Section 2.3 and Section 3.1.4.
8. Match `consents[].facetRef` against each projected facet's `@id`. When no matching consent entry exists, record `consent-missing` and MUST NOT process the facet's data. Treat expired or withdrawn consents as absent. See spec text Section 1.4.4 and Section 3.1.4.
9. For any claim relied upon at Tier 1 or above, require a verified holder binding; a claim lacking a verifiable holder binding MUST be capped at Tier 0 (`holderBindingStatus: "failed"` for a failed binding, `"absent"` for a missing one). See spec text Section 6.6 and EXT-T1.
10. When an interactive presentation is required, require a `presentationProof` whose `challenge` and `audience` match the verifier-issued challenge; reject or cap claims that fail this proof-of-possession. Of the three `proofType` values, only `did-auth` is exercised by the fixture suite and reference implementation today; `sd-jwt-kb` and `bbs-derived` are at-risk (see Section 9) and MUST fail closed when unimplemented. See EXT-T1 Section T1.2.
11. MUST NOT treat a stale liveness attestation as equivalent to current human presence; an attestation past its `validUntil` is `unknown` regardless of `attestedAt`. See EXT-T1 Section T1.3.
12. Respect `requiredTrustTier` declarations at the manifest, claim, and facet levels. Tier floors raise only - claim and facet floors MUST NOT lower the manifest-level floor. See spec text Section 6.4.5.
13. When claiming the per-facet floors PREVIEW feature, respect `requiredLiveness` floors: a facet carrying the floor MUST NOT be processed unless a validated `livenessAttestation` satisfies the requested `minFreshness` and, when required, `userVerified: true`. An evaluator that does not implement floor enforcement MUST fail closed: a floor-bearing facet is withheld, never processed (Base Section 4.7). See EXT-T1 Section T1.3.1.

14. When claiming the per-facet floors PREVIEW feature, respect `requiredAssuranceClass` floors: a facet below its required class MUST be withheld, MUST NOT be downgraded, and MUST be recorded `assuranceInsufficient` (the `assuranceStatus` echo object SHOULD additionally be recorded). The same fail-closed baseline applies to evaluators that do not implement floor enforcement. See EXT-T1 Section T1.3.3.
15. Record `trustTierUnsupported` in the receipt when the required tier has no implemented verification profile (e.g., a Tier-2 or Tier-3 requirement on a Tier-1 evaluator), and MUST NOT downgrade to a lower tier.
16. Apply the Section 3.1.2 clock-skew discipline at its stated normative levels: the tolerance for `issuedAt`/`expiresAt` comparisons SHOULD be no more than 60 seconds; manifests with `issuedAt` more than 60 seconds in the future SHOULD be rejected with `freshnessCheck: "stale"`; wider tolerances MUST be documented in the conformance claim. See spec text Section 3.1.2.
17. Preserve unknown members for processing purposes (so the signature still verifies) and act only on recognized members. Unknown members MUST NOT be stripped before signature verification and MUST NOT change a decision.
18. When `signature.statusRef` is present and the evaluator claims revocation-aware verification, resolve revocation status per the statusRef resolution schema (Section 3.4) and record `revocationStatus`. Evaluators not implementing revocation-aware verification MUST report `revocationStatus: "unchecked"` and MUST NOT claim revocation-aware conformance.
19. When claiming the private-data PREVIEW features, enforce their fail-closed limits: per-facet isolation claims are cross-field structural obligations, writes require target-facet write consent, unlock windows never lower per-facet floors or isolation, key-lifecycle receipt events are assertions rather than erasure proof, and derived writes MUST NOT down-label when a profile supplies a sensitivity ordering. See Base Sections 2.3.5, 3.1.4, 7.2 and EXT-OPT Sections O3 and O4.5.

3) Required behaviour (holder)

A holder claiming v0.5 conformance MUST:

1. Set `manifestVersion: "0.5"` and include the v0.5 versioned context `https://universalmanifest.net/ns/v0.5` in `@context`.
2. Generate an opaque `@id` (e.g., `urn:uuid:<uuidv4>`) per Section 1.2.2, and produce the manifest through a conformant production rule.

3. Bind `subject` to a stable identifier URI (typically a DID), and bound `expiresAt` to a sensible interaction lifetime relative to `issuedAt`.
4. Sign the manifest using a recognized signature profile, with `signature.keyRef` present and `signature.value` populated over the canonical signing bytes. Signature Profile A (`signature.algorithm: "Ed25519"`, `signature.canonicalization: "JCS-RFC8785"`) remains the default and the mandatory-to-implement evaluator baseline. A holder MAY use Signature Profile B (`signature.algorithm: "ES256"`, `signature.canonicalization: "JCS-RFC8785"`) when supported by the intended evaluator, under EXT-OPT O15; its signature value is the raw 64-byte `R || S` encoding. When evaluator support is unknown, the holder uses Profile A or accepts fail-closed rejection; it MUST NOT assume Profile B support or fall back to an unsigned manifest.
5. Where consents are used, include them as full v0.5 consent entries with `@id`, `@type: "um:Consent"`, `facetRef` matching a facet's `@id`, `scope`, `purpose`, `grantedAt`, and `expiresAt`.
6. Where encrypted facets are used, set `encryptionProfile: "jwe-inline-v1"` on the facet and supply a conformant JWE JSON Serialization object as the `entity` value, using an algorithm pair consistent with the JWE algorithm constraints. See spec text Section 2.3.2 and Section 2.4.
7. Where per-facet floors are used, provide well-formed `requiredLiveness` and `requiredAssuranceClass` values and the liveness/provenance evidence needed for the evaluator to satisfy or refuse those floors. See EXT-T1 Section T1.3.
8. For any claim intended to be relied upon at Tier 1 or above, include a `holderBinding` with a valid `mode` (`sd-jwt-kb`, `bbs-holder-commitment`, or `reciprocal-control`) and the fields that mode requires (e.g., `cnfThumbprint` for `sd-jwt-kb`; `boundDid`, `subjectProof`, and `boundDidProof` for `reciprocal-control`). See EXT-T1 Section T1.1.
9. When presenting interactively, include a `presentationProof` containing `proofType`, `challenge`, `audience`, `created`, and `proofValue`. See EXT-T1 Section T1.2.
10. Where Tier-2 binding is asserted, include a `bindingProof` of a defined `type` (`ZkLinkedSecretProof` or `ZkHdDerivationProof`) with the fields that type requires. See EXT-T2 and spec text Section 6.4.7.
11. Where Tier-3 ceremony is asserted, include a `ceremonyProof` (`type: "ThresholdAttestationProof"`) with `threshold`, `attesters`, `ceremonyId`, and `aggregateProof`. See EXT-T3 and spec text Section 6.4.8.
12. Where `identity.crossDidBinding` claims are issued, populate all required fields (`@type`, `issuer`, `boundDids` containing at least 2 DIDs) and, for attester-asserted bindings, the attester triple (`attester`, `attestationMethod`, `attestedAt`); the triple is OPTIONAL when a

cryptographic `bindingProof` or `ceremonyProof` carries the binding. See spec text Section 6.4.4.

13. Where `um:agentDelegation` pointers are issued, populate all required fields (`@type: "um:agentDelegation"`, `delegateType`, `delegatedBy` matching the manifest `subject`, `delegatedAt`, `expiresAt`), using registered scope values where the scope registry applies. See spec text Section 6.5 and EXT-OPT.
14. Where write authorization, unlock-window, key-lifecycle receipt, or sensitivity-ordering PREVIEW features are claimed, use the wire members and scope/event/profile tokens defined by Base Sections 3.1.4 and 7.2 plus EXT-OPT Sections O3 and O4.5.

4) Required behaviour (bilateral participant)

A Conformant Bilateral Participant MUST implement both Evaluator Behaviour (Section 2) and Holder Behaviour (Section 3). In a bilateral exchange:

- Both parties present manifests.
- Both parties independently evaluate the other's manifest through the six-stage sequence.
- The interaction tier floor for the exchange is the maximum of either party's `requiredTrustTier` — a negotiated floor, distinct from the `effectiveTrustTier` each party verifies and records in its receipt.
- For Sybil-critical or high-risk actions, parties MUST fail closed when required tier checks are not satisfied.
- For lower-risk actions, parties MAY degrade to a restricted mode that excludes trust-transitive or high-impact operations.
- Receipts produced within a bilateral session MUST carry `exchangeId` and `evaluatorId`. The full session object, paired-receipt correlation, and receipt hash-chaining are specified in EXT-OPT Section O4.

See spec text Section 4.3 and Section 6.4.6; EXT-OPT Section O4.

5) Conformance fixtures

Dated finite TypeScript qualification (2026-10-21). The candidate index, SHA-256 `6334fe4f61198d087ddb8260ba467ba42507dfb852d2643d5de84619dc804e3b`, contains **188 cases**, all matched by the actual command adapter. The strict Draft 2020-12 schema/ingress

sweep passed **192 checks** (188 indexed inputs, three examples, one targeted malformed-JWE control); the current API qualification passed **38 named groups**, including the packaged-context byte/hash check, corrupted-module rejection, trusted-context and scoped-refusal controls, resolver-not-called behavior, dense-list enforcement, opaque-data preservation and historical-version controls. This evidence pins TypeScript source SHA-256 `7f38f3480bda854ab66e75723fbeb2509768211f0aef85df9cad46ba1b54b29e`, built implementation SHA-256 `e561f61d6308a87879880aaf08c81208abe2835007e9dcb9fd12f365d0e4094b`, bundled context-module source SHA-256 `4ca50655a6bf28eb7f26cb16072a2cece498629fcdefde3533315bb4f988e7f8`, and built context module SHA-256 `44f0748b9e64491f054d3f46816fc344ed7ac2b46fa4d933388564abef7774a1`. These implementation hashes identify the bounded reference; they are not additional wire/context pins. All 164 earlier fixture bodies and every original verdict, receipt expectation and caller context remain unchanged, as does all supplemental JWE index data. The earlier 114-to-164 expansion added fifty signed consent/time/operation/gate-order/trust-floor controls and clarified two caller `read` contexts; this context repair does not change those again. The 24 additional cases have independently valid outer signatures: 11 accepted controls and 13 context refusals. Twelve refusals are structurally valid; the empty-root-array refusal is structurally invalid. Schema validity alone does not establish trusted context interpretation. The separately dated **46/46 I-JSON checks (2026-10-16)** remain distinct from the changed evaluator's current runs. The report deliberately returns `conformanceLevel: "none"`; positive Tier-1+ verification, authenticated services, other holder implementations and protected/private mechanisms remain unproved. Reader and public-package propagation require separate acceptance of these changed inputs.

Fixed offline reference context (implementation limit). Before interpreting UM terms, the reference implementation verifies the actual packaged normative UTF-8 context bytes against the existing content hash. Its trusted reference forms are the exact `https://universalmanifest.net/ns/v0.5` scalar or a nonempty dense list containing only that reference; repeated trusted references are supported. In interpreted UM scopes, including the root, facets, claims and consents, substitutions, inline remaps, resets and arbitrary remote contexts refuse before signature execution or resolver use. Refusal receipts report `signatureCheck: "not-evaluated"`, `freshnessCheck: "not-evaluated"` and `keyRefResolution: "not-attempted"`. This is fixed offline context support, not arbitrary JSON-LD expansion or remote loading, and not a new restriction on the normative wire model. Entity data, embedded credential/VP proof payloads and unknown-member payload contexts remain byte-preserved, uninterpreted signed data; they confer no UM interpretation, proof verification, identity or authority. A separate selected profile or proof processor must establish its own semantics and verification.

Receipt check assertions (2026-10-30). The candidate index now asserts `signatureCheck` and `freshnessCheck` on all 119 fixtures in evaluation mode, with values derived from each fixture's bytes and evaluation clock. Its SHA-256 is `940e23629c246596ef91c7ed7dc8f6fef6242fa6af3b287a2ba85555bc872a19`. The case count stays 188, and entries dated earlier describe the earlier index. The TypeScript and Rust adapters each match 188 of 188. Rewriting either field in a receipt now fails every fixture that asserts the rewritten value.

Known limit on malformed `created` (2026-10-30). A malformed `signature.created` fails validation under the schema, and section 1.6.2 of the Base says so. The current TypeScript and Rust evaluators do not yet agree with the schema on every malformed value, so no claim may say they conform on those values. A later package rebuild carries the correction.

Resource limits, consent entries and `created` (2026-10-31). The candidate index adds eight signed cases and now holds **196 cases**, SHA-256 `e8c72773514ddd0822c5b2b81f890ae9b557bad775309def1be70ed58b053434`. Six test the Base section 6.3 limits at each boundary, and two test more than one consent entry on a facet. The TypeScript and Rust command adapters each match 196 of 196, and 79 of 79 on the v0.4 index. Entries dated earlier describe the earlier index. Both evaluators now apply the schema's `date-time` rule to `signature.created` on the v0.5 path, so the known limit above is corrected for the values tested. The Base section 6.3 limits apply to the v0.5 path, and v0.4 keeps its own rules.

I-JSON ingress (2026-11-06). A fresh check found that the TypeScript evaluator skipped the size limit when a manifest could not be canonicalized, and accepted a 1.2 MB signed manifest with a lone surrogate inside `signature`. The Rust evaluator rejected it. Base section 1.6.3 now requires the whole manifest as received to be I-JSON, including members outside the signing input. Both evaluators now reject such input: Rust and the TypeScript ingress parser at parse, and the TypeScript evaluator at Arrive with `um:reason:structure:malformed`. The candidate index adds two signed cases and now holds **198 cases**, SHA-256 `f3c812984f47f284a3254dc8d51b1cba8140a06ba41238f6bce3b7f8e4cdb97c`. The TypeScript and Rust command adapters each match 198 of 198, and 79 of 79 on the v0.4 index. The v0.5 schema sweep passes 202 checks. Entries dated earlier describe the earlier index.

Known differences between the two reference evaluators (2026-11-06). These are outside the v0.5 suite, and both are documented deviations:

- **v0.4 path.** Rust applies nesting, array and size limits of 10, 1,000 and 1 MiB. TypeScript applies none. The two also differ on `signature.created` values with hour `24` and on a leap second outside 23:59 UTC. v0.4 is published, so neither v0.4 path changes.

- **No receipt at parse.** Rust refuses at parse, and returns no receipt, past 256 levels of nesting and for input that is not I-JSON. TypeScript returns a rejected receipt for the same input. Both reject.

Corrected evaluation boundaries (2026-10-21). The earlier fifty controls exercise `grantedAt <= now < expiresAt`, a caller-supplied nonempty operation list whose every member is covered by consent, malformed caller context/clock/ceiling refusal, and sealed-entry → consent → assurance/liveness/trust-floor order. A configured trust ceiling or successful key lookup does not create implemented Tier-1+ support; the current implementation's floor decisions use its actual Tier-0 support. Mixed-facet cases and adjacent positives check the resulting receipt and per-facet decisions. Matching purpose is necessary at the purpose gate; omitted, empty and mismatching purposes produce `consent-denied` / `purpose-mismatch`, while malformed required consent purposes structurally reject before signature verification. The two persisted encrypted-Entity cases retain valid outer signatures and all four opaque facet rows in the actual command run. Refusal evidence does not establish positive verification, decryption or full conformance.

Detached cryptographic evidence (2026-10-16). The TypeScript and Rust production Profile A predicates agreed with the retained expectations on all **937 identical inputs (3 accept, 934 reject)**. This qualifies that finite signature-predicate boundary only. It does not establish full evaluator or SDK parity. Four accepted two-recipient ciphertexts containing two real JSON `um:Entity` plaintexts remain unchanged in the two corpus manifests signed for that qualification. Separate Rust/panva-jose calls read those persisted signed files and passed **16 exact-plaintext opens: 8 cross-implementation and 8 self-controls**. TypeScript's four opaque facet results establish sealed-entry handling, not decryption. Earlier Entity evidence of **112 ordered wrong-key refusals reaching unwrap (56 per consumer) and 24 tamper refusals** is reused because the ciphertexts, decryptors and dependencies are exactly unchanged; those matrices were not rerun. The older primitive experiment's 8 cross-opens, 36 wrong-key refusals and 156 malformed/tampered refusals remain a separate experiment. Neither supplemental decryption calls nor reused negative controls add entries to the current 188-case ordinary suite; primitive free-text payloads are not conformant Entity fixtures.

The Rust JWE software subset's unsupported parameters and size/key limits are described as implementation limits in Base §2.4.2. They are not universal JOSE prohibitions. Successful authorized opens must precede ordered cross-facet key attempts, and lookup failures must remain distinct from unwrap/authentication failures (Base §2.3.5). Software evidence does not establish hardware custody, recipient authority, global entropy quality, or complete privacy guarantees. All 43 evaluator/holder/bilateral/suite obligations and conditional requirements remain in force.

To claim v0.5 conformance, an implementation MUST:

1. Process every fixture listed in `conformance/v0.5/expected.json`.
2. Match each fixture's `expectedResult`; do not infer expected behavior from the directory name alone.
3. For evaluation-mode fixtures, emit a structured receipt whose outcome and facet status match the expected outcome.
4. For schema-only verification, follow each fixture's `schemaValidation.expectedResult` / `classification` where present. The v0.5 ceremony shape requires unique attester IDs; the evaluator additionally enforces positive M,N and M ≤ distinct attesters ≤ N. A cross-DID `bindingProof` permits exactly two boundDids; ceremony alone may bind more. Valid structure does not verify an aggregate or choose its protocol.

5.1 Valid fixtures

Location: `conformance/v0.5/valid/`. The migrated index retains evaluation-mode `accepted-partial` / `accepted-with-warnings` outcomes and receipt-observable statuses, including an assurance-floor reject and an explicitly routed v0.3 envelope that the v0.5 schema rejects. The earlier fifty signed controls also live here: valid signatures can accompany denied consent, unsupported floors or malformed caller inputs. The new context cases add 11 accepted controls here and 13 refusals under `invalid/`; all 24 have independently valid outer signatures. Context refusals precede signature execution and resolver use, so compare the indexed `not-evaluated` / `not-attempted` receipt states rather than inferring verification from valid test signatures. Preserve each indexed `evaluationContext`, including intentionally omitted, empty or malformed values; never infer requested operations or purpose from a grant. Compare all indexed assertions, including exact ordered `expectedBehavior.receipt.facetStatuses` arrays for mixed facets, rather than only the first row. The portable fixture guide describes the full assertion contract. Directory placement alone is not an oracle; use each indexed expected outcome:

- `minimal-signed-manifest.jsonld` - baseline v0.5 signed envelope (no facets, no claims) verifying under Signature Profile A.
- `minimal-es256-signed-manifest.jsonld` - the same baseline envelope verifying under Signature Profile B (JCS + ES256; EXT-OPT Section O15). The Section 1.6.3 signing input is byte-identical to Profile A - only the signature primitive differs - and `signature.value` is the base64url of the 64-byte raw `R || S` (EXT-OPT Section O15.1).
- `es256-high-s-signature.jsonld` - a genuine Profile B signature whose `S` component is deliberately above `n/2` ("high-S"), which **MUST be accepted**. Profile B imposes no low-S rule and no RFC 6979 requirement (EXT-OPT Section O15.2): Secure Enclave / StrongBox / TPM signers expose no nonce control and do not guarantee low-S, so a canonical-S rule would

make every hardware-resident key non-conformant. An evaluator that rejects this fixture has added a malleability check the profile forbids.

- `es256-signed-manifest-with-facets.jsonld` - Profile B over a body carrying facets and consents, showing the signing input is suite-independent.
- `manifest-with-holder-binding.jsonld` - claim carrying a `holderBinding` in `sd-jwt-kb` mode per EXT-T1 Section T1.1.
- `manifest-with-bilateral-session-receipt.jsonld` - manifest with facets and consents suitable for bilateral session evaluation.
- `manifest-with-zkp-binding-proof.jsonld` - claim carrying a `ZkLinkedSecretProof` `bindingProof` per Section 6.4.7.
- `manifest-with-ceremony-proof.jsonld` - claim carrying a `ThresholdAttestationProof` `ceremonyProof` per Section 6.4.8 (transitional: predates the `protocol` member, which is REQUIRED only at wire freeze - EXT-T3 Section T3.1).
- `manifest-with-agent-delegation-scopes.jsonld` - `um:agentDelegation` pointer with core scope values per Section 6.5.
- `manifest-with-encrypted-facet.jsonld` - facet using the JWE inline profile (carried forward from v0.3).
- `manifest-with-isolated-facets.jsonld` - two encrypted facets with distinct recipient `kid` values and distinct IVs for the per-facet isolation smoke path.
- `facet-required-liveness.jsonld` - facet carrying a well-formed `requiredLiveness` floor.
- `consent-with-write-scope.jsonld` - consent whose `scope` includes a write operation for a target facet.
- `receipt-facet-key-shredded.jsonld` - receipt manifest carrying `facet-key-rotated` / `facet-key-shredded` lifecycle events.
- `consent-unlock-window.jsonld` - consent carrying `unlock.window` scope and an enumerated unlock-window facet set.
- `facet-required-assurance-class.jsonld` - locked facet carrying `requiredAssuranceClass: "hardware-bound"` plus a locked liveness floor.
- `facet-assurance-hardware-uv.jsonld` - facet carrying the intermediate `requiredAssuranceClass: "hardware-uv"` rung.
- `facet-assurance-software.jsonld` - facet carrying the default/lowest `requiredAssuranceClass: "software"` rung.

- `manifest-locked-facet-below-floor.jsonld` - structurally valid manifest that MUST reject in evaluation mode with `assuranceInsufficient` when presented with a software-class unlock below a `hardware-bound` floor.
- `manifest-with-unknown-members.jsonld` - unknown root members, an unknown entity field, and a devices array, all covered by a genuine signature; an implementation that strips unknown members before verification fails this fixture.
- `manifest-with-bbs-holder-binding.jsonld` - `bbs-holder-commitment` holder binding carrying its required `commitment` and `proofValue` plus an optional `pseudonymScope` per EXT-T1 Section T1.1.1.
- `manifest-with-reciprocal-holder-binding.jsonld` - `reciprocal-control` holder binding carrying its required `boundDid`, `subjectProof`, and `boundDidProof` per EXT-T1 Section T1.1.
- `manifest-with-presentation-proof.jsonld` - interactive presentation with a genuine did-auth `presentationProof` per EXT-T1 Section T1.2; the manifest signature excludes `presentationProof` from the signing input, and `keyRefResolution: "unresolved"` caps keyRef-derived identity at Tier 0.
- `manifest-with-liveness-attestation.jsonld` - well-formed `livenessAttestation` evaluated inside its validity window; `livenessStatus.freshnessClass` records `live` per EXT-T1 Section T1.3.
- `liveness-attestation-past-valid-until.jsonld` - stale-liveness replay check: an attestation past its `validUntil` classifies `unknown` regardless of `attestedAt` and is never treated as current human presence.
- `agent-delegation-fail-closed-defaults.jsonld` - delegation with an empty `scope` (grants no capabilities) and no `delegateId` (attributed to no agent); both fail-closed defaults are receipt-observable warnings (`accepted-with-warnings`).
- `facet-without-consent.jsonld` - facet with no governing consent entry records `consent-missing` and is not processed (`accepted-partial`).
- `consent-expired.jsonld` - expired consent treated as absent: facet `consent-denied` with consent status `expired`.
- `consent-withdrawn.jsonld` - withdrawn consent (`withdrawnAt` set) treated as absent: facet `consent-denied` with consent status `withdrawn`.
- `consent-scope-mismatch-read.jsonld` - intended `read` operation not literally present in the consent `scope`: facet `consent-denied` with consent status `scope-mismatch`.
- `consent-unknown-condition.jsonld` - consent condition the evaluator cannot recognize or enforce fails closed: facet `consent-denied` with consent status `condition-violated`.

- `encrypted-facet-sealed-entry.jsonld` - encrypted facet the evaluator cannot decrypt is acknowledged as a sealed entry (`opaque`, outcome `accepted-partial`), never rejected solely for being undecryptable.
- `claim-unrecognized-type.jsonld` - unrecognized claim `@type` treated as unprocessable (present but unverifiable above Tier 0) and recorded in `claimStatuses`.
- `pointer-unrecognized-type.jsonld` - unrecognized pointer `@type` recorded as present (receipt `unprocessedEntries`) and not acted on.
- `claim-tier-cap-no-holder-binding.jsonld` - claim relied upon at Tier 1 with no `holderBinding` is capped at Tier 0: `holderBindingStatus: "absent"`, `effectiveTrustTier` 0, claim recorded `trustTierUnsupported`.
- `claim-tier-unsupported-by-evaluator.jsonld` - Tier-2 claim on a Tier-1 evaluator treated as unverifiable and recorded `trustTierUnsupported` without downgrading (`accepted-partial`).
- `cross-did-binding-proof-only.jsonld` - `identity.crossDidBinding` carrying a cryptographic `bindingProof` and no attester triple; the attester fields are required only for attester-asserted bindings per Section 6.4.4.
- `v03-backcompat-manifest.jsonld` - v0.3 envelope processed through the EXT-T1 Section T1.6.4 backwards-compatibility path (`absent` binding statuses, tier capped at 0, unbound-claims warning); the v0.5 structural schema rejects it while the evaluator accepts it.

5.2 Invalid fixtures

Located under `conformance/v0.5/invalid/`:

- `holder-binding-invalid-mode.jsonld` - `holderBinding.mode` outside the enum (`sd-jwt-kb` | `bbs-holder-commitment` | `reciprocal-control`).
- `holder-binding-missing-cnf.jsonld` - `sd-jwt-kb` mode without the required `cnfThumbprint`.
- `ceremony-proof-insufficient-attesters.jsonld` - `ceremonyProof.attesters` smaller than the M required by `threshold`.
- `binding-proof-invalid-type.jsonld` - `bindingProof.type` not `ZkLinkedSecretProof` or `ZkHdDerivationProof`.
- `missing-signature.jsonld` - no `signature` member.
- `claim-tier-below-manifest-floor.jsonld` - `claim.requiredTrustTier` lower than `manifest.requiredTrustTier` (Section 6.4.5; the floor raises only).

- `required-liveness-bad-minfreshness.jsonld` - `requiredLiveness.minFreshness` uses a value outside `live` / `recent` / `stale`.
- `required-liveness-not-object.jsonld` - `requiredLiveness` is not an object.
- `facet-assurance-class-bad-enum-shape.jsonld` - `requiredAssuranceClass` uses a value outside `software` / `hardware-uv` / `hardware-bound`.
- `facet-assurance-class-not-string-shape.jsonld` - `requiredAssuranceClass` is not a string enum value.
- `bad-signature-value.jsonld` - cryptographically wrong `signature.value` (bit-flipped after signing); Ed25519 verification over the Section 1.6.3 signing input fails, while a signature-presence-only implementation would accept.
- `tampered-after-signing.jsonld` - `subject` mutated after signing, so the signature no longer covers the canonical bytes.
- `key-substitution-mismatch.jsonld` - embedded `publicKeySpkiB64` substituted with a foreign key after signing (the Section 1.6.4 key-substitution attack).
- `expired-manifest.jsonld` - evaluation clock past `expiresAt`; the signature is genuine and only TTL math rejects (`freshnessCheck: "expired"`).
- `issued-after-expires.jsonld` - `issuedAt` later than `expiresAt` (cross-field temporal misuse over a genuine signature).
- `future-issued-at.jsonld` - `issuedAt` more than 60 seconds in the future; rejected with `freshnessCheck: "stale"` per Section 3.1.2.
- `manifest-version-mismatch.jsonld` - `manifestVersion: "0.6"` is supported by no v0.5 processing path (v0.3 envelopes instead take the EXT-T1 Section T1.6.4 backwards-compatibility path).
- `unsupported-signature-profile.jsonld` - `signature.algorithm: "ES384"` is not a recognized suite, so the `(algorithm, canonicalization)` pair names no registered profile; rejected with `signatureCheck: "unsupported-profile"` (Section 1.5.1/1.6.5). ES384 is a well-formed IANA/JOSE algorithm token that Universal Manifest does not register, so the fixture exercises the unknown-profile path and nothing else: the manifest carries a genuine Ed25519 Profile A signature over a matching Ed25519 SPKI, and only the suite token is unrecognized. Rejection MUST occur at the profile-pair check, before any key or value inspection, and the pair MUST NOT be reinterpreted as a recognized suite. The test is re-keyed off `"ES256"`, which became a recognized suite (Signature Profile B) in v0.5; it remains unconditional because the schema `algorithm` enum structurally rejects any token outside the registered set.

- `es256-bad-signature-value.jsonld` - cryptographically wrong Profile B `signature.value` (bit-flipped after signing); ECDSA P-256/SHA-256 verification over the Section 1.6.3 signing input fails.
- `es256-tampered-after-signing.jsonld` - `subject` mutated after Profile B signing, so the signature no longer covers the canonical bytes.
- `es256-key-substitution-mismatch.jsonld` - embedded `publicKeySpkiB64` substituted with a foreign P-256 key after signing (the Section 1.6.4 key-substitution attack); the suite and key type are both correct, only the key is wrong.
- `es256-der-encoded-signature-value.jsonld` - a cryptographically genuine Profile B signature encoded as 72-byte DER/X9.62 instead of the mandated 64-byte raw `R || S` (EXT-OPT Section O15.1); DER MUST be rejected. Because the profile pair is recognized, the reason code is `signatureCheck: "invalid"` (a malformed value under a recognized profile), **not** `"unsupported-profile"`.
- `es256-spki-algorithm-mismatch.jsonld` - an Ed25519 SPKI declared under `algorithm: "ES256"`; the REQUIRED SPKI-algorithm-vs-`algorithm` consistency check (EXT-OPT Section O15.4) fails, in either direction. Reason code `signatureCheck: "invalid"`, not `"unsupported-profile"`.
- `type-missing-um-manifest.jsonld` - `@type` does not include `um:Manifest`.
- `context-wrong-namespace.jsonld` - `@context` carries a substituted namespace that cannot match the pinned v0.5 context document (Section 6.10 context substitution).
- `holder-binding-reciprocal-missing-fields.jsonld` - `reciprocal-control` mode without `subjectProof` / `boundDidProof`; rejected as a binding failure.
- `holder-binding-bbs-missing-commitment.jsonld` - `bbs-holder-commitment` mode without its required `commitment` field (EXT-T1 Section T1.1.1).
- `presentation-proof-challenge-mismatch.jsonld` - `presentationProof.challenge` does not match the verifier-issued nonce; replay-suspect, rejected for interactive verification.
- `presentation-proof-audience-mismatch.jsonld` - `presentationProof.audience` does not match this verifier; rejected for interactive verification.
- `presentation-proof-missing-when-challenged.jsonld` - the verifier issued a challenge and the manifest carries no `presentationProof`; `presentationProofStatus: "absent"`, rejected for interactive verification.
- `liveness-missing-valid-until.jsonld` - `livenessAttestation` without its required `validUntil` (EXT-T1 Section T1.3).

- `agent-delegation-delegatedby-mismatch.jsonld` - `um:agentDelegation` whose `delegatedBy` does not match the manifest `subject` (Section 6.5.1).
- `agent-delegation-expired.jsonld` - `um:agentDelegation` past its `expiresAt`; expired delegations are rejected (Section 6.5.1).
- `encrypted-facet-malformed-jwe.jsonld` - encrypted facet whose JWE lacks `iv` and `tag`; structural conformance requires all five JWE JSON Serialization fields (Section 2.3.2).
- `facet-tier-below-manifest-floor.jsonld` - `facet.requiredTrustTier` lower than `manifest.requiredTrustTier` (Section 6.4.5; the floor raises only).
- `v03-backcompat-tier-floor-unsatisfiable.jsonld` - v0.3 envelope declaring a Tier-1 floor that cannot be satisfied at the backwards-compatibility Tier-0 cap; rejected per Section 6.4.5 and EXT-T1 Section T1.6.4.
- `claim-proof-oversize-vp.jsonld` - embedded `claimProof` VP over the 50 KB per-VP limit (Section 6.4.3).
- `missing-subject.jsonld` - missing required `subject` (Section 1.3.2).
- `missing-expires-at.jsonld` - missing required `expiresAt` (Section 1.3.3).
- `consent-missing-facet-ref.jsonld` - consent entry missing its required `facetRef` (Section 1.4.4).
- `signature-keyref-without-embedded-key.jsonld` - `keyRef` resolution unavailable by construction and no `publicKeySpkiB64` embedded; the manifest cannot be verified (Section 1.6.4 key-resolution ladder).

The TypeScript command adapter executes the new signed corpus under `conformance/v0.5/expected.json`. Valid Tier-2/3-shaped objects are only structure controls: they do not establish aggregate verification, attester authorization, holder identity or higher-tier authority.

5.3 Schema-detectable vs evaluator-only rejections

The direct JSON Schema contract is intentionally structural. `conformance/v0.5/expected.json` is the normative fixture matrix for schema sweep behavior:

- Schema-detectable rejects: `invalid/holder-binding-invalid-mode.jsonld`, `invalid/holder-binding-missing-cnf.jsonld`, `invalid/binding-proof-invalid-type.jsonld`, `invalid/missing-signature.jsonld`, `invalid/required-liveness-bad-minfreshness.jsonld`, `invalid/required-liveness-not-object.jsonld`, `invalid/facet-assurance-class-bad-enum-shape.jsonld`, `invalid/facet-assurance-

class-not-string-shape.jsonld, invalid/manifest-version-mismatch.jsonld, invalid/unsupported-signature-profile.jsonld, invalid/type-missing-um-manifest.jsonld, invalid/context-wrong-namespace.jsonld, invalid/holder-binding-reciprocal-missing-fields.jsonld, invalid/holder-binding-bbs-missing-commitment.jsonld, invalid/liveness-missing-valid-until.jsonld, invalid/encrypted-facet-malformed-jwe.jsonld, invalid/v03-backcompat-tier-floor-unsatisfiable.jsonld, invalid/missing-subject.jsonld, invalid/missing-expires-at.jsonld, and invalid/consent-missing-facet-ref.jsonld.

- Evaluator-only rejects (the schema accepts these; only the evaluator catches them — crypto verification, temporal math, challenge matching, tier logic, and resource limits are not schema-expressible): invalid/ceremony-proof-insufficient-attesters.jsonld, invalid/claim-tier-below-manifest-floor.jsonld, valid/manifest-locked-facet-below-floor.jsonld, invalid/bad-signature-value.jsonld, invalid/tampered-after-signing.jsonld, invalid/key-substitution-mismatch.jsonld, invalid/expired-manifest.jsonld, invalid/issued-after-expires.jsonld, invalid/future-issued-at.jsonld, invalid/presentation-proof-challenge-mismatch.jsonld, invalid/presentation-proof-audience-mismatch.jsonld, invalid/presentation-proof-missing-when-challenged.jsonld, invalid/agent-delegation-delegatedby-mismatch.jsonld, invalid/agent-delegation-expired.jsonld, invalid/facet-tier-below-manifest-floor.jsonld, invalid/claim-proof-oversize-vp.jsonld, invalid/signature-keyref-without-embedded-key.jsonld, invalid/resource-limit-depth-11.jsonld, invalid/resource-limit-array-1001.jsonld, and invalid/resource-limit-size-1000001.jsonld.
- I-JSON ingress rejects (refused before schema validation, because the input is not I-JSON): invalid/duplicate-member.jsonld, invalid/escaped-duplicate-member.jsonld, invalid/malformed-json.jsonld, invalid/ijson-lone-surrogate-in-signature.jsonld, and invalid/ijson-number-overflow-in-signature.jsonld.
- One inversion runs the other way: valid/v03-backcompat-manifest.jsonld fails the v0.5 structural schema (it is a v0.3 envelope) but is accepted by the evaluator through the EXT-T1 Section T1.6.4 backwards-compatibility path; its expected.json entry records both expectations.

Schema-only implementers MUST NOT claim full evaluator conformance from a schema sweep alone. The dated TypeScript evidence separately exercised structural/I-JSON ingress and focused API behavior. Those development commands and adapter dependencies are not part of the public specification package; external implementations consume the exact versioned fixture index and portable contract. A green finite fixture report is not full normative qualification and yields level none for the candidate report described here.

5.4 New v0.5 boundary controls

These finite controls distinguish point/scalar acceptance, P15 structure/threshold limits, honest proof states and raw I-JSON ingress. They add no aggregate-verification or identity-authority claim. Their independent per-case outcomes and schema classifications are in the index.

- `invalid/ed25519-identity-a-r.jsonld`
- `invalid/ed25519-noncanonical-a.jsonld`
- `invalid/ed25519-noncanonical-r.jsonld`
- `invalid/ed25519-signbit-zero-x-a.jsonld`
- `invalid/ed25519-signbit-zero-x-r.jsonld`
- `invalid/ed25519-s-plus-order.jsonld`
- `invalid/ed25519-s-order.jsonld`
- `invalid/ed25519-mixed-order-a.jsonld`
- `invalid/ed25519-mixed-order-r.jsonld`
- `valid/ed25519-ordinary-v05.jsonld`
- `invalid/ceremony-zero-m.jsonld`
- `invalid/ceremony-zero-n.jsonld`
- `invalid/ceremony-m-above-n.jsonld`
- `invalid/ceremony-more-attesters-than-n.jsonld`
- `invalid/ceremony-duplicate-attesters.jsonld`
- `valid/ceremony-three-dids-shape-only.jsonld`
- `invalid/ceremony-shape-not-aggregate-verification.jsonld`
- `invalid/binding-proof-three-dids.jsonld`
- `valid/present-proof-not-evaluated.jsonld`
- `valid/absent-proof-offline.jsonld`
- `invalid/present-proof-unimplemented-interactive.jsonld`
- `invalid/did-auth-invalid-point-signature.jsonld`
- `invalid/duplicate-member.jsonld`
- `invalid/escaped-duplicate-member.jsonld`
- `invalid/malformed-json.jsonld`

- `valid/escaped-member-without-duplicate.jsonld`

5.5 Further indexed controls

These signed controls are in the candidate index, and the index holds each one's exact expectations. The text below is the index's own reason for each.

- `valid/purpose-request-matched.jsonld` - Explicit caller purpose matches the signed grant.
- `valid/purpose-request-mismatched-withheld.jsonld` - No disclosure: caller purpose is different, omitted or empty; the envelope remains accepted-partial.
- `valid/purpose-request-omitted-withheld.jsonld` - No disclosure: caller purpose is different, omitted or empty; the envelope remains accepted-partial.
- `valid/purpose-request-empty-withheld.jsonld` - No disclosure: caller purpose is different, omitted or empty; the envelope remains accepted-partial.
- `invalid/consent-purpose-missing.jsonld` - Malformed required consent purpose is invalid, never an unrestricted grant.
- `invalid/consent-purpose-nonstring.jsonld` - Malformed required consent purpose is invalid, never an unrestricted grant.
- `invalid/consent-purpose-empty.jsonld` - Malformed required consent purpose is invalid, never an unrestricted grant.
- `valid/encrypted-entity-rust-producer.jsonld` - Fresh signed envelope over exact independently encrypted JSON Entities; the TypeScript evaluator records both facets opaque without decrypting.
- `valid/encrypted-entity-jose-producer.jsonld` - Fresh signed envelope over exact independently encrypted JSON Entities; the TypeScript evaluator records both facets opaque without decrypting.
- `valid/consent-boundary-before-grant.jsonld` - Signed consent boundary before-grant: withhold the governed facet; condition-violated.
- `valid/consent-boundary-at-grant.jsonld` - Signed consent boundary at-grant: explicit active authorization.
- `valid/consent-boundary-before-expiry.jsonld` - Signed consent boundary before-expiry: explicit active authorization.

- `valid/consent-boundary-at-expiry.jsonld` - Signed consent boundary at-expiry: withhold the governed facet; expired.
- `valid/consent-boundary-after-expiry.jsonld` - Signed consent boundary after-expiry: withhold the governed facet; expired.
- `valid/consent-boundary-scope-omitted.jsonld` - Signed consent boundary scope-omitted: withhold the governed facet; scope-mismatch.
- `valid/consent-boundary-scope-empty.jsonld` - Signed consent boundary scope-empty: withhold the governed facet; scope-mismatch.
- `valid/consent-boundary-scope-wrong.jsonld` - Signed consent boundary scope-wrong: withhold the governed facet; scope-mismatch.
- `valid/consent-boundary-scope-matching.jsonld` - Signed consent boundary scope-matching: explicit active authorization.
- `valid/consent-boundary-multiple-operations-covered.jsonld` - Signed consent boundary multiple-operations-covered: explicit active authorization.
- `valid/consent-boundary-multiple-operations-partly-uncovered.jsonld` - Signed consent boundary multiple-operations-partly-uncovered: withhold the governed facet; scope-mismatch.
- `valid/consent-boundary-scope-null.jsonld` - Signed consent boundary scope-null: withhold the governed facet; scope-mismatch.
- `valid/consent-boundary-scope-string.jsonld` - Signed consent boundary scope-string: withhold the governed facet; scope-mismatch.
- `valid/consent-boundary-scope-object.jsonld` - Signed consent boundary scope-object: withhold the governed facet; scope-mismatch.
- `valid/consent-boundary-scope-nonstring-member.jsonld` - Signed consent boundary scope-nonstring-member: withhold the governed facet; scope-mismatch.
- `valid/consent-boundary-scope-empty-member.jsonld` - Signed consent boundary scope-empty-member: withhold the governed facet; scope-mismatch.
- `valid/consent-boundary-withdrawn-precedence.jsonld` - Signed consent boundary withdrawn-precedence: withhold the governed facet; withdrawn.
- `valid/consent-boundary-expired-precedence.jsonld` - Signed consent boundary expired-precedence: withhold the governed facet; expired.
- `valid/consent-boundary-not-yet-granted-precedence.jsonld` - Signed consent boundary not-yet-granted-precedence: withhold the governed facet; condition-violated.

- `valid/consent-boundary-unknown-condition-precedence.jsonld` - Signed consent boundary unknown-condition-precedence: withhold the governed facet; condition-violated.
- `valid/consent-boundary-purpose-matching.jsonld` - Signed consent boundary purpose-matching: explicit active authorization.
- `valid/consent-boundary-purpose-mismatching.jsonld` - Signed consent boundary purpose-mismatching: withhold the governed facet; purpose-mismatch.
- `valid/facet-gate-missing-consent-before-assurance.jsonld` - Explicit signed facet-gate-missing-consent-before-assurance oracle: accepted-partial; configured capability never establishes achieved identity trust.
- `valid/facet-gate-withdrawn-consent-before-assurance.jsonld` - Explicit signed facet-gate-withdrawn-consent-before-assurance oracle: accepted-partial; configured capability never establishes achieved identity trust.
- `valid/facet-gate-scope-mismatch-before-assurance.jsonld` - Explicit signed facet-gate-scope-mismatch-before-assurance oracle: accepted-partial; configured capability never establishes achieved identity trust.
- `valid/facet-gate-valid-consent-unmet-assurance.jsonld` - Explicit signed facet-gate-valid-consent-unmet-assurance oracle: rejected; configured capability never establishes achieved identity trust.
- `valid/facet-gate-valid-consent-met-assurance.jsonld` - Explicit signed facet-gate-valid-consent-met-assurance oracle: accepted; configured capability never establishes achieved identity trust.
- `valid/facet-gate-sealed-before-assurance.jsonld` - Explicit signed facet-gate-sealed-before-assurance oracle: accepted-partial; configured capability never establishes achieved identity trust.
- `valid/facet-gate-sealed-without-floor.jsonld` - Explicit signed facet-gate-sealed-without-floor oracle: accepted-partial; configured capability never establishes achieved identity trust.
- `valid/facet-gate-missing-consent-before-liveness.jsonld` - Explicit signed facet-gate-missing-consent-before-liveness oracle: accepted-partial; configured capability never establishes achieved identity trust.
- `valid/facet-gate-valid-consent-unmet-liveness.jsonld` - Explicit signed facet-gate-valid-consent-unmet-liveness oracle: rejected; configured capability never establishes achieved identity trust.
- `valid/trust-floor-manifest-0.jsonld` - Explicit signed trust-floor-manifest-0 oracle: accepted; configured capability never establishes achieved identity trust.

- `valid/trust-floor-manifest-1.jsonld` - Explicit signed trust-floor-manifest-1 oracle: rejected; configured capability never establishes achieved identity trust.
- `valid/trust-floor-manifest-2.jsonld` - Explicit signed trust-floor-manifest-2 oracle: rejected; configured capability never establishes achieved identity trust.
- `valid/trust-floor-manifest-3.jsonld` - Explicit signed trust-floor-manifest-3 oracle: rejected; configured capability never establishes achieved identity trust.
- `valid/trust-floor-facet-0.jsonld` - Explicit signed trust-floor-facet-0 oracle: accepted; configured capability never establishes achieved identity trust.
- `valid/trust-floor-facet-1.jsonld` - Explicit signed trust-floor-facet-1 oracle: accepted-partial; configured capability never establishes achieved identity trust.
- `valid/trust-floor-facet-2.jsonld` - Explicit signed trust-floor-facet-2 oracle: accepted-partial; configured capability never establishes achieved identity trust.
- `valid/trust-floor-facet-3.jsonld` - Explicit signed trust-floor-facet-3 oracle: accepted-partial; configured capability never establishes achieved identity trust.
- `valid/trust-floor-mixed-1.jsonld` - Explicit signed trust-floor-mixed-1 oracle: accepted-partial; configured capability never establishes achieved identity trust.
- `valid/trust-floor-claim-1.jsonld` - Explicit signed trust-floor-claim-1 oracle: accepted-partial; configured capability never establishes achieved identity trust.
- `valid/trust-floor-mixed-2.jsonld` - Explicit signed trust-floor-mixed-2 oracle: accepted-partial; configured capability never establishes achieved identity trust.
- `valid/trust-floor-claim-2.jsonld` - Explicit signed trust-floor-claim-2 oracle: accepted-partial; configured capability never establishes achieved identity trust.
- `valid/trust-floor-mixed-3.jsonld` - Explicit signed trust-floor-mixed-3 oracle: accepted-partial; configured capability never establishes achieved identity trust.
- `valid/trust-floor-claim-3.jsonld` - Explicit signed trust-floor-claim-3 oracle: accepted-partial; configured capability never establishes achieved identity trust.
- `valid/trust-floor-malformed-ceiling-string.jsonld` - Explicit signed trust-floor-malformed-ceiling-string oracle: rejected; configured capability never establishes achieved identity trust.
- `valid/trust-floor-malformed-ceiling-null.jsonld` - Explicit signed trust-floor-malformed-ceiling-null oracle: rejected; configured capability never establishes achieved identity trust.
- `valid/trust-floor-malformed-ceiling-too-high.jsonld` - Explicit signed trust-floor-malformed-ceiling-too-high oracle: rejected; configured capability never establishes achieved

identity trust.

- `valid/trust-floor-malformed-ceiling-negative.jsonld` - Explicit signed trust-floor-malformed-ceiling-negative oracle: rejected; configured capability never establishes achieved identity trust.
- `valid/trust-floor-malformed-ceiling-fractional.jsonld` - Explicit signed trust-floor-malformed-ceiling-fractional oracle: rejected; configured capability never establishes achieved identity trust.
- `valid/context-integrity-root-pinned-scalar.jsonld` - Accept only pinned interpreted scopes; literal/unknown JSON is preserved without context authority.
- `valid/context-integrity-root-pinned-array.jsonld` - Accept only pinned interpreted scopes; literal/unknown JSON is preserved without context authority.
- `valid/context-integrity-root-pinned-repeated-array.jsonld` - Accept only pinned interpreted scopes; literal/unknown JSON is preserved without context authority.
- `invalid/context-integrity-root-empty-array.jsonld` - Reject context substitution before interpreting UM fields; the independently valid outer signature is not executed by evaluation.
- `invalid/context-integrity-root-reset-after-pinned.jsonld` - Reject context substitution before interpreting UM fields; the independently valid outer signature is not executed by evaluation.
- `invalid/context-integrity-root-prefix-substitution.jsonld` - Reject context substitution before interpreting UM fields; the independently valid outer signature is not executed by evaluation.
- `invalid/context-integrity-root-remote-after-pinned.jsonld` - Reject context substitution before interpreting UM fields; the independently valid outer signature is not executed by evaluation.
- `valid/context-integrity-facet-pinned-scalar.jsonld` - Accept only pinned interpreted scopes; literal/unknown JSON is preserved without context authority.
- `valid/context-integrity-facet-pinned-array.jsonld` - Accept only pinned interpreted scopes; literal/unknown JSON is preserved without context authority.
- `invalid/context-integrity-facet-reset.jsonld` - Reject context substitution before interpreting UM fields; the independently valid outer signature is not executed by evaluation.
- `invalid/context-integrity-facet-prefix-substitution.jsonld` - Reject context substitution before interpreting UM fields; the independently valid outer signature is not executed by evaluation.

- `invalid/context-integrity-facet-remote.jsonld` - Reject context substitution before interpreting UM fields; the independently valid outer signature is not executed by evaluation.
- `valid/context-integrity-claim-pinned-scalar.jsonld` - Accept only pinned interpreted scopes; literal/unknown JSON is preserved without context authority.
- `valid/context-integrity-claim-pinned-array.jsonld` - Accept only pinned interpreted scopes; literal/unknown JSON is preserved without context authority.
- `invalid/context-integrity-claim-reset.jsonld` - Reject context substitution before interpreting UM fields; the independently valid outer signature is not executed by evaluation.
- `invalid/context-integrity-claim-prefix-substitution.jsonld` - Reject context substitution before interpreting UM fields; the independently valid outer signature is not executed by evaluation.
- `invalid/context-integrity-claim-remote.jsonld` - Reject context substitution before interpreting UM fields; the independently valid outer signature is not executed by evaluation.
- `valid/context-integrity-consent-pinned-scalar.jsonld` - Accept only pinned interpreted scopes; literal/unknown JSON is preserved without context authority.
- `valid/context-integrity-consent-pinned-array.jsonld` - Accept only pinned interpreted scopes; literal/unknown JSON is preserved without context authority.
- `invalid/context-integrity-consent-reset.jsonld` - Reject context substitution before interpreting UM fields; the independently valid outer signature is not executed by evaluation.
- `invalid/context-integrity-consent-prefix-substitution.jsonld` - Reject context substitution before interpreting UM fields; the independently valid outer signature is not executed by evaluation.
- `invalid/context-integrity-consent-remote.jsonld` - Reject context substitution before interpreting UM fields; the independently valid outer signature is not executed by evaluation.
- `valid/context-integrity-literal-entity-preserved.jsonld` - Accept only pinned interpreted scopes; literal/unknown JSON is preserved without context authority.
- `valid/context-integrity-unknown-member-preserved.jsonld` - Accept only pinned interpreted scopes; literal/unknown JSON is preserved without context authority.
- `valid/resource-limit-depth-10.jsonld` - Nesting of exactly 10 levels, counting containers with the manifest object as level 1, is within the Base Section 6.3 default.
- `invalid/resource-limit-depth-11.jsonld` - Validly signed nesting of 11 levels exceeds the Base Section 6.3 default and refuses at Arrive, before the signature check.
- `valid/resource-limit-array-1000.jsonld` - An array of exactly 1,000 entries is within the Base Section 6.3 default.

- `invalid/resource-limit-array-1001.jsonld` - A validly signed array of 1,001 entries exceeds the Base Section 6.3 default and refuses at Arrive, before the signature check.
- `valid/resource-limit-size-1000000.jsonld` - Exactly 1,000,000 octets over the JCS-canonical form is within the Base Section 6.3 default.
- `invalid/resource-limit-size-1000001.jsonld` - A validly signed 1,000,001 octets over the JCS-canonical form exceeds the Base Section 6.3 default and refuses at Arrive, before the signature check.
- `valid/consent-two-entries-both-satisfied.jsonld` - Two consent entries on one facet, both satisfied: the facet is processed.
- `valid/consent-two-entries-one-purpose-mismatch.jsonld` - Two consent entries on one facet, one satisfied and one with a purpose mismatch: the facet is consent-denied.
- `invalid/ijson-lone-surrogate-in-signature.jsonld` - A validly signed manifest with a lone surrogate inside `signature`, outside the signing input, is not I-JSON and refuses at ingress.
- `invalid/ijson-number-overflow-in-signature.jsonld` - A validly signed manifest with a number outside the IEEE 754 double range inside `signature`, outside the signing input, is not I-JSON and refuses at ingress.

6) Adversarial and misuse expectations

v0.5 conformance verification MUST include rejection tests for the v0.2/v0.3 baseline cases (signature spoofing, signature misuse, temporal misuse, manifest-version mismatch) plus v0.5-specific cases:

1. **Holder-binding bypass** - a claim relied upon at Tier 1+ with no holder binding, or with an invalid `mode`, MUST NOT be treated as Tier 1; it caps at Tier 0. See EXT-T1.
2. **Missing binding fields** - `sd-jwt-kb` without `cnfThumbprint`, or `reciprocal-control` without `boundDid` / `subjectProof` / `boundDidProof`, MUST be rejected as a binding failure.
3. **Trust-tier downgrade attempts** - claim or facet floors that attempt to specify a lower `requiredTrustTier` than the manifest-level floor MUST be rejected (the floor raises only, Section 6.4.5).
4. **Malformed binding/ceremony proofs** - a `bindingProof` with a `type` outside the defined set, or a `ceremonyProof` whose `attesters` count is below the `threshold`, MUST be rejected.
5. **Stale liveness replay** - a `livenessAttestation` past its `validUntil`, or an interactive presentation whose `presentationProof` challenge/audience does not match the verifier's,

MUST NOT be accepted as current human presence.

6. **Malformed JWE encrypted facets** - incomplete JWE structures that an evaluator would otherwise sealed-entry through. Structural conformance MUST require all five JWE fields (`protected`, `recipients`, `iv`, `ciphertext`, `tag`).
7. **Context substitution** - a manifest whose context document fails its content-hash check MUST NOT have its term definitions trusted (Section 6.10).
8. **Agent delegation with delegator/subject mismatch** - `um:agentDelegation` with `delegatedBy` not matching the manifest `subject`, and **expired delegation** (`expiresAt` in the past), MUST be rejected. See Section 6.5.

7) Standalone conformance suite (external adopter path)

For implementation-neutral, external conformance execution, use the standalone suite documentation:

- Suite package root (repo path): `conformance/README.md`
- v0.5 fixture root: `conformance/v0.5/`
- Expected outcomes: `conformance/v0.5/expected.json`

The standalone suite is language-neutral and encoding-neutral: it packages fixtures plus expected outcomes so implementations can prove equivalent v0.5 verification behaviour across runtimes and across conformant production rules.

For a purpose-bearing evaluation case, the harness MUST pass the case's explicit caller purpose from `evaluationContext.intendedPurpose` when supplied. It MUST preserve omitted or empty caller-purpose controls as such and MUST NOT derive the caller's purpose from `consents[].purpose` or other grant data. Fixture context is test input outside the signed manifest; these instructions introduce no request carrier or manifest member.

Only entries in the index's `fixtures` array define the ordinary finite suite. Supplemental `jweInteroperabilityVectors`, when present at the index root, is separate test evidence: honor its public-test-only key warning, consume and report its specified decryption controls separately, and never use disposable private test keys for real data or put them into manifest wire fields. TypeScript sealed-entry assertions do not prove that decryption ran; supplemental results do not establish full conformance.

8) Reference implementation evidence (non-normative)

The separately maintained TypeScript and Rust reference evaluators are two possible implementations. This public specification package excludes their development runners. The dated evidence below has the limits described in Section 5; it does not establish full conformance.

Offline signing-key resolution (2026-10-26). Both v0.5 evaluators offer optional offline `did:key` resolution for Ed25519 and P-256. The `offlineKeyResolution` context option requires a boolean; absent or `false` preserves existing behavior. Rust's typed context names it `offline_key_resolution`. TypeScript rejects `true` combined with a `resolveKeyRef` callback; that application hook remains available when offline mode is absent or `false`. Legacy v0.4 and standalone assertion helpers retain their defaults.

The bounded resolver accepts canonical base58btc (`z`) or unpadded base64url (`u`), canonical multicodec varints and valid key points. It selects the bare DID or its exact `#<same-multibase>` signing method. Queries, paths, foreign fragments, unsupported curves and malformed encodings do not resolve.

A resolved key can verify without an embedded key. When both are present, their decoded SPKI bytes must match exactly. The evaluator refuses substitution.

Failed or unavailable resolution may fall back to an embedded key and reports `unresolved`; without an embedded key, verification refuses. If no lookup occurs, the receipt reports `not-attempted`. Resolution makes no network request. Declared-suite and signature checks still apply, and root `did-auth` remains Ed25519-only.

SD-JWT holder binding (2026-10-26). The reference helpers perform separate, bounded issuer-signature and holder-binding mathematics using caller-authenticated issuer and holder role keys. They check the binding's signature, audience, challenge, time and presentation hash.

These checks do not establish complete subject or issuer authorization, consent, revocation status or claim identity. Root `sd-jwt-kb` remains a refused PREVIEW presentation-proof mode. Neither key resolution nor holder binding elevates claim or effective identity trust above Tier 0. The implementations provide no aggregate or Tier-2/3 verifier. All 43 obligations remain applicable; full conformance remains unevaluated.

Dated qualification (2026-10-26). The original TypeScript holder-binding acceptance covered 18 controls. Rust acceptance covered 43 controls, including 23 inputs shared with TypeScript. Subsequent resolver acceptance covered 108 fresh signed inputs in each runtime, including four SD-JWT composition cases.

Rust 1.85.0 separately built the unchanged locked `um-core` library on macOS arm64. The two focused resolver and holder-binding targets passed 12 tests. Two shared-input tests remained ignored. That compiler result covers only this library build and those targets. It does not qualify other platforms, packages or consumers. The historical corpus, ABI and bilateral results below remain bound to their original dates and artifact hashes; they are not fresh executions against later binaries.

Rust evaluator phase one (2026-10-21). Explicit APIs include `assert_structural_v05`, `verify_manifest_v05`, `assert_manifest_v05_fresh`, `evaluate_manifest_v05`, `crypto::verify_manifest_signature_v05` and `EvaluationContext::from_json_v05`. The existing versioned generator supplies additive `model_v05`, `schema_v05` and `pins::v05` from the frozen schema/context; unqualified v0.4 models, pins and defaults remain. Shared parsing, JCS, profile selection and evaluator stages require a finite caller-pinned clock, check the actual pinned offline context before cryptography, use strict Profile A verification, retain Profile B raw-64/high-S behavior, enforce explicit purpose and fully covered operation scope, and order sealed entry → consent → floors. Achieved trust remains capped at Tier 0; unsupported proof/authority paths fail closed or retain honest unevaluated states. Sealed facets are not decrypted by the evaluator.

The phase-one command adapter matched **188/188 unchanged indexed cases**, including **119 cases with strict receipt assertions**; its server passed **25 HTTP controls**, with **12 independent command controls**. The phase-one suites passed **420 core, 22 adapter and 7 generator tests**. Those results identify the phase-one adapter/server below; they are not a complete-corpus rerun against later binaries.

Local bilateral execution (2026-10-21). Selected version `0.5` reaches both counterparty evaluations and the existing durable session contract. The explicit receipt/challenge helpers use the same v0.5 strict wire predicate for action, challenge and receipt boundaries. Two persistent Swift callers commit all five operations in each positive encounter, verify actual evaluator and receipt digests and reciprocal acknowledgments, and refuse version changes after restart. Channels are locally injected evidence, not authenticated peer identity or physical radio observations. Independently installed role keys authenticate the tested signed bodies only within that local test contract.

The preserved bilateral predecessor passed the **52-control online caller suite for each of v0.4 and v0.5**, including eight completed encounters per version. The observation union covers 78 v0.5 cases per command/HTTP transport and 76 v0.4 cases per transport; retained failed runs received only named version-specific expectation corrections and focused failed-subcase reruns, without changing evaluator semantics or weakening expected behavior. **422 core and 22 adapter tests** passed before the final narrow implicit-v0.4 fallback correction. After that correction, one focused adapter test, two old/new byte-equal legacy comparisons and four byte-equal witness comparisons

passed. The final successor then passed all eight positive v0.5 encounters, one v0.4 encounter and eight actual HTTP responses. Fresh independent command/HTTP controls cover changed-version restart, opposite-version manifests and strict action/challenge/receipt boundaries. These focused successor results do not transfer the predecessor's complete fault matrix to another binary.

Independent causal inspection of four retained caller runs plus a fresh final-binary run traced **270 action/delivery/persistence links and 54 verified signed receipts across 27 completed encounters**. Both evaluator digests match the receipt bodies, whose signatures verify with independently installed role keys. The fresh run adds two mixed-suite positives and a signed-body tamper refusal. The totals describe distinct scoped observations, not a single new matrix or a certification grade.

Executed binary scope	**SHA-256**
Phase-one adapter: complete 188-case corpus	`3abdbac5be2851f8c1b84ca0c7afb7e7226938cfbe8f55c7bea8b77a744cbd18`
Phase-one server: 25 HTTP controls	`8e79763fad2fedae3203d0957417cb87e388dfadcd60c716f5ae5b8add5e5e0e`
Bilateral predecessor adapter: full online suite	`f5d29ef0a93719c9934c699d595d7b1ae12dffadbdeefab46b63fd17c1930a3f`
Bilateral predecessor server	`0052b4c54ae0d24ca19a39c2624f55334b45856523ac100edd26b48bb61a8a84`
Final successor adapter: focused compatibility/caller controls	`00545a2347c7f8ffb3f6a9c7499813005c97ac4798d11dcd8015ef5b31b7fe34`
Final successor server: eight HTTP responses and focused controls	`24c9a38c2a1dd668882838b565da5027c5071c7ac54182a71511d4ee7ff4f847`
Swift caller and identical actor executable	`67e1b8c11589bb7345b1aafa96174df005b1fb1b8194b690d9dcb33dc2c42849`

Executed binary scope	SHA-256
Unchanged FFI signing/prerequisite library	`22c302bda3f544533fb19036b231c328ab0c02d55db302b61ed409ddca4fbb8d`

These are implementation-evidence identities, not wire pins or public package dependencies. The old FFI signing library is not a newly qualified v0.5 ABI evaluator. Local software execution does not establish holder generation, peer identity, real radio, source permission or withdrawal, authenticated services, protected adoption, physical/device behavior, whole-SDK parity or complete Standard conformance. All 43 obligations and unmet baselines remain in force; conformance stays `none`. The separately dated TypeScript 188/192/38, 46 I-JSON, 937-signature and 16-JWE results retain their own scopes and were not rerun for this documentation refresh.

Third-party adopters do not need Node/TypeScript; the fixtures are designed to be portable. You can build a conformant harness in any language and over any conformant encoding.

9) Notes on PREVIEW features

Several v0.5 capabilities are built into the draft on the editors' recommended defaults and are marked **PREVIEW** - under active review and revisable before the schema locks. The Base defines the candidate Tier-0 / Tier-1 surface; this source change does not establish new-version runtime interoperability or completion of its requirements. PREVIEW comes in two grades (the "two-grade PREVIEW policy" cited from Base Section 4.4): **Grade-1 (structural) PREVIEW** - members already on the wire (context term + shape fixtures) whose enforcement semantics remain revisable. **Grade-2 (design-stage) PREVIEW** - candidate members not yet in the schema/context/fixtures at all. PREVIEW surfaces include:

- **Tier-2 ZKP binding profiles** (EXT-T2) and **Tier-3 ceremony** (EXT-T3) - the proof objects are structurally defined; the verification profiles are PREVIEW.
- **Presentation-proof mechanisms `sd-jwt-kb` and `bbs-derived`** (EXT-T1 Section T1.2) - **at-risk / specification-only.** Of the three `presentationProof.proofType` values, only `did-auth` was exercised by the earlier v0.4 fixture suite and the reference implementation; `sd-jwt-kb` (whose manifest binding depends on the unratified decision UM-v04-DEC-07) and `bbs-derived` have no fixtures and no implementation, so interoperability for the two is not yet demonstrable. An implementation MUST fail them closed rather than report an unverified proof as verified, and a conformance claim relying on interactive presentation SHOULD state which

`proofType` it implements. Both stay at risk in v0.5: UM-v04-DEC-07 is not ratified, and its ratification and the fixtures are later-version work.

- **Per-facet liveness and assurance floors**, **locked-tier portable unlock**, **per-facet cryptographic isolation**, **write authorization**, and **key-lifecycle receipt events** are present in the source, schema/context where structural, and fixtures where currently covered; enforcement is PREVIEW where marked by the source profiles. **Facet sensitivity ordering** is profile-supplied: the label ordering that gates derived writes comes from a registered profile the deployment pins, not from the manifest under evaluation; the derived-write join rule (Base Section 7.2) stays normative. The transitional manifest-carried ordering member was removed from the wire on 2026-07-15 at the v0.4 schema lock — it is absent from the context, the fixture set, and the reference behavioral harness, which now sources the q3 test ordering from harness-level profile configuration, exactly as a conforming evaluator must.
- **Receipt as a first-class manifest class** with hash-chaining, the **bilateral session model**, **federation** and manifest forwarding, **profile registration**, the **agent-delegation scope registry**, **category trust / trustWeight**, and the **extended structural schemas** (EXT-OPT) - claimed as optional features, PREVIEW per part. **`trustWeight` (EXT-OPT Section O8) is specification-only in v0.5:** its term was removed from the frozen context at the 2026-07-15 schema lock because no fixture, example, or worked example exercises it and its value shape (closed enum versus extensible vocabulary) is an open question. The O8 semantics stand as specified — including that an evaluator encountering the deprecated `interpretedAs` MUST ignore it and default to `"hint"` - but the member has no v0.5 context term and so is not carried losslessly on the v0.5 wire; it re-enters the context with fixture and schema coverage once the value-shape question is settled (Base Appendix B.1).
- **Post-quantum signatures** (Base Section 6.7) - named as a migration path; PREVIEW.
- **Design-stage candidate profiles** (EXT-OPT Sections O2.6, O12, O13, O14): **current policy-state resolution**, **asset-bound represented-person consent**, **signed referral-route attribution**, and **GeoPose primary pose values** - PREVIEW at the design stage. Their candidate members and value tokens are not yet in the published schema/context or the fixture suite, and they carry no conformance tests until their designs are finalized (Section 4.4 discipline). A conformance claim MUST NOT include them yet; for evaluators that do not implement them, the existing fail-closed baselines apply (unrecognized claim types unprocessable above Tier 0, unrecognized pointer types recorded but not acted on, unrecognized or unenforceable consent conditions fail closed, and missing or unverifiable GeoPose data is not inferred).

One boundary note that is **not** itself a PREVIEW surface: **selective minimum disclosure** is production-candidate Base behavior (Sections 1 and 2 above), claimed only in its bounded v0.5 form — projection-scoped disclosure, consent gating, sealed-entry handling, and receipt evidence.

What remains open is the deeper high-privacy per-target chain: unlinkability, full or partial, cover-slot indistinguishability, resolver-operator privacy, and residual-fingerprint measurement. Do not infer those guarantees from a Base conformance claim.

A v0.5 conformance claim shows the mechanism, with reference suppliers where a supplier is needed. It shows no real-world identity, issuer or revocation result. The release boundary states the limit for first contact, issuer authority and status.

A second boundary note of the same kind: **carrying an external-standard artifact** (for example C2PA content-credential material, an OGC GeoPose payload, or a W3C Verifiable Credential) inside or by reference from a manifest does not extend a v0.5 conformance claim to that artifact. The carried standard verifies under its own rules, independently of the UM envelope evaluation, and a v0.5 receipt records the UM envelope outcome — not the carried standard's verdict. Absent a registered carry profile, an evaluator that cannot evaluate a carried standard preserves it (sealed/opaque or unprocessed, per the fail-closed baselines above) and does not report it as verified. A dedicated carried-standard verification-result field and an unknown-carried-standard disposition value are future registered-profile work, not Base members.

A conformance claim SHOULD state which PREVIEW features, if any, it implements, since their wire shape may still change.

10) Implementation onboarding package (non-normative)

- Read the Primer, Base, and Cookbook, then the profiles applicable to the implementation claim.
- Consume the finite index using the public fixture guide, selecting the immutable revision associated with the specification version when released.
- Document implemented and unmet obligations using EXT-OPT O11.1; publish exact executed evidence and limits.