# Universal Manifest v0.5 — Overview

## Scope and versioning

**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/readme/

**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 Spec (v0.5)

> **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.


Copyright (c) 2026 SumSet Tech LLC and the Universal Manifest contributors. Grig Bilham — inventor, architect, and author of the Universal Manifest. This specification is available under the Apache License, Version 2.0. Earlier published revisions retain their original licence notices. Earlier versions were reviewed in the OMA3 Technical Working Group, and publication asserts no OMA3 ratification. This specification does not settle standards IPR or patent adoption.

> Universal Manifest is an open specification for portable state manifests. A manifest is not static metadata: v0.5 pairs the manifest with a six-stage evaluation that checks signature, freshness, projection, and consent, then emits a structured receipt of what was actually verified. Current permission additionally requires the authority and target checks described in the Base; it is not implied by an envelope signature. The specification is independent of programming language, framework, or runtime: an abstract data model is written to bytes through a production rule. JSON-LD is the reference encoding; CBOR-LD is a separate profile. The public specification package supplies the contract and fixtures; separately maintained reference implementations provide bounded evidence, not a required implementation dependency.

v0.5.0 is a new-version contract derived from the accepted v0.4 editorial set, published as tag `v0.5.0` with the same wire files as release candidate 1 (`v0.5.0-rc.1`). The immutable revision of UniversalManifest/universal-manifest identified by a specification release is normative; main is the working copy. This complete first-class source set lives in `spec/v0.5/`. The `/spec/v0.5/` website is its generated reading view. `/spec/v0.5/` is always this version's reading view; the maintainers set where `/spec/latest/` points.

The v0.5 document set is organized as a **conformance-tier stack**. The question that decides what you read and build is "what level of assurance am I certifying to?" A breadth-first Primer orients you; the Base specifies everything required for a Tier-0 / Tier-1 evaluator and holder; four companion profiles add depth by tier or by optional feature; and the Cookbook collects worked examples.

## The v0.5 document set

| Document | Role | Status |
|---|---|---|
| Primer | Breadth-first overview: what UM is, the shape of a manifest, the lifecycle, the core guarantees, a one-glance map of tiers and extensions. No normative requirements. | Informative |
| Base (Tier-0 / Tier-1 Core) | Everything required to build a conformant Tier-0 / Tier-1 evaluator and holder. Names advanced and optional features and points to the extensions for their mechanics. | Production-candidate (some members PREVIEW) |
| EXT-T1 (Tier-1 Binding Profile) | Tier-1 binding mechanics: holder binding, presentation proof, liveness attestation, the Stage-2 sub-steps, the end-to-end claim-proof path, v0.3 backwards compatibility. | Normative (Sections |

| Document | Role | Status |
|---|---|---|
| | | T1.3.1–T1.3.3 PREVIEW) |
| EXT-T2 (Tier-2 Cryptographic-Binding Profile) | Tier-2 ZKP proof profiles, the BBS+ selective-disclosure track, the post-quantum migration path. | PREVIEW |
| EXT-T3 (Tier-3 Ceremony Profile) | Tier-3 multi-party ceremony: model, signer-role taxonomy, threshold-protocol guidance. | PREVIEW |
| EXT-OPT (Optional-Feature Profiles) | Tier-orthogonal optional features: the abstract data model and its CBOR-LD encoding, the statusRef resolution protocol, receipts-as-a-class with hash-chaining, the bilateral session model, federation, profile registration, the agent-delegation scope registry, trustWeight and category trust, extended schemas, the cryptographic-requirements summary, and the design-stage candidate profiles (current policy-state resolution, represented-person consent, referral-route attribution, GeoPose primary pose values). | Mixed (PREVIEW per part) |
| Cookbook | Worked examples, organized by scenario and tagged by tier. | Informative |

## Inherited feature surface

The following features are carried from the accepted v0.4 editorial candidate. The migration and release-candidate notes state the actual changes from the published v0.4 snapshot; this list does not claim every feature was published or implemented.

- **Abstract data model and production rules (Format Independence).** v0.5 defines the manifest as an abstract data model independent of serialization, plus production rules that write it to a concrete wire format. JSON-LD is the **reference encoding** and the default for interoperation; **CBOR-LD** is a second encoding defined in EXT-OPT to demonstrate format independence. Conformance is defined against the abstract data model, not against any single serialization. See Base Section 1.7 and Section 4.5; EXT-OPT Section O1.
- **Versioned context document with context integrity.** Term definitions for v0.5 are fixed by the versioned context document identified by `https://universalmanifest.net/ns/v0.5`; the exact current source body is `schema.jsonld` in this directory, reproduced verbatim with its content hash in Base Appendix B. The current public and local-candidate snapshots are distinguished under Namespace URIs below. Evaluators MUST pin the content-hashed

context and verify its multihash (the expected value is under Namespace URIs below and in Base Appendix B) before trusting term definitions, rather than fetching an arbitrary remote context at evaluation time. The `um:` term prefix is version-independent and expands to `https://universalmanifest.net/ns/um#`. See Base Section 6.10 and Appendix B.

- **Bounded selective minimum disclosure.** v0.5 names the core privacy behavior in plain language: each viewer, verifier, system, or use case receives only the minimum information it is allowed or needs to see. The bounded v0.5 profile is specified through Stage 3 Project, Stage 4 Consent, sealed-entry handling, and structured receipts. It does not by itself claim unlinkability, full or partial, cover-slot indistinguishability, resolver-operator privacy, or complete resistance to residual fingerprinting. See Base Sections 3.1.3, 3.1.4, 3.3, and 7.1.
- **Tier-1 binding made concrete.** Any claim relied upon at Tier 1 or above now requires a verified **holder binding** proving the presenting subject is the credentialed holder, in three modes (`sd-jwt-kb`, `bbs-holder-commitment`, `reciprocal-control`). **Presentation proof** (`presentationProof`) provides proof-of-possession that binds a manifest to a specific verifier and moment, preventing replay. **Liveness attestation** (`livenessAttestation`) proves human presence at its `attestedAt` time but not at verification time. See Base Section 6.6 and EXT-T1.
- **Per-facet liveness and assurance floors.** A facet can require a validated `requiredLiveness` floor (`minFreshness` plus optional `userVerified`) and, for locked-tier portable-unlock profiles, a `requiredAssuranceClass` floor (`software` < `hardware-uv` < `hardware-bound`). These floors compose with `requiredTrustTier`, can only raise the bar, and fail closed with receipt evidence such as `assuranceInsufficient` / `assuranceStatus`. PREVIEW. See Base Sections 2.1, 3.3.1, and EXT-T1 Section T1.3.
- **Tier-2 cryptographic binding profile.** The Tier-2 ZKP proof object, named but deferred in v0.3, is specified here: two proof profiles (`ZkLinkedSecretProof`, `ZkHdDerivationProof`) for proving control of multiple DIDs in zero knowledge, plus the BBS+ selective-disclosure track and a post-quantum migration path. PREVIEW. See EXT-T2 and Base Section 6.4.7.
- **Tier-3 multi-party ceremony.** Tier 3, reserved in v0.3, gains a concrete `ceremonyProof` (`ThresholdAttestationProof`) carrying a `threshold`, `attesters`, `ceremonyId`, and an `aggregateProof` for co-signed, highest-stakes deployments. PREVIEW. See EXT-T3 and Base Section 6.4.8.
- **Receipt-disposition vocabulary (extended).** The structured receipt carries richer disposition fields: `keyRefResolution`, per-consent `consentStatuses`, per-claim `claimStatuses` (with `claimRef`, status, and optional `tier`), `unprocessedEntries`, the credential-binding status fields (`holderBindingStatus`, `presentationProofStatus`, `livenessStatus`, `crossDidBindingStatus`), and `effectiveTrustTier`. The `facetStatuses` status set adds

`assuranceInsufficient` and `written` (both PREVIEW-driven) to v0.3's values. See Base Section 3.3.

- **Private-data handling and locked-facet auditability.** v0.5 adds per-facet cryptographic isolation obligations for deployments that claim isolation, write authorization via target-facet write-consent, private-data derived-write sensitivity ordering, key-lifecycle receipt events (`facet-key-rotated`, `facet-key-shredded`), and unlock-window audit fields for time-boxed access. PREVIEW where marked in the source profiles. See Base Sections 2.3.5, 3.1.4, 7.2 and EXT-OPT Sections O3 and O4.5.
- **statusRef resolution schema.** v0.5 defines the revocation-status resolution protocol and its response schema (`manifestId`, `status` in `active` / `revoked` / `suspended`, `updatedAt`, optional `reason` / `cursor` / `nextCheck`) and the status-endpoint conformance criteria. See Base Section 3.4 and EXT-OPT Section O2.
- **Bilateral session model and manifest forwarding.** The Base carries the bilateral field surface (`exchangeId` and `evaluatorId` on receipts) and the bilateral-exchange security model. EXT-OPT specifies the full session object (`um:BilateralSession` with `sessionId`, `exchangeId`, two `participants`, `initiatedAt`, `expiresAt`, and strictly-forward lifecycle `state`), paired-receipt correlation, receipt hash-chaining (`seq` / `prevHash`), federation, and manifest forwarding across resolver operators. See Base Section 3.5 and Section 6.4.6; EXT-OPT Sections O3 and O4.
- **JWE algorithm constraints.** Encrypted facets gain a normative MUST-implement baseline algorithm pair and an evaluation contract for algorithm negotiation. See Base Section 2.4.
- **Optional structural extensions.** EXT-OPT adds the agent-delegation scope registry, category trust and `trustWeight`, an `actorState` member, a two-component `devices` schema, and a derived-variant sensor-consent vocabulary, each independently optional. See EXT-OPT Sections O6 through O9.
- **Open signature-suite set, and Signature Profile B (JCS + ES256).** v0.5 stops pinning signing to a single algorithm. The recognized suite set is stated normatively to be **governed and open**: profile identity is the explicit (`algorithm`, `canonicalization`) pair, the `algorithm` enumeration in `schema.json` is the current materialization of the profile registry (EXT-OPT Section O6, `signature` category), and any number of further suites are recognized by that same registration step rather than by re-architecture. The first additive suite lands with it: **Signature Profile B** (`ES256` — ECDSA over NIST P-256 with SHA-256; `JCS-RFC8785` canonicalization and a signing input both identical to Profile A; `signature.value` is the base64url of the raw 64-byte `R || S`, with DER rejected; and deliberately **no low-S rule and no RFC 6979 requirement**, so a Secure Enclave, StrongBox, or TPM key is conformant as it signs). Profile B exists for hardware key custody: those chips hold P-256 keys and only P-256 keys, so without it the `hardware-bound` assurance rung is unreachable on the hardware

people carry. Signature Profile A (Ed25519) remains the default and the mandatory-to-implement baseline for every evaluator; Profile B is optional-additive for evaluators; an unrecognized (`algorithm`, `canonicalization`) pair is rejected fail-closed and never reinterpreted. Profile B itself adds no context term. The v0.5 context reuses the accepted local editorial definitions, while the published v0.4 context has different bytes and remains immutable; see Migration below. See Base Sections 1.5.1 and 1.6; EXT-OPT Sections O6, O10, O11, and O15.

- **Design-stage candidate profiles (PREVIEW, not yet on the wire).** Four candidate registered profiles are drafted in EXT-OPT: **current policy-state resolution** (supersession pointers, policy windows with deny/escalate dispositions, and `um:reason:policy:*` codes — is a consent/authorization record still current before a governed or agentic action proceeds?), **asset-bound represented-person consent** (the consent state of the person depicted, scanned, performed, or replicated *inside* a portable asset, as distinct from asset/token ownership, with carrier binding and currency resolution), **signed referral-route attribution** (verifiable route provenance, route-scoped disclosure via consent conditions, and minimal attribution receipts — with ad-network, marketplace, payment-rail, and buyer-data-sharing behavior explicitly out of UM scope), and **GeoPose primary pose values** (an OGC GeoPose payload or pointer as the primary position-plus-orientation value for an asset, object, dataset record, or scene element, with UM governing signature, provenance, consent/current-state policy, projection, and receipts around that value). These are design drafts: their members are not in the published schema/context or fixture suite and carry no conformance tests until finalized. See EXT-OPT Sections O2.6, O12, O13, and O14.

Signature Profile A (JCS + Ed25519) remains the default and mandatory evaluator baseline, but its acceptance predicate is explicitly tightened for wire `0.5` by Base §1.6.1.1. Its discriminator pair, SPKI carriage and signing-input procedure remain; Profile B is additive and its raw `R || S`/high-S behavior is unchanged. The six-stage evaluation sequence (Arrive, Verify, Project, Consent, Compose, Receipt) remains the conceptual spine, with binding-verification sub-steps in EXT-T1.

## Build your own implementation (any language)

Use the specification and fixtures as your source of truth, regardless of language, runtime, or encoding:

- Read the normative spec text at the immutable canonical repository revision identified by the chosen specification release; use `/spec/v0.5/` as its generated reading view (start with the Base). The main branch remains the working copy.
- Read conformance requirements: `spec/v0.5/CONFORMANCE.md`

- Validate against the structural schema: `spec/v0.5/schema.json`
- Use the JSON-LD context: `spec/v0.5/schema.jsonld`
- Run the implementation-neutral suite flow: `conformance/README.md` with v0.5 fixtures at `conformance/v0.5/`
- Treat reference-implementation reports as bounded evidence for their named versions and inputs; the public specification package does not include a development runner.

## v0.5 artifacts (this directory)

- `README.md` - this file
- `CONFORMANCE.md` - conformance requirements (evaluator, holder, bilateral participant) plus the fixture index
- `schema.json` - JSON Schema for structural validation of v0.5 manifests
- `schema.jsonld` - JSON-LD context defining v0.5 terms
- Signature Profile A: Base §1.6, including all six v0.5 acceptance predicates. Historical v0.2 profile text remains under `/spec/v0.2/SIGNATURE-PROFILE.md`; it does not define the tightened new-version predicate. Signature Profile B remains in EXT-OPT Section O15.

## Namespace URIs

- Manifests MUST set `@context` to include the v0.5 versioned context document `https://universalmanifest.net/ns/v0.5` (or include this URI when an array form is used).
- The versioned context URI identifies an immutable document. Its normative body is `schema.jsonld` in this directory; the verbatim body and its content hash are reproduced in Base Appendix B. Evaluators verify the sha2-256 multihash `zQmR1W9vsPrdbXmhH38jdNhdEhBB1C7JUL1Z9kZMi6u16i7` over the exact document bytes rather than trusting any retrieval channel (Base Section 6.10; computation rule in EXT-OPT Section O1.7).
- The `um:` term prefix is version-independent and expands to the base IRI `https://universalmanifest.net/ns/um#`.
- Endpoints: `https://universalmanifest.net/ns/v0.5`, `https://universalmanifest.net/ns/universal-manifest/v0.5/schema.jsonld`, `https://universalmanifest.net/ns/universal-manifest/v0.5/schema.json` and

`https://universalmanifest.net/spec/v0.5/`. The `schema.jsonld` and `schema.json` in this directory are the release artifacts, and the served bytes must equal them.

- `https://universalmanifest.net/ns/universal-manifest/v0.5/schema.jsonld` — context endpoint for the exact bytes of `schema.jsonld` in this directory
- `https://universalmanifest.net/ns/universal-manifest/v0.5/schema.json` (`schema.json` in this directory)
- `https://universalmanifest.net/spec/v0.5/` (rendered from the document set in this repository)

**Public v0.4 and new-candidate context identity.** The published v0.4 state is the canonical repository snapshot `c13fb2c3de157050a5862695d1e6ae8d8943deb5`, which carries no release tag. Its repository and hosted v0.4 context bytes have SHA-256 `91167fd510c39f93a023fc15a7474b77bc4414188ddb2d73806dc08e0f69345c`, multihash `zQmY71ixd6ez6UkXpzGJ7fiMpwL26jzQwWp9CV6kwX4Kb4f`. Those bytes remain valid and must be preserved at the old identity. This v0.5 candidate intentionally reuses the exact local editorial context bytes, SHA-256 `27ab5a3cc491cee7a5947b73f550c07ecbff285a81a6ba801aa98f60046dcea4`, multihash `zQmR1W9vsPrdbXmhH38jdNhdEhBB1C7JUL1Z9kZMi6u16i7`; its URI differs because the contract version differs, although no version-specific IRI occurs inside the context body. Base Appendix B reproduces that same body. A content hash identifies bytes, not the complete versioned processing contract. Do not replace the old public context or reinterpret old signatures.

## Six-stage evaluation sequence (summary)

Every conformant evaluator MUST execute these stages in order for every v0.5 manifest it processes. Each stage produces a defined output that feeds the next, and the evaluator MAY stop early at any stage by emitting a rejection.

1. **Arrive** - parse the manifest from a conformant encoding; confirm required envelope members and that `@type` includes `um:Manifest`; preserve (but do not act on) unknown members so they survive signature verification.
2. **Verify** - verify the signature over the canonical signing bytes; enforce freshness (TTL and 60s clock-skew); when a claim leans on stronger identity binding, run the Stage-2 binding sub-steps (EXT-T1); optionally resolve revocation status.
3. **Project** - apply bounded selective minimum disclosure: extract only the facets, claims, pointers, and devices the viewer, verifier, system, or use case is allowed or needs to see. A facet that is absent is *not projected*, not evidence of absence.

4. **Consent** - match `consents[].facetRef` against each projected facet's `@id`; check scope, purpose, and validity window. Sealed (encrypted) facets are acknowledged as present, never guessed at.
5. **Compose** - produce one of four outcomes: `accepted`, `accepted-with-warnings`, `accepted-partial`, or `rejected`. The result MUST be machine-readable and MUST include per-facet status.
6. **Receipt** - emit a structured receipt that honestly records what the evaluator did, including the gap between what a manifest declared it needed and what could be verified. Receipts MAY be signed (Signature Profile A) and MAY themselves be chainable manifests (EXT-OPT).

See Base Section 3.1 for normative details and CONFORMANCE for the portable contract and evidence limits.

## Tiered trust model (summary)

Four tiers, strictly additive. A claim verified at a higher tier automatically satisfies every lower tier. Relying parties choose their tier from their own threat model; the specification mandates no minimum.

- **Tier 0 - Signature-only.** Zero friction. Claims are self-asserted; the only cryptographic check is the manifest signature. MUST NOT be used as sufficient assurance for Sybil-critical decisions.
- **Tier 1 - Attested / proof-backed.** Low friction. Claims carry external proof of issuance plus a verified holder binding proving the presenter is the credentialed holder. Evaluators MUST enforce attester trust policy and freshness on proof material. Mechanics in EXT-T1.
- **Tier 2 - Cryptographic binding (ZKP).** Medium friction. Control of multiple DIDs is proven in zero knowledge without revealing key material. Profile in EXT-T2 (PREVIEW).
- **Tier 3 - Multi-party ceremony.** High friction. Multiple attesters co-sign via a threshold protocol. Profile in EXT-T3 (PREVIEW). Evaluators encountering a tier with no implemented verification profile MUST record `trustTierUnsupported` and MUST NOT downgrade.

See Base Section 6.4.2 for normative tier definitions, Section 6.4.5 for `requiredTrustTier` enforcement, and Section 6.4.6 for bilateral exchange semantics.

## Format independence (summary)

v0.5 separates *what a manifest means* from *how it is written down*.

- **Abstract data model.** A manifest is a set of abstract properties and semantics (the envelope members, facets, claims, consents, pointers, devices, the signature, and the receipt). Conformance is defined against this model.
- **Production rules.** A production rule writes the abstract model into, and reads it back from, a concrete wire format. JSON-LD is the **reference encoding** that conforming implementations are expected to interoperate with by default; **CBOR-LD** is a second encoding defined in EXT-OPT. The same manifest MAY be produced in any conformant encoding without changing its meaning.
- **Canonical signing.** The signature is computed over the canonical signing bytes defined by the production rule, so a manifest verifies independently of the encoding it traveled in.

See Base Section 1.7 and Section 4.5; EXT-OPT Section O1.

## Security considerations (selected)

v0.5 carries forward all v0.2 and v0.3 security guidance and adds:

- **Context integrity** is normative (Base Section 6.10): evaluators MUST pin the content-hashed v0.5 context document and verify its hash before trusting term definitions, rather than dereferencing an arbitrary remote context at evaluation time. The expected multihash and the verbatim context body are reproduced in Base Appendix B (and under Namespace URIs above).
- **Holder binding is required for Tier 1+** (Base Section 6.6, EXT-T1): a claim relied upon at Tier 1 or above without a verified holder binding MUST be capped at Tier 0. Liveness attestation proves presence only at `attestedAt`; evaluators MUST NOT treat a stale liveness attestation as current human presence.
- **Replay resistance** (EXT-T1): an interactive presentation MUST carry a `presentationProof` binding the manifest to the verifier's `challenge` and `audience`.
- **JWE algorithm constraints** (Base Section 2.4): encrypted facets MUST support the baseline algorithm pair; the evaluation contract governs which algorithms an evaluator accepts.
- **Bilateral correlation tokens** (EXT-OPT Section O4): `sessionId` and `exchangeId` MUST each be generated with at least 128 bits of entropy and MUST NOT encode party identifiers.

- `requiredTrustTier` **enforcement** remains normative and raise-only - claim and facet floors MUST NOT lower the manifest-level floor.

## Privacy considerations (selected)

- **Selective minimum disclosure is a bounded Base behavior** (Base Sections 3.1.3 and 7.1): each viewer, verifier, system, or use case receives only the minimum information it is allowed or needs to see. This is holder-controlled projection plus consent gating, sealed-entry handling, and receipt evidence.
- **Do not overclaim unlinkability.** The bounded v0.5 profile does not by itself prove verifier-verifier unlinkability, issuer-verifier unlinkability, cover-slot indistinguishability, resolver-operator privacy, or absence of residual fingerprinting. The BBS+ holder-binding mode (EXT-T1, EXT-T2) and the Tier-2 Track B path are PREVIEW, and they establish none of these properties in this version.
- **Holder-controlled disclosure** is a first-class behavior: evaluators cannot infer what was withheld, and a not-projected facet is never treated as evidence of absence.

## Complementary standards and scope boundaries

UM is an envelope, evaluation, and receipt layer that composes with neighboring standards. It does not replace them, and several capability families are deliberately deferred rather than silently missing.

**Carrier and provenance standards (C2PA, COG, and similar).** A manifest can carry — or point to — an external-standard artifact unmodified, while UM supplies the governance record around it: signature, consent and current-state policy, projection, and receipt evidence. The carried artifact remains independently verifiable under its own standard: a receiver performs two independent checks (the UM envelope evaluation, and the carried standard's own verification), and passing one never substitutes for the other. UM defines no C2PA/COG trust-list, certificate-chain, asset-binding, or media-provenance verification semantics. See EXT-OPT Section O1.6. The same posture holds for **OGC GeoPose** (O14 PREVIEW carries or points to GeoPose payloads; validating GeoPose data and reference frames remains OGC's job) and for **W3C VC/DID material** (carried and verified as claims/claimProof; UM issues nothing and mandates no DID method).

**Explicitly deferred capabilities.** The following capability families remain outside the current Base contract or require a future registered profile. PREVIEW work remains subject to the maturity and evidence limits in CONFORMANCE; this list does not certify implementation coverage:

- **C2PA-carry / external-standard-wrapper profile** — a dedicated carried-standard verification-result receipt field and an unknown-carried-standard disposition value (the wrapper *architecture* itself is v0.5 PREVIEW, Section O1).
- **Holder-generated ZK predicates and range proofs** — v0.5 supports issuer-asserted threshold claims (e.g. `ageOver`) and Tier-2 privacy hooks; holder-side predicate synthesis is a future registered profile, and an issuer-signed boolean is not equivalent to a holder-generated proof.
- **Proof-of-personhood uniqueness** — v0.5 carries provider attestations at Tier 1 with namespace separation and freshness; a one-person-one-account primitive (nullifier-style per-context uniqueness) is a future profile. The Tier-2 profiles prove cross-DID *control*, not uniqueness.
- **Dedicated storage-policy vocabulary** — v0.5 composes consent `scope`/`conditions`/purpose-binding, encrypted facets, sealed entries, and pointer-first storage; a declarative `storagePolicy` object, retention-duration tiers, and a storage-conflict reason code are future registered-profile work, and unenforceable storage requests fail closed today.
- **WASM content trust and inter-service authorization** — spatial/RP1-driven registered-profile candidates (publisher/module-hash trust chains; service-to-service message authorization), not Base additions.
- **Compact CBOR-LD/COSE handshake profile** — the CBOR-LD encoding is PREVIEW now; the finalized compact profile (token registry, packet-size/latency budgets, fail-closed timeout disposition, carried-status-only freshness) is future. UM here is compliance-enabling and audit-friendly; it certifies no aviation, traffic, facility, medical, or industrial-control system.
- **Cross-encoding signature portability** — v0.5 signs per production (re-encoding requires re-signing); one abstract-model signature verifying across encodings is an open decision.
- **Gig/creator profile family** — engagement-agreement objects, a no-custodial-intermediary invariant, dispute/non-delivery dispositions, and non-custodial settlement conventions are future profiles; payment rails, custody, pricing, fees, and escrow execution are out of UM scope but compose with UM-carried context and receipts.
- **Child-protection profile family** — age-band assurance context, guardian delegation, policy-snapshot-bound revocable consent, and privacy-preserving minor status are described as *capabilities*; normative promotion is gated on a legal/children's-rights review, and UM certifies no COPPA/GDPR/Children's-Code/eSafety compliance.
- **W3C Data Integrity proof suites** — future additive profiles beside Signature Profile A, never a silent replacement. The alternate-signature-suite slot this line reserved is no longer empty: v0.5 recognizes **Signature Profile B** (JCS + ES256) under a suite set that is now normatively

governed and open, so a Data Integrity suite is added by registration rather than by re-architecture. A Data Integrity proof suite specifically remains FORWARD. See *Inherited feature surface* above, Base Section 1.6, and EXT-OPT Sections O6 and O15.

**Domain integration guidance (non-normative).** RP1/spatial-fabric, smart-glasses/bystander consent, healthcare handoff, education/competence credentials, social/profile projection, agent ecosystems (ERC-8004, MCP, A2A, OASF), OMATrust proof alignment, and public discovery/publication policy are supported today by composing Base + EXT mechanics; they are documented as integration guidance (`integrations/`), not Base normative scope. Domain-specific profiles may be registered later through EXT-OPT Section O6.

**Not UM scope (composes, never owned).** Ad-network/marketplace operation, payment rails, custody, escrow execution, buyer-data sharing; C2PA/COG verification semantics; GeoPose validation; VC issuance and DID-method governance; authentication and federation protocols (OIDC, ActivityPub/AT); legal or regulatory compliance certification; and device/TLS/network side-channel control.

## Status

v0.5.0 is published for review and implementation. It keeps the wire files of release candidate 1 unchanged and corrects text only. Its supported runtimes are the two reference evaluators: TypeScript (`universal-manifest` 0.1.0 on Node.js 22) and Rust (`um-core`, built for macOS on Apple silicon). Each matches all 198 cases of the v0.5 index and all 79 of the v0.4 index. No browser, game-engine or device runtime is supported in v0.5. Qualification of the protected exchange, with the release obligations that depend on it, is planned for v0.6. Individual advanced/optional surfaces retain their existing PREVIEW grades; a Grade-2 candidate member is not promoted into the schema/context by this version change. The public v0.4 snapshot and older versions remain available at their existing identities. No full capability, hardware, protected-carrier or institutional-approval claim is made.

Selective minimum disclosure is in v0.5 as a bounded Base profile: holder-controlled projection, consent gating, sealed-entry handling, and receipt evidence. The deeper high-privacy per-target work remains a separate implementation chain: v0.5 does not claim unlinkability, full or partial, cover-slot indistinguishability, resolver-operator privacy, or complete high-privacy conformance unless a later profile explicitly does so.

Future changes are proposed as issues and pull requests on the canonical `UniversalManifest/universal-manifest` repository, and its maintainers accept them after

review there. Change control for the media types and the profile registry rests with those maintainers.

## Release boundary

v0.5 releases as a mechanism with reference suppliers. It claims no real-world identity, issuer or revocation result. All 43 obligations and the 40 permission outcomes stay in force and stay counted. This boundary limits what a result can show, and it removes no requirement. Real suppliers are later work, and this release does not drop them.

- **First contact.** A counterpart check holds only inside a provisioned deployment, where the deployment installs trust roots and party records out of band. It shows that a party controls a key that the deployment's root named for this use. It does not show who the party is. This release's evidence uses a reference root. This release makes no claim about first contact between strangers.
- **Issuer authority.** A relying party names the issuers it accepts for each claim. The reference issuer issues test credentials only. v0.5 does not show that any issuer is authoritative, that a claim is true of a person, or that a holder is the subject.
- **Status.** This release makes no revocation-aware claim. It selects no status method, and receipts report `unchecked`. Short-lived re-issuance limits staleness only up to the expiry. It is not immediate revocation, and it gives the issuer a log of who refreshed and when. No public status pointer is required.

## High-privacy mode

High-privacy mode is an unresolved draft. This candidate selects no high-privacy mechanism, and this release does not offer the mode as available. In the draft, a presentation shows a new holder key for each check in place of a stable holder.

The mode does not make a holder unlinkable. This release claims no unlinkability, full or partial, and gives no measured result that the mode hides a holder or keeps two checks apart.

The draft is for one-off checks only. It requires a new holder key for each check. It must refuse delegated and long-lived use, and this release states that those uses are linkable. It is not promised for checks where lending defeats the purpose, such as age checks. This release offers no age check for real use. Its age tests run on reference credentials, reliance on them is refused, and they state no real fact about a person. Base §7.1 states these limits.

The open parts are:

- **No adopted method.** No method for this mode is adopted into this specification. Private method work continues toward a later version, and no privacy measurement of it exists.
- **Lending and pooling.** The proof shows only that someone knows the credential's secret. The draft gives a verifier no way to detect or limit lending or pooling. The limit is stated here, and stating it does not remove the cost.
- **Holder association.** Nothing ties the credential's secret to a person, or shows that the holder is the subject.
- **Key reuse.** A reused holder key repeats in clear on the wire, so one reuse links those two checks outright. The draft cannot detect or prevent this.
- **Values a verifier sets.** These values can link one holder's visits, and a holder cannot make them fresh. A holder can catch a repeat. A verifier can still give each holder its own value, verifiers can share logs, and network observers can watch.
- **Group values.** Authority, policy and credential-group values must be the same for every holder. If they differ, they sort holders into smaller groups or mark them. Where these values come from is open.
- **Issuer and verifier together.** The reference issuer holds a log of who refreshed and when. If an issuer and a verifier compare records, refresh times can link a holder. Refreshes would need a schedule independent of use, and no scheduled refresh exists.
- **Linkable receipt fields.** Some receipt fields stay classed as linkable. No receipt has actual recipients or a retention period set, so no route that sends a receipt to a recipient is open.
- **Not measured.** Other carried values, size, timing and retry surfaces, linkage on matched data, credential-group size against credential lifetime, and the quality of the random source.

## Migration from published v0.4

This candidate selects `manifestVersion: "0.5"`, `@context` including `https://universalmanifest.net/ns/v0.5`, and the `spec/v0.5/` artifact family. Every new v0.5 manifest must be newly signed over its exact production signing input; changing the version/context of a v0.4 document invalidates its old signature. Keep original signed v0.4 bytes, context and interpretation unchanged. Consumers supporting old versions must route by the declared version to that version's rules, not apply the new Profile A predicate retroactively. A consumer without a supported declared-version path rejects the document; it does not silently downgrade or relabel it.

Relative to the saved published v0.4 snapshot, the candidate context omits exactly `facetSensitivityOrdering`, `labels`, `order`, `profile` and `trustWeight`; shared term definitions are unchanged. Sensitivity ordering remains profile-supplied and the derived-write join requirement remains normative. `trustWeight` remains specification-only until its value shape and schema/fixture/context coverage are settled; this candidate adds no replacement private carrier. The context pins under Namespace URIs distinguish public old bytes from candidate bytes, even though the accepted local v0.4 editorial context and the v0.5 context are intentionally identical.

The candidate schema materializes Profile B (`ES256`, JCS, P-256) alongside Profile A. Profile B uses raw 64-byte `R || S`, rejects DER signatures, accepts high-S signatures and imposes no RFC 6979 requirement; these rules are unchanged from the accepted editorial contract. The inherited fixture plan adds three valid ES256 controls (minimal, facets, high-S) and five invalid controls (bad value, DER encoding, key substitution, SPKI algorithm mismatch, post-signing tampering). The old sensitivity-ordering fixture remains preserved historically but is absent from the accepted candidate index. The unsupported-profile negative uses `ES384` instead of now-recognized `ES256`. These inherited controls are now freshly signed and exercised for wire0.5 through the TypeScript command adapter; the two explicit v0.3 compatibility fixtures retain their original bytes. Finite fixture success does not qualify all normative behaviors.

For v0.5 Profile A, canonical non-identity prime-order A and R, canonical S and the full-challenge uncofactored equation are required by Base §1.6.1.1. Weak identity, small-order and mixed-order points, noncanonical point encodings and noncanonical scalars cannot gain acceptance through library-specific normalization or a cofactored shortcut. Some inputs accepted by older library routes will be refused by the new rule. An ordinary valid signature is not an identity-authority proof: the existing key-resolution/authorization and tier rules still apply. Do not infer this rule from a function called strict verification; qualify the actual adapter against all six mathematical requirements.

## Release-candidate notes and remaining qualification

The new source set has thirteen documents/artifacts: the eight-document tier stack, README, CONFORMANCE, draft profile registry, structural schema and context. It retains all 43 evaluator/holder/bilateral/suite obligations and conditional advanced/protected requirements, along with existing honest-result, evidence-ceiling and proximity limitations. Version selection does not select an unresolved high-privacy mechanism, adopt a private request/permission/receipt carrier, grant private initial trust or turn PREVIEW into implemented cryptography.

**Dated finite qualification, 2026-10-21.** The candidate index contains **188 cases**, all matched by the actual TypeScript command adapter. Its SHA-256 is `6334fe4f61198d087ddb8260ba467ba42507dfb852d2643d5de84619dc804e3b`. The strict

schema/ingress sweep passed **192 checks**: 188 indexed inputs, three examples, and one targeted malformed-JWE control; the current API qualification passed **38 named groups**. 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. Caller `intendedScope` and `intendedPurpose` remain explicit context, never inferred from a grant; omitted, empty, or mismatching purposes cannot authorize a purpose-bearing consent. The separate **46/46 I-JSON checks are dated 2026-10-16** and are not a rerun of the changed evaluator. The finite report remains at conformance level `none`; positive Tier-1+ verification, authenticated services, other holder implementations and protected/private mechanisms remain unproved. Refusal evidence is not positive conformance.

**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.

**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. This is finite local evidence, with conformance level none.

**I-JSON ingress (2026-11-06).** Base section 1.6.3 now requires the whole manifest as received to be I-JSON, including members outside the signing input. A fresh check had found that the TypeScript evaluator accepted a 1.2 MB signed manifest with a lone surrogate inside `signature`, which the Rust evaluator rejected. Both now reject such input. 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. Entries dated earlier describe the earlier index. `CONFORMANCE.md` section 5 lists the known differences between the two evaluators outside the suite.

**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.

Detached TypeScript and Rust production Profile A predicates, in the separately dated **2026-10-16** experiment, agreed on **937 identical inputs: 3 accept, 934 reject**, with every result matching its retained expectation. This is signature-predicate agreement, not full evaluator, SDK, application, or runtime parity. The current corpus retains the two encrypted-Entity manifests signed for that qualification, preserving four accepted two-recipient ciphertexts. TypeScript verifies their outer signatures and records all **four facets as opaque**; it does not decrypt them. Separate Rust/panva-jose calls read those persisted signed files and recovered the exact canonical Entity bytes in **16 opens: 8 cross-implementation and 8 self-controls**. Earlier Entity evidence of **112 ordered wrong-key refusals reaching unwrap and 24 tamper refusals** is reused through exact unchanged ciphertext and dependency identities, not rerun or counted as additional fixtures. The older primitive experiment (8 cross-opens, 36 wrong-key refusals, 156 malformed/tampered refusals) remains separate and is not combined into these totals. See the conformance qualification limits and implementation-specific JWE limits.

**Bounded Rust and local two-party evidence (2026-10-21).** An explicit v0.5 Rust core/command adapter separately matched all **188 unchanged cases**, including 119 with strict receipt assertions, and passed 25 HTTP controls on its phase-one binaries. Two persistent Swift callers subsequently exercised actual Rust evaluations, signed actions, receipt verification and reciprocal acknowledgments over injected local channel evidence. The full **52-control online suite per version**, including eight completed encounters each, belongs to the preserved bilateral predecessor. The final compatibility successor was qualified by all eight positive v0.5 encounters, one v0.4 encounter, eight HTTP responses, two byte-equal legacy comparisons, four byte-equal witness comparisons and focused independent controls; the predecessor's full matrix is not reassigned to that successor. Exact binary identities, API boundaries and test lineage are recorded in the reference-implementation evidence in CONFORMANCE §8.

Across four retained caller runs and a fresh final-binary run, independent inspection traced **270 action/delivery/persistence links and 54 verified signed receipts across 27 completed encounters**. These are sums across distinct scoped runs, not a single matrix or certification total. Receipt bodies match both actual evaluator digests and signatures verify against independently installed role keys. The fresh run contains two mixed-suite positives and a signed-body tamper refusal. The reused FFI library supplies unchanged signing/prerequisite machinery, not a newly qualified v0.5 ABI evaluator. This local evidence does not establish holder generation, peer identity, real radio, source permission or withdrawal, protected adoption, physical/device behavior, whole-SDK parity or complete conformance; level remains `none`. No fixture, wire member, profile or normative obligation changed.

The contract clarifications preserve distinct boundaries: a status locator/cursor inside the excluded `signature` object is not envelope-authenticated; current permission needs independently trusted authority and target binding; opaque cursors and HTTP ETags supply no implicit ordering or permission grant. Capability claims identify implemented, partial, and unimplemented obligations with actual checks and unmet baselines. Circuit verification uses trusted pinned key/public-input bindings, and model-integrity claims require a profile-defined computation and evidence binding. A request omitting its purpose cannot match purpose-bearing consent. These clarifications add no protected current-state mechanism, capability carrier, signed-receipt mandate, atomic-write mechanism, or new proof-system choice.

All 43 mandatory/conditional obligations and advanced protected/privacy requirements remain. The finite corpus and source evidence above are reconciled, and each release records its own reader, public byte, route and citation checks. Historical v0.4 reports remain old-version evidence. Publication as a tagged release claims no native publication, full implementation qualification, hardware provenance, privacy certification, or institutional ratification.