Universal Manifest v0.5 — EXT-T3

Tier-3 Multi-Party Ceremony Profile · Preview

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/ext-t3/

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.

---

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.

This is a **companion profile** to the Universal Manifest v0.5 Base specification. It specifies the **Tier-3** multi-party ceremony model — co-signing by multiple independent keyholders for the highest-assurance identity binding. Read the Base, EXT-T1, and EXT-T2 first.

Status: All content in this profile is PREVIEW — under active review, built on the editors' recommended defaults, and fully revisable before the v0.5 schema is locked.

EXT-T3: Tier-3 Multi-Party Ceremony Profile PREVIEW

Companion to: Base specification, building on EXT-T1 and EXT-T2. **Registry category:** `trust`.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC 2119] [RFC 8174] when, and only when, they appear in ALL CAPITALS, as shown here.

T3.0. Scope and Relationship to the Base

The Base defines **Tier 3 — Multi-party ceremony** (Base §6.4.2) as the highest assurance level, in which multiple keyholders (potentially different people in different locations) co-sign, analogous to multi-sig wallets, and reserves the `ceremonyProof` field on `identity.crossDidBinding` claims (Base §6.4.4). This profile specifies the ceremony model, the `ceremonyProof` schema, the attester-role taxonomy, threshold-protocol selection, and the verification procedure.

Because trust tiers are strictly additive, a Tier-3 deployment MUST also implement EXT-T1 (Tier-1 binding) and, where it relies on Tier-2 cryptographic binding as the per-attester proof, EXT-T2.

T3.1. Ceremony Model

A Tier 3 ceremony is a protocol by which multiple independent attesters jointly verify and co-sign a cross-DID binding. The ceremony produces a single aggregate proof represented in a `ceremonyProof` object on the `identity.crossDidBinding` claim.

A `ceremonyProof` MUST contain:

- `type` — MUST be `"ThresholdAttestationProof"`.
- `threshold` — A string in the format `"M-of-N"` (e.g., `"3-of-5"`) declaring the quorum requirement: `M` is the quorum, `N` the size of the ceremony's signing group. `M` and `N` MUST be positive integers with `M ≤ N`; a `threshold` violating this (e.g., `"5-of-3"`) is malformed.
- `attesters` — An array of DID strings identifying the ceremony participants who *contributed* to the aggregate proof. The array MUST contain at least `M` and at most `N` entries, and the entries MUST be distinct — a duplicated DID MUST NOT be counted toward the quorum.
- `ceremonyId` — A globally unique URI identifying this ceremony instance.
- `aggregateProof` — The aggregate proof value. The encoding is defined by the protocol identified in `protocol` (below).

A `ceremonyProof` additionally carries:

- `protocol` — OPTIONAL in v0.5. The v0.5 schema and fixture suite do not require it, and a later version is expected to make it REQUIRED. Identifies the threshold protocol — and thereby the encoding and verification algorithm — that produced `aggregateProof`. Values are registered through the registration mechanism's subsidiary value registries (EXT-OPT §O6); candidate tokens: `"frost-ed25519"` (FROST, [RFC9591]) and `"threshold-bbs"` (Threshold BBS+ [Threshold-BBS]). Without this discriminator an `aggregateProof` is interpretable only with out-of-band knowledge (contrast `bindingProof.cryptosuite` / `proofSystem` in EXT-T2 §T2.1); a `ceremonyProof` without `protocol` is therefore, in v0.5, evaluable only by out-of-band protocol knowledge and is otherwise recorded `"trustTierUnsupported"` (T3.4).

A `ceremonyProof` violating any of the MUST-contain structural constraints above is **malformed** (the absence of `protocol` is not a malformation in v0.5); a malformed `ceremonyProof` MUST be treated as a proof that fails verification (T3.4). *(Wire note: the v0.5 schema defines the `ceremonyProof` slot with `type`/`threshold`/`attesters`/`ceremonyId`/`aggregateProof`, requires unique attester strings and tolerates additional members. The evaluator enforces positive M,N, M ≤ N and M ≤ distinct attesters ≤ N; the schema threshold pattern alone still admits `"5-of-3"`. `protocol` — like `attesterRoles`, T3.2 — remains a design-stage candidate carried as a tolerated member. These structural checks do not select a protocol or verify an aggregate.)*

T3.2. Attester-Role Taxonomy

Ceremony participants MAY declare roles. The following roles are recognized:

- `"subject"` — The manifest subject, asserting their own identity.
- `"witness"` — An independent party who attests to the subject's identity through direct verification.
- `"custodian"` — A keyholder who holds a key share on behalf of an organization.
- `"auditor"` — A party who reviews the ceremony transcript and co-signs as a quality check.

Role carrier (candidate). Roles are declared in a candidate OPTIONAL `attesterRoles` member of `ceremonyProof`: a map from attester DID to role token, where each key MUST be a member of `attesters` (T3.1) and each value MUST be a role from the taxonomy above or a role registered through EXT-OPT §O6. `attesters` itself remains an array of plain DID strings — the parallel map adds roles without changing that array's published shape; the two representations are reconciled at wire freeze. A map containing a key that is not a member of `attesters` is malformed, and a malformed `attesterRoles` MUST be treated as absent (roles undeclared).

Profile documents MAY define role constraints (e.g., "at least 1 witness and 1 custodian"). When role constraints are declared, evaluators MUST verify that the declared roles satisfy them (T3.4); when `attesterRoles` is absent or malformed, roles are undeclared, an undeclared role satisfies no role constraint, and the verification therefore fails (fail closed). **Binding:** a profile that declares role constraints MUST also define how role declarations are bound into the ceremony transcript covered by `aggregateProof`; a role declaration not covered by the ceremony's signed material is unattested presenter-supplied metadata and MUST NOT be used to satisfy a role constraint — otherwise any presenter could relabel participants to satisfy a constraint the ceremony never established.

T3.3. Threshold Protocol Selection

v0.5 defines the ceremony slot (the `ceremonyProof` schema) and proof format. The specific threshold protocol used to produce the aggregate proof is deferred to profile documents: the protocol in use is *identified* in-band by the `protocol` member (T3.1), while its *definition* — the `aggregateProof` encoding, the verification algorithm, and the ceremony-transcript binding (including any role binding, T3.2) — lives in the registered profile document behind that identifier. Implementations MAY use any threshold protocol that produces a verifiable aggregate proof.

Non-normative guidance: FROST (Flexible Round-Optimized Schnorr Threshold signatures, [RFC9591]) is a candidate threshold protocol for Ed25519-based ceremonies; it produces standard Schnorr/Ed25519 signatures from a threshold of signers (FROST "signers" are UM "attesters" here), and verifiers see a normal signature. Threshold BBS+ (Doerner et al. 2023) [Threshold-BBS] is a candidate for anonymous credential issuance ceremonies. Neither protocol is normatively required; profile documents define the binding.

T3.4. Verification Procedure

When an evaluator encounters an `identity.crossDidBinding` claim carrying a `ceremonyProof`, the evaluator MUST apply the following procedure (the declared `requiredTrustTier` governs what the verified result must satisfy, not whether the proof is verified):

1. Parse the `threshold` string to extract `M` and `N`, and confirm they satisfy `M ≤ N` (T3.1).
2. Verify that the `attesters` array contains at least `M` and at most `N` distinct entries.
3. Identify the threshold protocol from `protocol` (T3.1) when present. `protocol` is OPTIONAL in v0.5 (T3.1), so a `ceremonyProof` without it is evaluable only where the evaluator has out-of-band protocol knowledge. If the evaluator cannot identify the protocol, or does not implement

the identified protocol, it MUST record `"trustTierUnsupported"` (below) rather than guess at the `aggregateProof` encoding.

4. If role constraints are declared in the governing profile, verify that a well-formed `attesterRoles` declaration satisfies them, under the transcript-binding rule of T3.2.
5. Verify the `aggregateProof` against the declared attesters using the protocol identified in step 3, as defined by its registered profile document.
6. For each attester DID, confirm the attester is on the evaluator's configurable trust list (per Base §6.4.4).
7. Record the result. A verified ceremony elevates the claim to Tier 3. A `ceremonyProof` that is malformed (T3.1) or that *fails* verification under this procedure MUST cause the claim to be recorded as `failed` and treated at Tier 0, regardless of any accompanying attestations; only an *absent* `ceremonyProof` falls back to lower-tier evaluation of whatever binding material is present. (An unsupported `protocol` is recorded `"trustTierUnsupported"`, not `failed` — the evaluator cannot verify and MUST NOT downgrade, per Base §6.4.5.)

Evaluators that do not implement any Tier 3 threshold protocol — or none matching the declared `protocol` — MUST record the claim as `"trustTierUnsupported"` per Base §6.4.5.

(Worked example: Tier 3 ceremony proof — see Cookbook.)

Open question: Whether the ceremony model should support asynchronous signing (signers sign at different times) or require synchronous co-presence is open. Asynchronous ceremonies are more practical for geographically distributed keyholders. Feedback is requested.

References

- **[RFC9591]** FROST: Flexible Round-Optimized Schnorr Threshold Signatures.
- **[Threshold-BBS]** Doerner et al. (2023), Threshold BBS+ Signatures.

✧ ✧ ✧

Companion to the Base specification. For the lower tiers this builds on, see EXT-T1 and EXT-T2.