HUMANVERIFIED
CHAIN OF CUSTODY

# Building the Evidence Trail

HUMANVERIFIED CHAIN OF CUSTODY

Technical implementation guide for a bounded proof of concept across media, software and manual craft.

> ***A clearer record of how work moved from source to outcome.***

| | |
|---|---|
| **STATUS** | Proposed proof of concept |
| **VERSION** | v0.1 |
| **DOCUMENT ID** | HV-COC-TIG-2026-09-03-v0.1 |
| **PUBLICATION DATE** | 3 September 2026 |
| **PREPARED FOR** | everwished innovation |

This publication describes a proposed engineering evidence trail. It is not a deployed verification service, a legal certificate, or proof of complete authorship or truth.

# Executive summary

This guide specifies a proposed implementation of HumanVerified Chain of Custody. The system would create a readable Evidence Trail from signed checkpoints, artifact digests, version relationships, review decisions and independently anchored log records. It is intended to test whether selected stages in research, documentation, code, video editing and manual craft can be made more legible without implying more than the evidence supports.

The core technical claim is deliberately narrow:

> *A valid record can support the conclusion that a registered credential completed a defined ceremony for identified artifact bytes or versions, and that the resulting receipt was accepted and logged under a named policy.*

That evidence does not establish the participant's uncoerced intent, complete authorship, semantic truth, genuine sensor input, legal ownership, originality, competence or uninterrupted physical custody. A signed declaration remains a declaration. Device attestation describes selected device, app or key properties; it does not attest camera photons or the physical scene. A transparency log makes certain changes or omissions detectable relative to trusted checkpoints; it does not make the submitted claim true.

This document is ready to support architecture review, estimation and a bounded design spike; it is not yet a frozen implementation contract. Before production PoC work begins, the team must select and freeze the signed-envelope profile, exact Merkle encoding and test vectors, versioned API schemas, key-management provider, evidence-retention policy and supported adapter fixtures. It does not state that the service is live. All effectiveness, performance, usability and cost questions remain proposed measurements for the PoC.

# 1. Scope and non-goals

## 1.1 PoC tracks

The common protocol will be tested through three tracks:

1. Research, documentation and code: sources, tool-use declarations, draft versions, review checkpoints, source commits, build inputs and release artifacts.
2. Video capture and editing: independently keyed capture streams, closed source segments, ingredient relationships, edit/export declarations and C2PA 2.4 Content Credentials.
3. Manual craft and physical custody: material declarations, work-in-progress checkpoints, inspections, custody transfers and a deliberately limited QR/NFC-to-object binding.

## 1.2 Non-goals

Version 0.1 will not attempt to:

1. determine whether content is true;
1. infer whether a natural person authored all or part of an artifact;
1. infer an absence of AI contribution from missing metadata;
1. decide ownership, copyright, consent, legality, safety, quality or suitability;
1. create a universal reputation score;
1. make an automated recruitment or employment decision;

- treat Steam, a device platform, a security key, a camera, a C2PA manifest or a ledger as a universal trust root;
- publish raw identity evidence, video, private prompts or stable device identifiers; or
- describe an incomplete PoC as a live assurance service.

Steam may be evaluated later as an optional distribution or account-authentication channel. It is not evidence about camera input, OBS sources, editor completeness, runtime integrity or legal identity.

# 2. Claim taxonomy and public language

## 2.1 Evidence labels

Every public event must carry one of four labels. Labels are not assurance levels and must not silently upgrade one another.

| Label | Meaning | Required basis | Reader-facing limit |
|---|---|---|---|
| Declared | A participant asserted a fact and bound the assertion to a checkpoint credential | Valid participant signature and exact claim scope | The signature does not make the assertion true or complete |
| Evidenced | Machine-verifiable data or an independently recorded observation supports the displayed property | Passing validator, digest match, log proof or observation record | Only the named property is supported |
| Reviewed | A named reviewer assessed specified evidence under a displayed review policy | Reviewer credential, decision, scope, policy version and time | Review is bounded and may be mistaken |
| Unresolved | Evidence is missing, disputed, unavailable or outside scope | Explicit reason code | Absence must not be converted into a positive or adverse inference |

Cryptographic validity and human review are separate fields. A cryptographically valid false declaration remains false. A careful review without a valid artifact binding remains unable to authenticate the bytes.

## 2.2 Supported conclusions

Subject to passing the relevant validator, the Evidence Trail may report that:

- supplied bytes match a recorded SHA-256 digest;
- a registered credential responded to a fresh HumanVerified challenge with stated WebAuthn flags;
- the protected service accepted an event under a named policy at its recorded receipt time;
- an artifact version names specific parent or ingredient digests;
- an event receipt is included in a named Merkle checkpoint;
- a C2PA, in-toto, DSSE, SLSA or document-signature structure validates under the displayed trust policy;
- particular tools, actions, AI contributions and human-oversight states were declared to the displayed scope; and
- the record is current, superseded, revoked, expired or unable to be verified at lookup time.

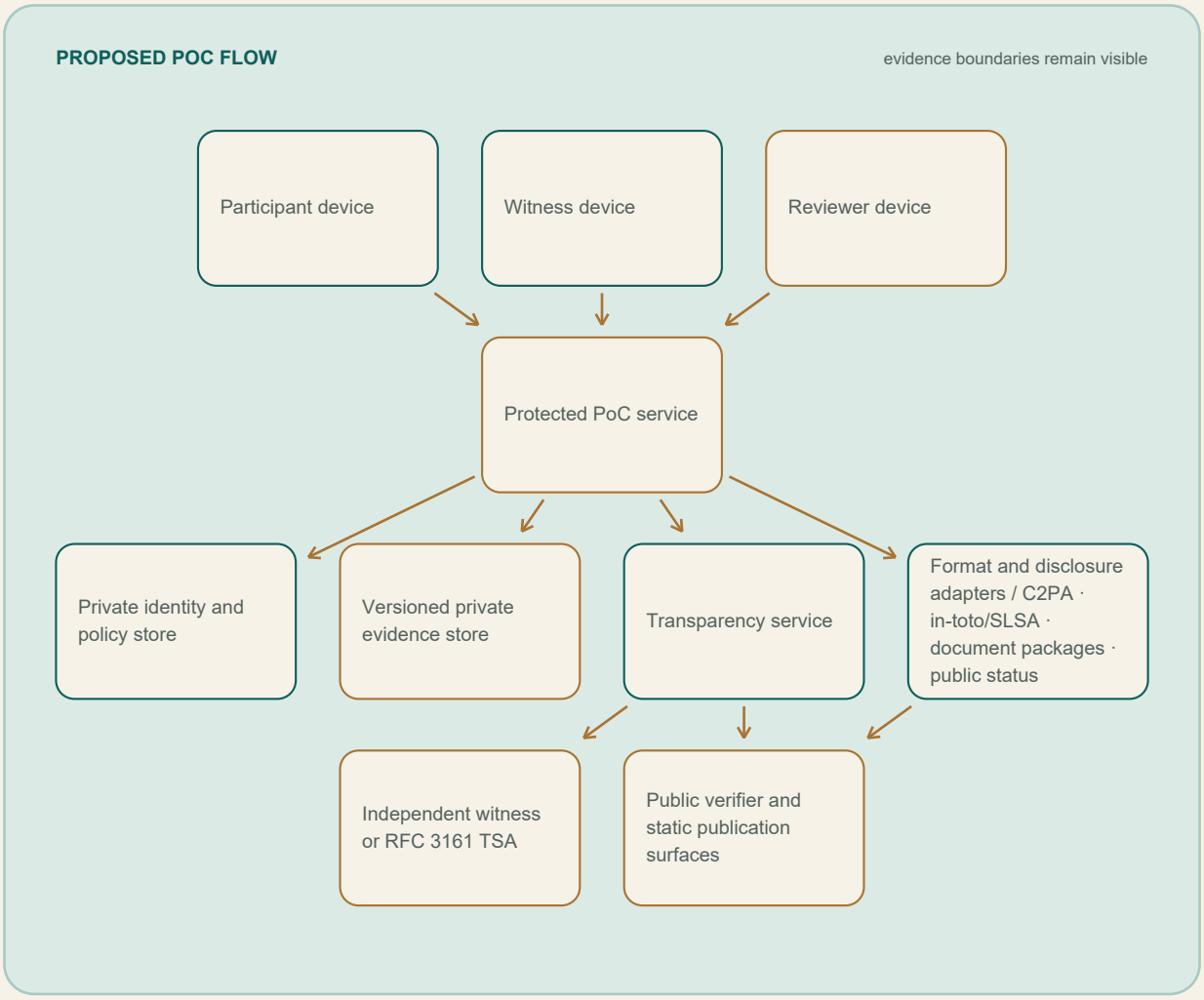
## 2.3 Unsupported conclusions

The same evidence does not by itself establish:

- that a credential holder understood, intended or freely approved the action;
- that one natural person exclusively controlled the credential;
- that every action or tool was observed or declared;
- that a recording shows a live or truthful scene;
- that a physical identifier stayed attached to the same object;
- that the signer owns or authored an artifact;
- that source material was lawful, accurate or consensually obtained;
- that a current status screenshot remains current; or
- that the record has legal chain-of-custody status or admissibility.

Public copy should use phrases such as “credential-signed checkpoint,” “artifact-version lineage,” “recorded to the stated scope” and “evidence of participation at named checkpoints.” It must not collapse signed-key evidence, person intent and sensor truth into one claim.

# 3. System context and trust zones




## 3.1 Trust zones

1. Participant zone. Devices perform local hashing, capture, declaration and credential approval. This zone is potentially compromised. Passing app or device attestation changes an evidence tier; it does not make the event content true.
2. Protected service zone. The service issues challenges, validates credentials and policy, maps pseudonyms to enrolled accounts, stores private evidence and signs acceptance receipts. Administrative access is auditable and separated from public publishing.
3. Transparency zone. Accepted receipt digests enter an append-only Merkle structure. Signed tree heads are witnessed outside the primary service. Raw evidence and identity records never enter the public log.
4. Public zone. The verifier exposes opaque record identifiers, artifact digests where disclosure is safe, bounded assertions, proofs, policy version and current status. The existing static websites remain publication surfaces rather than identity or evidence processors.

### 3.2 Data stores

| Store | Proposed contents | Public | Required controls |
|---|---|---|---|
| Identity registry | Account-to-pseudonym mapping, enrolment evidence reference, credential status | No | Encryption, least privilege, strong operator authentication, audited access, defined deletion/correction process |
| Credential registry | Credential IDs, public keys, attestation class, AAGUID where justified, counters as risk signals, revocation state | No | Field minimisation, key-status history, no cross-service tracking disclosure |
| Event store | Canonical events, signatures, policy decisions, parent relationships, status transitions | Limited projection only | Append-only application policy, version history, transactional sequence checks |
| Evidence object store | Closed artifact bytes, media segments, supporting observations | No by default | Versioning or retention lock, encryption, malware handling, access logging, retention and deletion schedule |
| Transparency log | Leaf commitments, tree roots, checkpoint signatures, consistency data | Roots and opaque proofs | Independent checkpoint publication, monitor comparison, key rotation history |
| Public receipt cache | Sanitized receipt, proof bundle, current state and explanatory text | Yes | Privacy review, anti-enumeration controls, signed responses or verifiable bundle |

Public commitments to low-entropy private data can be guessed by hashing likely values. Private evidence identifiers should therefore use random opaque IDs or keyed commitments. Plain artifact digests are suitable only when revealing equality and enabling independent file verification is intentional.

## 4. Roles, keys and failure domains

### 4.1 Roles

| Role | Responsibility | Must not be inferred |
|---|---|---|
| Participant | Declares an action and approves a checkpoint | Sole author, truthful declarant or uncoerced actor |
| Capture device | Produces a keyed capture record and artifact digest | Truthful sensor or independent witness |
| Witness participant/device | Records a second view or observation under a distinct key | Independence if it shares operator, account, app supply chain or administrator |

| Role | Responsibility | Must not be inferred |
|---|---|---|
| Reviewer | Assesses named evidence under a versioned policy | Universal approval or legal determination |
| Custodian | Accepts or transfers control of an artifact | Continuous physical possession before or after the recorded transfer |
| Service operator | Applies policy and signs an acceptance receipt | Source of truth about the underlying claim |
| Log operator | Builds Merkle trees and signs checkpoints | Non-equivocation without independent comparison |
| External witness/TSA | Confirms or timestamps a checkpoint commitment | Truth of leaves or participant intent |
| Public verifier | Recomputes selected validations and explains status | Authority beyond its configured roots and policy |

## 4.2 Key classes

Keys must be purpose-separated. A compromise in one class must not automatically authorize another class.

| Key class | Holder | Use | Proposed protection |
|---|---|---|---|
| Actor WebAuthn credential | Participant or reviewer | Approve event or decision | User verification required; hardware-backed tier tested separately |
| Capture-device key | Capture app/device | Sign segment metadata and close records | Platform keystore where available; attestation tier recorded |
| Receipt-signing key | Protected service | Sign accepted receipts | KMS/HSM target for a live pilot; dual control for policy changes |
| Log checkpoint key | Transparency service | Sign tree heads | Separate key and service identity from receipt signer |
| Witness key | Independent monitor | Countersign observed checkpoints | Separate organisation, account or administrative domain where possible |
| C2PA claim-signing credential | Publishing organisation | Sign Content Credentials | C2PA-compatible certificate and explicit trust policy |
| Document-signing credential | Publishing organisation | Future PAdES signatures | Organisation-controlled key with revocation and archival policy |

## 4.3 Two-device independence

“Two devices” is not equivalent to “two independent witnesses.” Each trial must record the failure domains actually separated:

1. distinct device keys and credential registrations;
1. distinct hardware and operating-system instances;
1. distinct user accounts and, where practicable, different administrators;
1. distinct camera paths and local storage;
1. different network paths where available;
1. whether the same app build, signing account, backend, cloud account or operator remains a shared dependency; and
1. whether a separate natural person controls the witness device.

The PoC will classify independence rather than assert it. Suggested classes are `dual_key_same_operator`, `dual_device_same_admin`, `dual_operator_shared_service` and `separate_operator_and_admin`. The protected service and common software supply chain remain shared failure domains in every class unless separately diversified.

# 5. Canonical event and receipt protocol

## 5.1 Why two canonical objects are required

The participant must sign a stable event before the service can record acceptance time or log position. The protocol therefore has two distinct canonical objects:

1. the checkpoint event, whose final fields are confirmed and signed by the participant credential; and
2. the acceptance receipt, created and signed by the service after verification.

`receivedAt`, policy outcome, assurance tier and log position belong in the acceptance receipt. Putting them into the participant-signed event would either require the client to sign unknown future values or create circular serialization.

## 5.2 Checkpoint event schema

The normative serialization is JSON conforming to the I-JSON constraints used by RFC 8785. Property ordering, whitespace and input object order do not affect the canonical bytes. Implementations must reject duplicate property names, non-finite numbers, invalid Unicode and values they cannot reproduce consistently.

| Field | Type | Requirement |
|---|---|---|
| `schemaVersion` | string | Fixed to `hv.event/0.1` for this PoC |
| `eventId` | string | Server-assigned opaque 128-bit-or-greater identifier |
| `projectId` | string | Opaque project identifier |
| `artifactId` | string | Stable opaque artifact identifier |
| `versionId` | string | Identifier unique within the artifact |
| `parentVersionDigests` | array of strings | Sorted lowercase SHA-256 digests; empty only for an origin event |
| `actorPseudonym` | string | Project-scoped pseudonym, not a public account identifier |
| `actorRole` | string | Controlled vocabulary |
| `actionType` | string | Controlled vocabulary such as `capture.segment.close`, `edit.export`, `review.accept`, `custody.transfer` |
| `declaredTool` | object | Name, version, provider and capture basis; unknown values stay explicit |
| `aiContribution` | string | `none_declared`, `assist_declared`, `generate_declared`, `mixed_declared`, `unknown` |
| `humanOversight` | string | `not_recorded`, `prompt_guided`, `human_validated` |
| `inputDigests` | array | Media type, algorithm, digest and disclosure class |
| `outputDigests` | array | Media type, algorithm, digest and disclosure class |
| `claimScope` | object | Purpose, recipient class, covered actions and exclusions |

| Field | Type | Requirement |
|---|---|---|
| `captureClass` | string | Recorded device/capture evidence tier |
| `occurredAtClaimed` | string or null | Participant-reported RFC 3339 time, never shown as trusted time |
| `sessionId` | string | Opaque session identifier |
| `streamId` | string or null | Capture/edit stream identifier where applicable |
| `sequenceNumber` | integer | Server-reserved monotonically increasing number for the stream |
| `challengeId` | string | Opaque identifier issued by the service |
| `nonce` | string | Base64url encoding of at least 256 random bits |
| `previousRecordHash` | string or null | Prior accepted receipt digest for this stream |
| `retentionClass` | string | Versioned policy identifier, not a free-form duration |
| `publicDisclosureClass` | string | `private`, `receipt_only`, `selected_fields`, or `public_artifact` |

Example event before canonicalization:

```
{
  "schemaVersion": "hv.event/0.1",
  "eventId": "evt_7f2b3c81d95f4e709178",
  "projectId": "prj_c32e1a08",
  "artifactId": "art_0e8f912c",
  "versionId": "v003",
  "parentVersionDigests": [
    "sha256:5d07ad6734625d7f23f9c9bcf171bce4e68782ba52d541dd44f3c5c78dca9320"
  ],
  "actorPseudonym": "actor_project_41",
  "actorRole": "editor",
  "actionType": "edit.export",
  "declaredTool": {
    "name": "Example Editor",
    "version": "captured-by-client",
    "provider": "declared",
    "captureBasis": "client_report_and_project_file"
  },
  "aiContribution": "assist_declared",
  "humanOversight": "human_validated",
  "inputDigests": [
    {
      "mediaType": "video/mp4",
      "algorithm": "sha256",
      "digest": "5d07ad6734625d7f23f9c9bcf171bce4e68782ba52d541dd44f3c5c78dca9320",
      "disclosureClass": "receipt_only"
    }
  ],
  "outputDigests": [
    {
      "mediaType": "video/mp4",
      "algorithm": "sha256",
      "digest": "017898a84788b4f43ca5ed390829f60df8f6b5c57ca12fc434c7909c64f919ad",
      "disclosureClass": "public_artifact"
    }
  ],
  "claimScope": {
    "purpose": "poc_video_export",
    "recipientClass": "public_verifier",
    "coveredActions": ["declared_export"],
    "exclusions": ["sensor_truth", "complete_edit_history"]
  },
  "captureClass": "device_key_platform_state_recorded",
  "occurredAtClaimed": "2026-09-03T10:15:30Z",
  "sessionId": "ses_b77a4901",
  "streamId": "stream_editor_01",
  "sequenceNumber": 14,
  "challengeId": "chl_e0ff141c",
  "nonce": "V3i5MO0dxF7B_fictitious_base64url_256_bit_value",
  "previousRecordHash": "sha256:
1149b75a4ae88dbfef0d71305fd6d901c34f451f46f6ef3b76a3cc5bf90f38aa",
  "retentionClass": "poc-sensitive-90d-v1",
  "publicDisclosureClass": "selected_fields"
}
```


The example values are fixtures, not production records.

## 5.3 Digest and challenge construction

```
eventBytes  = JCS(checkpointEvent)
eventDigest = SHA-256(eventBytes)
challenge   = SHA-256(
  UTF8("hv-event-v1") || 0x00 ||
  eventDigest || 0x00 ||
  serverNonce
)
```

The zero-byte separators and fixed domain string are normative. The service stores the exact challenge bytes and passes their base64url form to WebAuthn. It must compare the returned `clientDataJSON.challenge` to those exact bytes, validate `type`, origin, RP ID hash, signature, credential allow-list entry and required user-verification flag.

## 5.4 Nonce lifecycle and replay cache

1. An authenticated client submits a proposed event without `eventId`, `sequenceNumber`, `challengeId` or `nonce`.

2. The service validates field shape, artifact access, role, parent relationships and requested action.
3. In one transaction, it reserves an event ID and sequence number, creates a challenge ID and at least 256 random nonce bits, finalizes the event, computes its canonical digest and stores a replay-cache entry.
4. The replay entry contains credential allow-list, event digest, session, sequence, expiry and state `issued`.
5. The client displays the finalized event summary, verifies relevant fields locally, and requests credential approval.
6. On receipt, the service atomically changes the entry from `issued` to `consuming` before signature validation. Success becomes `consumed`; failure returns to `issued` only for explicitly retryable transport errors. Invalid signatures consume the challenge.
7. Expired, consumed, unknown, wrong-session, wrong-device or wrong-sequence challenges are rejected.
8. A short retention window preserves used challenge IDs for replay detection without retaining client secrets indefinitely.

The challenge supports server-observed freshness only for the ceremony it is bound to. It does not prove sensor freshness unless the unpredictable value is visibly or audibly captured and even then remains susceptible to sufficiently capable real-time simulation.

### 5.5 Acceptance receipt

The service validates the ceremony and signs a canonical DSSE-style envelope or equivalent purpose-bound structure containing:

```
{
  "schemaVersion": "hv.receipt/0.1",
  "receiptId": "rcp_d6821397",
  "eventId": "evt_7f2b3c81d95f4e709178",
  "eventDigest": "sha256:
16e289f08c584e2aa4c716fbac66fb04a013e045ab1486c21309aa46236cc248",
  "artifactDigests": [
    "sha256:017898a84788b4f43ca5ed390829f60df8f6b5c57ca12fc434c7909c64f919ad"
  ],
  "credentialEvidence": {
    "method": "webauthn",
    "userPresence": true,
    "userVerification": true,
    "attestationTier": "policy-recorded-not-public"
  },
  "policy": {
    "id": "hv-poc-video-export",
    "version": "0.1",
    "decision": "accepted_with_limits",
    "limits": ["sensor_truth_not_established", "complete_history_not_established"]
  },
  "receivedAt": "2026-09-03T10:15:43.281Z",
  "receiptSignerKeyId": "hv-receipt-poc-01",
  "publicDisclosureClass": "selected_fields"
}
```

The receipt signature authenticates service acceptance, not the truth of the participant declaration. `receivedAt` is server-observed time, not a trusted timestamp until an external timestamp or witnessed checkpoint covers the receipt.

## 6. WebAuthn policy

### 6.1 Registration

1. Enrol credentials only in an authenticated account ceremony linked to the existing HumanVerified identity-vetting scope.

1. Record the identity-vetting method and date separately from the credential. Do not imply that WebAuthn performed identity proofing.
1. Use a stable HumanVerified RP ID and an allow-list of permitted origins.
1. Request user verification for checkpoint credentials.
1. Generate random user handles that reveal no email address or civil identity.
1. Store credential public key, credential ID, transports where necessary, backup-eligibility/state flags, attestation tier and key-status history.
1. Validate attestation only under a written metadata and revocation policy. Unknown or absent attestation produces a lower/unknown tier rather than silent rejection unless a particular action requires hardware evidence.

## 6.2 Assertion validation

The service must validate all of the following before accepting a checkpoint:

1. expected challenge, type and origin;
1. RP ID hash;
1. credential ownership and project authorization;
1. signature over authenticator data and client-data hash;
1. user-presence and user-verification flags required by policy;
1. challenge state and expiry;
1. event digest, session, stream and sequence binding;
1. credential not revoked, suspended or outside allowed use;
1. extension outputs only when understood; and
1. backup/counter signals recorded as risk inputs rather than categorical identity evidence.

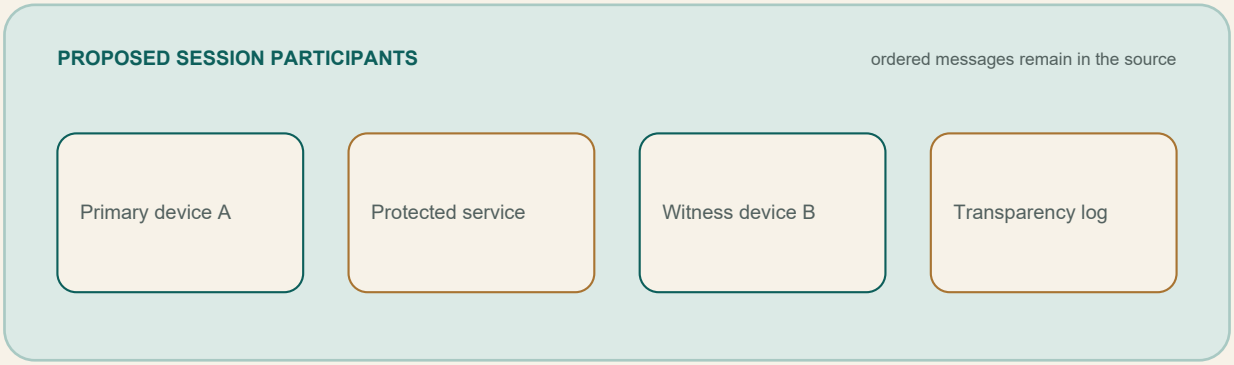
WebAuthn user verification indicates local verification by the authenticator. The W3C specification cautions that it does not provide concrete identity and that an authenticator may be shared. Signature counters may remain zero; a decrease is a signal for investigation, not proof of cloning.

## 6.3 Assurance tiers

The PoC will test, not pre-endorse, these descriptive tiers:

| Tier | Minimum evidence | Mandatory display limit |
|---|---|---|
| Credential signed | Valid WebAuthn assertion with user verification | Hardware and device state unknown |
| Hardware evidence recorded | Above plus validated authenticator metadata/attestation under named policy | Attestation does not identify an author or validate content |
| Dual-device corroborated | Two keyed streams with cross-recorded fresh challenges and terminal records | Shared operator/software/service failures remain |
| Reviewed | Scoped reviewer decision added to one of the above | Review is bounded to displayed material and policy |

# 7. Dual-device capture protocol




## 7.1 Session opening

The service creates a session only after both proposed devices authenticate. The session record contains expected streams, roles, credential IDs, independence classification, policy version, maximum duration and challenge schedule. A late-joining device creates a visible gap; the service does not backfill witness coverage.

## 7.2 Cross-challenges

Each device receives a different unpredictable challenge. The PoC may represent it as a short colour/shape sequence, an audible phrase and a machine-readable token. Device A records B presenting B's challenge; device B records A presenting A's challenge. Each signed segment event references the peer challenge ID and expected peer stream.

The challenge is evidence of interaction with fresh server data if captured and linked correctly. It remains unable to establish a genuine scene because a display, feed injector, virtual camera, compromised media path or coordinated simulator may produce the same pixels and sound.

## 7.3 Closed segments

Capture uses short finalized MP4 segments rather than one open-ended file. Each segment event contains:

1. stream and sequence number;
1. exact digest, byte length, media type and capture duration claim;
1. prior accepted segment receipt digest;
1. challenge and peer-stream references;
1. device-key signature and app/platform evidence tier;
1. local claimed start/end time and server receipt time kept distinct;
1. upload completeness status; and
1. privacy/retention class.

Original segment bytes enter the private versioned evidence store. The public log receives only the receipt commitment and privacy-reviewed opaque identifiers.

## 7.4 Terminal close records

Every stream ends in a signed terminal record with:

1. final sequence number and segment count;

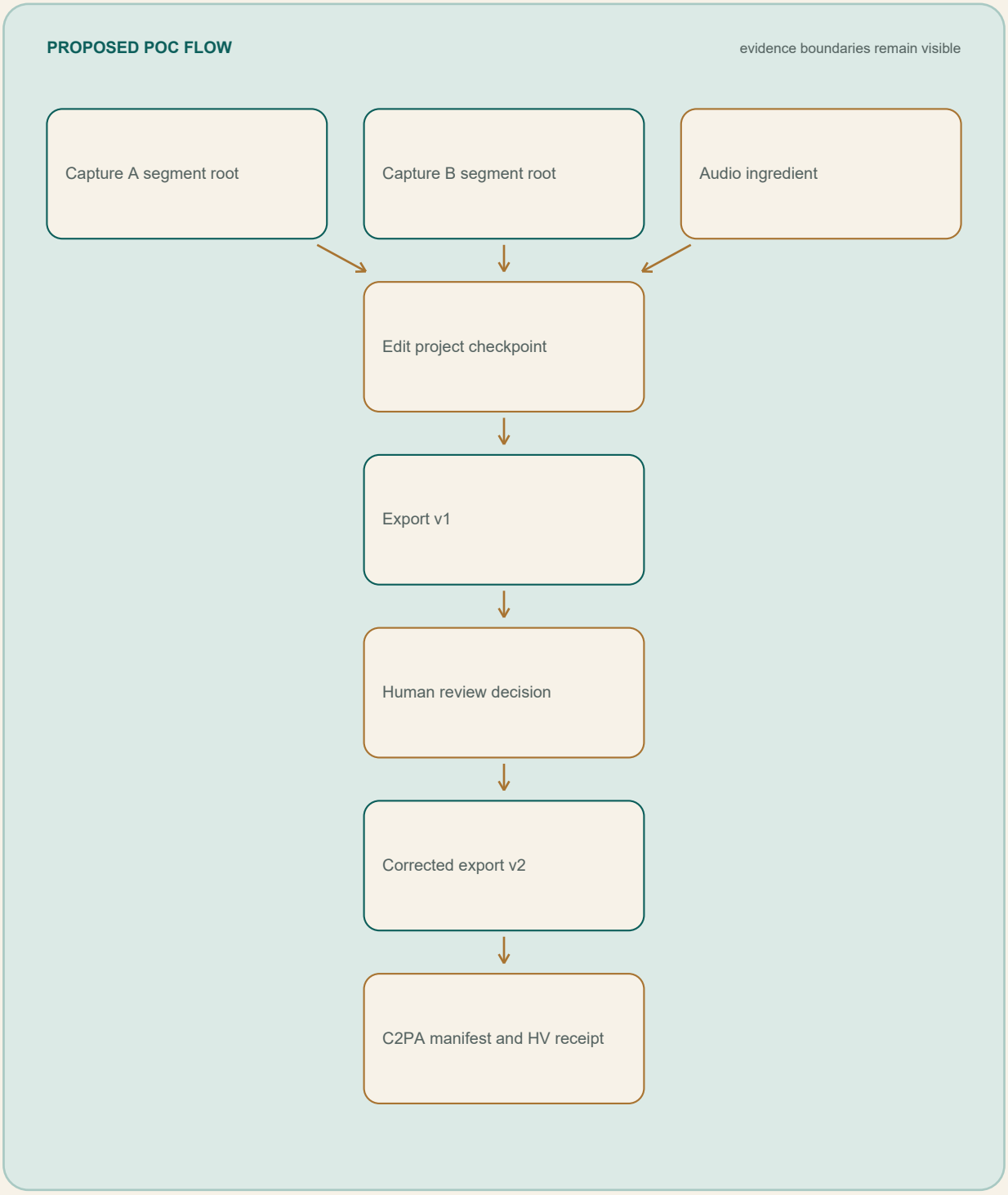
1. ordered segment-receipt digests or a stream Merkle root;
1. most recent peer-stream reference;
1. close reason: normal, interrupted, device failure, participant withdrawal or service failure;
1. missing-segment and upload status;
1. credential/device signature; and
1. service acceptance receipt.

A record without a valid terminal close remains `incomplete`; the verifier must not infer that the visible tail is the whole stream. Unexpected sequence gaps, a close count mismatch or peer streams that terminate inconsistently become `unresolved` events.

# 8. Editing and artifact lineage

## 8.1 Version graph

Artifact lineage is a directed acyclic graph. Every non-origin version names one or more parent digests. An edit export can have multiple ingredients; a split export can produce multiple children. The event stream records actions, while the version graph records byte relationships.




Rules:

1. parent digests are exact artifact hashes, not filenames;
1. arrays used as sets are sorted before canonicalization;

- every output digest is unique within an artifact version;
- origin events must declare why no parent exists;
- missing parent bytes may be recorded as unavailable but never silently ignored;
- supersession adds a new record and does not overwrite prior events; and
- a reviewer decision identifies the exact digest assessed.

### 8.2 Controlled editor records

The editing adapter will record proposed fields for:

- ingredient hashes and disclosure classes;
- editor project/configuration hash;
- application, plugin and export version strings captured or declared;
- edit categories and generative actions;
- export settings and output digest;
- known unrecorded operations;
- reviewer decision, exceptions and exact reviewed output; and
- signed handover to the next custodian.

An observation camera may corroborate an editing session but is not the primary action ledger. Deterministic action records, project-file snapshots and artifact hashes are preferred over claims that every displayed pixel has been understood.

## 9. Format adapters

### 9.1 C2PA 2.4 media adapter

For supported media, the adapter will propose a C2PA 2.4 manifest containing:

- claim generator identity and version;
- ingredient relationships and hashes;
- actions and digital-source types;
- AI-use disclosure compatible with the 2.4 assertion model;
- `humanOversightLevel` when it accurately maps to recorded evidence;
- `allActionsIncluded` only when the implementation can support that declaration;
- HumanVerified receipt identifier or repository receipt where appropriate; and
- organisation claim signature and validation material.

The public verifier checks structural validity, asset binding, signer certificate path under the configured C2PA trust policy and consistency with the HumanVerified receipt. It displays unknown or untrusted signers separately from invalid manifests.

C2PA validation supports cryptographic bindings and signer-linked assertions. It does not establish truth, authorship, ownership, a genuine camera scene or absence of undisclosed actions. CAWG identity assertions, if later used, indicate authorization or active participation and must not be described as attribution

or ownership.

The C2PA 2.4 specification includes PDF embedding, but current SDK format support must be tested. The PoC go/no-go gate is based on actual fixture writing and validation, not specification capability alone.

## 9.2 Research, documentation and code adapter

The adapter will record:

1. source URL or repository reference and captured retrieval time;
1. digest of retained source snapshot where lawful;
1. declared tool/model use and exact identifiers only when accurately captured;
1. prompt/evidence category without publishing a private prompt transcript;
1. draft and review digests;
1. Git source commit or an explicit `sourceRevision: unavailable` state;
1. in-toto/DSSE statements for named process steps; and
1. SLSA provenance for build outputs.

Git identifies source state but not natural-person authorship. in-toto and DSSE bind typed statements to keys and subjects but do not define the truth of the payload. SLSA provenance describes build inputs, builder and process at its stated level; it is not a person-intent signal.

## 9.3 Document release adapter

This v0.1 publication release uses:

1. editable Markdown as the source;
1. a publicly readable PDF with AES-256/R6 permission restrictions;
1. printing and accessibility extraction permitted;
1. modification, annotation, form filling, assembly and ordinary copying requested as denied;
1. a SHA-256 checksum over the exact Markdown bytes;
1. a SHA-256 checksum over the exact final PDF bytes; and
1. an unsigned JSON release manifest and checksum list.

These permission settings are advisory controls interpreted by conforming software. Checksums detect a changed copy only when a verifier compares it with a trusted checksum source. This release does not include an organisation-controlled document signature.

A future pilot target is PAdES signing with an organisation-controlled certificate and RFC 3161 trusted timestamps. PAdES B-LT or B-LTA may be selected where long-term validation material and archival policy justify it. An RFC 3161 timestamp supports the conclusion that referenced data existed before the timestamp; it does not establish authorship or semantic truth. If C2PA PDF writing is not supported by the selected library, the document lineage remains in a detached provenance package until compatibility is demonstrated.

## 9.4 Manual craft and tag adapter

The craft adapter records material input declarations, work-in-progress artifact hashes or observations, maker-signed checkpoints, review observations, custody transfers and final multi-angle captures.

A QR code or ordinary NFC UID is transferable and cloneable. A challenge-response cryptographic NFC tag can support possession of the responding key and make simple copying harder, but it still does not prove continuous physical attachment or that the tagged object is unchanged. The PoC will attempt tag transfer, object substitution, staged work, display replay and omission of unobserved steps. Any undetected attack remains a documented boundary.

Tag public data must contain only an opaque artifact identifier and verifier URL. It must not expose the participant's identity, private evidence location or bearer authorization token.

# 10. Transparency log, checkpoints and witnesses

## 10.1 Leaf construction

The public log leaf is a privacy-reviewed commitment, not a copy of the receipt:

```
leafInput = UTF8("hv-log-leaf-v1") || 0x00 || receiptDigest
leafHash  = SHA-256(0x00 || leafInput)
nodeHash  = SHA-256(0x01 || leftChild || rightChild)
```

The leaf/node prefix pattern follows the separation principle used by certificate-transparency Merkle trees. The exact PoC encoding must be frozen in test vectors. The public leaf endpoint may expose opaque receipt ID, receipt digest, leaf index and proof; it must not expose private event fields.

## 10.2 Signed tree heads

A tree head contains log ID, tree size, root hash, checkpoint time, algorithm, log key ID and prior witnessed checkpoint reference. The log signs each tree head. Verifiers request:

1. inclusion proof from receipt leaf to tree root;
1. consistency proof from a previously trusted tree size to the new tree size; and
1. witness or timestamp evidence covering the signed tree head.

## 10.3 Witness model

At least one failure domain outside the primary service should observe selected tree heads. Proposed options are:

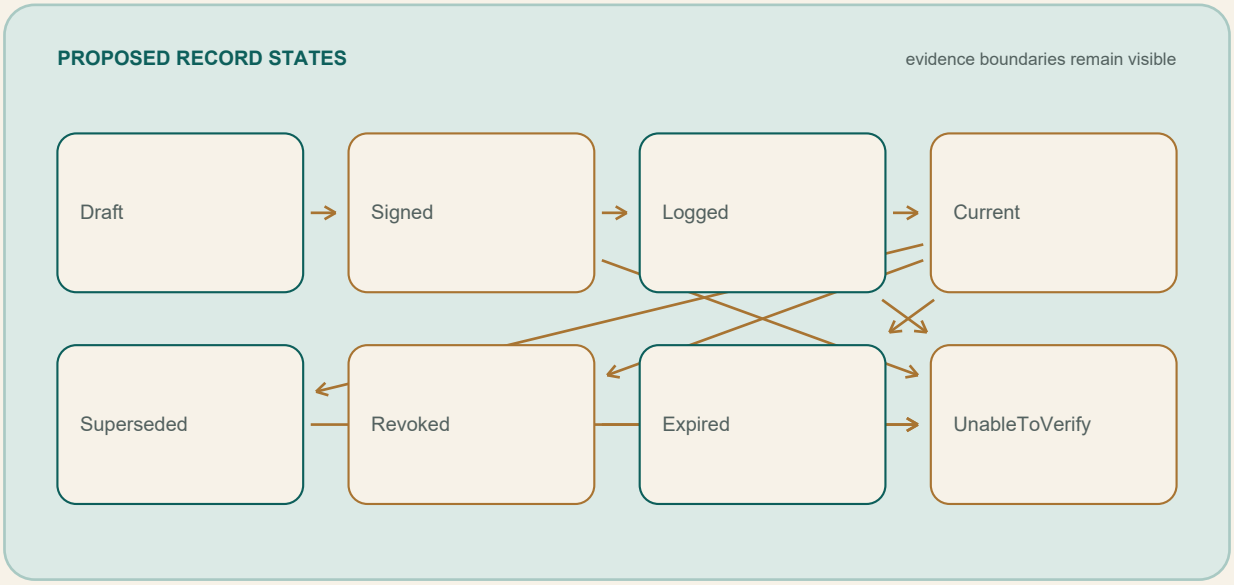
1. an independent monitor that stores and countersigns tree heads;
2. an RFC 3161 timestamp authority covering the signed tree-head digest; or
3. publication of tree heads to a separately administered public repository plus active monitoring.

An external timestamp alone does not detect every split view. A monitor must compare tree heads and consistency proofs across observations. The PoC will simulate equivocation by presenting two roots for the same tree size and verify that the witness raises an alert.

No raw video, identity evidence, prompts, credential IDs, device identifiers or private locations enter the public ledger. Public entries carry opaque identifiers and cryptographic roots only.

# 11. Public verifier

## 11.1 State machine




`unable-to-verify` is not equivalent to invalid, false or malicious. The verifier must give a reason code and safe next action.

## 11.2 Verification order

1. Parse record and enforce schema/version limits.
2. Hash supplied artifact bytes, if present, and compare exact digest.
3. Validate receipt signature and key status at relevant times.
4. Recompute event digest from canonical event when the event is disclosed.
5. Validate participant assertion evidence and recorded WebAuthn flags.
6. Validate parent/version graph and terminal stream records.
7. Validate C2PA, in-toto, SLSA or document adapter evidence.
8. Verify log inclusion, consistency and witness/timestamp evidence.
9. Fetch current status, revocation, supersession and policy information.
10. Render evidence labels and an explicit “supports / does not establish” explanation.

### 11.3 Public response example


```
{
  "recordId": "hve_4f83bc12",
  "state": "current",
  "checkedAt": "2026-09-03T12:00:00Z",
  "artifact": {
    "digestMatch": true,
    "algorithm": "sha256",
    "digest": "017898a84788b4f43ca5ed390829f60df8f6b5c57ca12fc434c7909c64f919ad"
  },
  "evidence": [
    {
      "label": "Declared",
      "statement": "An editor declared an assisted workflow for this export.",
      "valid": true
    },
    {
      "label": "Evidenced",
      "statement": "The receipt is included in witnessed checkpoint 482.",
      "valid": true
    },
    {
      "label": "Unresolved",
      "statement": "Complete edit history was not recorded.",
      "valid": null
    }
  ],
  "supports": [
    "exact_bytes_match",
    "registered_credential_checkpoint",
    "witnessed_log_inclusion"
  ],
  "doesNotEstablish": [
    "sensor_truth",
    "complete_authorship",
    "ownership",
    "complete_edit_history"
  ]
}
```


### 11.4 Accessibility and failure presentation

1. Do not encode status by colour alone.
1. Put state and reason in text before technical details.
1. Provide keyboard access and logical focus order.
1. Use headings, lists and tables compatible with assistive technology.
1. Provide a compact human explanation and an expandable technical proof view.
1. Identify the exact time of the latest online status check.
1. Mark screenshots, printable badges and exported receipts as point-in-time copies.
1. Do not punish a participant for missing evidence without a human, context-aware process and an accessible alternative.

## 12. Proposed API surface

The API is illustrative but sufficiently specific to build contract tests. All write endpoints require authenticated, authorized sessions and idempotency keys. JSON requests use UTF-8 and reject unknown critical fields.

| Method and path | Purpose | Important responses |
|---|---|---|
| `POST /v1/credentials/registration-options` | Begin WebAuthn enrolment | Challenge, RP data, allowed algorithms, policy |
| `POST /v1/credentials/registrations` | Complete and validate enrolment | Credential status and evidence tier |

| Method and path | Purpose | Important responses |
|---|---|---|
| `POST /v1/sessions` | Open a project/capture session | Session ID, policy, expected roles/streams |
| `POST /v1/events/challenges` | Validate proposal and issue finalized event/challenge | Canonical event, digest, challenge, expiry |
| `POST /v1/events` | Consume challenge and submit signed event | Signed acceptance receipt or structured rejection |
| `POST /v1/streams/{id}/close` | Submit terminal close record | Close receipt and completeness state |
| `POST /v1/reviews` | Record scoped reviewer decision | Review receipt |
| `POST /v1/custody-transfers` | Offer/accept transfer with two ceremonies | Linked offer and acceptance receipts |
| `GET /v1/records/{opaqueId}` | Return authenticated private record | Full authorized view |
| `GET /public/v1/records/{opaqueId}` | Return privacy-reviewed verification view | State, explanations and proof links |
| `GET /public/v1/log/checkpoints/latest` | Return current signed tree head | Checkpoint and witness material |
| `GET /public/v1/log/proofs/{receiptDigest}` | Return inclusion/consistency material | Proof bundle |

Event creation errors use stable codes such as `challenge_expired`, `challenge_consumed`, `origin_mismatch`, `rp_id_mismatch`, `user_verification_required`, `credential_revoked`, `sequence_conflict`, `parent_missing`, `artifact_digest_mismatch`, `policy_denied` and `evidence_unavailable`. Error text must not reveal whether unrelated record IDs or credentials exist.

# 13. Privacy, safety and governance

## 13.1 Data minimisation

1. Use project-scoped pseudonyms and random public IDs.
1. Do not put civil identity, email, identity-document attributes or raw attestations into public receipts.
1. Retain raw prompts only when a specific test requires them, with separate access and retention policy.
1. Keep raw video, audio, locations and device evidence private and retention-limited.
1. Publish only opaque ledger IDs, receipt commitments, tree roots and necessary validation material.
1. Treat pseudonymized information as personal data for parties able to reconnect it.
1. Record purpose and recipient class on each event to reduce recontextualization.
1. Provide correction and supersession without rewriting historical receipts.
1. Define participant withdrawal and deletion consequences before collection begins.

## 13.2 Pre-participant gates

Before trial capture begins, the operator must complete:

1. DPIA screening and, where indicated, a full DPIA;
1. lawful-basis and special-category analysis where biometric recognition is proposed;
1. retention, deletion, subject-rights and incident workflows;

1. processor and international-transfer review;
1. informed participant notice and non-coercive alternatives;
1. accessibility and exclusion assessment;
1. safeguarding scope confirmation; and
1. human responsibility for any candidate or worker decision.

The PoC must not turn into covert or compulsory worker monitoring. Missing evidence is an unresolved state, not an automatic adverse decision.

# 14. Threat model

## 14.1 Assets and adversaries

Protected assets include identity mappings, actor and service credentials, raw evidence, artifact bytes, event order, policy decisions, status/revocation history, public interpretation and participant rights records.

Relevant adversaries include a dishonest participant with a valid credential, an account thief, malware on a capture/editor device, a colluding witness, a malicious reviewer or operator, a compromised build or media tool, an external attacker, a dishonest log operator and a party reusing a valid receipt in a misleading context.

## 14.2 Priority threats and controls

| Priority | Threat | Proposed controls | Residual boundary |
|---|---|---|---|
| P0 | Valid credential signs a false declaration | Exact scope, visible `Declared` label, reviewer separation, anomaly review | Signatures cannot make statements true |
| P0 | Account takeover or signing oracle | WebAuthn UV, short challenges, transaction summary, revocation, rate limits, device-risk signals | Malware or coercion may still obtain approval |
| P0 | Capture or editor substitution | Hash before/after boundaries, closed segments, project snapshots, second stream, build/media adapters | Compromised paths may coordinate false evidence |
| P0 | Omitted actions or stream tail | Sequence reservation, parent graph, terminal close record, visible incomplete state, C2PA completeness semantics | Unobserved activity outside the system remains possible |
| P0 | Log deletion or split view | Merkle consistency, signed tree heads, independent witness/monitor, external timestamp | Witness collusion or outage remains |
| P0 | Privacy and correlation leakage | Private/public separation, opaque IDs, keyed commitments, retention limits, DPIA, anti-enumeration | Public artifact equality may itself reveal information |
| P1 | Receipt or tag recontextualization | Bind artifact, version, purpose, recipient class and expiry; show record context | Misleading screenshots or copied physical labels remain possible |
| P1 | Physical tag transfer or clone | Cryptographic NFC experiment, secondary inspection, destructive/visible attachment, custody events | Tag response is not proof of attachment |

| Priority | Threat | Proposed controls | Residual boundary |
|---|---|---|---|
| P1 | Screen, optical or virtual-camera replay | Cross-challenges, dual views, platform evidence, adversarial replay trials | Real-time coordinated simulation remains |
| P1 | Metadata stripping/transcoding | Store source receipt, detached recovery link, verify transformed asset separately | New bytes may lose embedded provenance |
| P1 | Canonicalization or MIME confusion | RFC 8785 fixtures, strict parsers, content sniffing in isolation, digest exact bytes | Parser ecosystem differences require compatibility testing |
| P1 | Insider reviewer or operator abuse | Role separation, signed decisions, dual control for policy/key changes, audit trail | Colluding privileged actors remain |
| P2 | Service/log unavailability | Queued local evidence, explicit incomplete state, backups, restore drills | Freshness and status may be unavailable |
| P2 | Algorithm, key or certificate expiry | Crypto agility, key history, trusted timestamps, archival validation material | Ecosystem trust changes over time |
| P2 | Accessibility or device exclusion | Alternative ceremony, assisted path, accessible verifier, human exception review | Higher-evidence hardware may remain unavailable to some users |

# 15. Acceptance and adversarial tests

All thresholds and expected rates must be chosen before a trial and labelled as product decisions. This guide makes no unmeasured efficacy, latency, false-acceptance, false-rejection or cost claim.

## 15.1 Canonicalization and hashing

1. The same event fixture produces identical JCS bytes and SHA-256 digest in browser, server and an independent implementation.
2. Reordered object properties preserve the digest; reordered arrays do not unless the schema defines the array as a sorted set.
3. Duplicate keys, invalid Unicode and non-finite numbers are rejected.
4. Changing one artifact byte fails the digest match.
5. MIME type, byte length and digest disagreement becomes `artifact_digest_mismatch`, not a warning-only state.

## 15.2 WebAuthn and replay

1. Reject wrong origin, RP ID, ceremony type, challenge, credential or signature.
2. Reject missing user presence or user verification when required.
3. Reject expired, consumed, wrong-session, wrong-device and wrong-sequence challenges.
4. Race two submissions using the same challenge; exactly one may consume it.
5. Test a known hardware authenticator against the chosen metadata/status policy.
6. Classify self, indirect, none and unknown attestation without silent elevation.
7. Accept a legitimate zero signature counter according to policy; flag a counter decrease for investigation.

8. Revoke a credential: new events fail, while historical receipts retain explicit key-status-at-time semantics.

## 15.3 Capture and editing

1. Capture both cross-challenges in expected peer streams and reject swapped session IDs.
2. Delete a middle segment; sequence and stream-root validation fails.
3. Truncate the tail; absence of a terminal close yields `incomplete`.
4. Submit inconsistent terminal counts; mark the stream unresolved.
5. Replay a prerecorded screen and document whether cross-challenges expose it.
6. Inject a virtual-camera feed and record the evidence gap if accepted.
7. Use two devices under the same account/admin and confirm the verifier shows the weaker independence class.
8. Change an editor project, ingredient or export byte after checkpoint; the corresponding digest fails.
9. Omit an edit action and verify that the system does not infer completeness from a valid signature.

## 15.4 C2PA, code and documents

1. Validate a supported C2PA 2.4 asset with a configured signer; distinguish untrusted signer from invalid structure.
2. Modify the asset or manifest and observe validation failure.
3. Strip embedded provenance through transcoding and test the detached/source-record path.
4. With `allActionsIncluded` absent or false, prohibit a complete-lineage presentation.
5. Validate ingredient and HumanVerified receipt references against exact digests.
6. Validate in-toto/DSSE subject digests and reject a statement with the wrong subject.
7. Verify SLSA build provenance corresponds to the actual source and builder; reject an unsigned substitute build.
8. Verify this release's Markdown and PDF checksums against exact bytes.
9. Modify PDF content, metadata or permissions and confirm checksum mismatch.
10. Demonstrate that removing permission restrictions or producing a derivative copy does not preserve the trusted checksum.
11. For a future signed fixture, validate PAdES before and after modification and test timestamp/certificate archival material.

## 15.5 Transparency and time

1. Verify inclusion proofs for first, middle and final leaves.
2. Verify consistency proofs across increasing tree sizes.
3. Alter, delete or reorder a leaf and observe proof failure.
4. Present two roots for the same tree size; the independent monitor must detect the conflict.
5. Rotate the log key with an authorized transition record and verify old checkpoints.
6. Manipulate the protected service clock; verify that it cannot replace the external timestamp.

7. Test payload-time and signature-time ordering explicitly.

### 15.6 Privacy and authorization

1. Scan public receipts for names, emails, biometrics, raw prompts, raw video, stable device identifiers, credential IDs and private object paths.
2. Attempt enumeration of record IDs and cross-project pseudonym correlation.
3. Run a dictionary attack against deliberately low-entropy public hash fixtures.
4. Test horizontal and vertical authorization across participants, reviewers and operators.
5. Exercise access, correction, supersession, withdrawal, retention expiry and deletion workflows.
6. Confirm log leaves remain minimal and non-identifying after private evidence deletion under the approved policy.
7. Verify logs and errors do not contain challenge secrets, evidence content or signing material.

### 15.7 Physical craft

1. Copy a QR code and confirm the verifier does not treat it as possession proof.
2. Clone an ordinary NFC UID and record the outcome.
3. Relay or transfer a cryptographic tag and test secondary attachment inspection.
4. Substitute the object while retaining the tag.
5. Stage work and omit an interval; confirm the Evidence Trail exposes gaps rather than asserting continuous custody.
6. Replay captured evidence from a display and record the residual limitation.

### 15.8 Accessibility and comprehension

1. Complete all verifier tasks by keyboard.
2. Validate headings, landmarks, table headers, focus order, accessible names and error association with a screen reader.
3. Confirm every state has text/icon semantics independent of colour.
4. Test zoom, reflow, narrow viewport and reduced-motion settings.
5. Give representative readers signed-valid-but-false, unsigned-but-plausible and incomplete examples. Measure whether they distinguish signature validity, review, person intent and content truth.
6. Test an alternative path for a participant unable to use the selected hardware authenticator.

### 15.9 Recovery

1. Rotate actor, receipt, log, C2PA and future document keys separately.
2. Simulate receipt-key compromise and publish the affected-time/status rule.
3. Restore the relational store, evidence objects and log from backups and verify roots.
4. Operate with the witness or TSA unavailable; new records must show the missing anchor.
5. Recover from a partially uploaded segment without creating a false closed stream.
6. Preserve historical verification after supersession and policy changes.

# 16. Six-week build plan

| Week | Build focus | Exit evidence |
|---|---|---|
| 1 | Claim taxonomy, event/receipt schema, JCS fixtures, data map, threat-model review, DPIA screening | Approved schemas, cross-language digests, visible assumption register, participant-data gate decision |
| 2 | Account binding, WebAuthn enrolment/assertion, nonce/replay service, key-status model | Negative WebAuthn suite passing; hardware/unknown tiers displayed distinctly |
| 3 | Event store, version DAG, sequence rules, terminal records, Merkle log, external checkpoint experiment | Inclusion/consistency fixtures, split-view exercise, restore procedure |
| 4 | Capture/edit adapters, C2PA fixture, in-toto/DSSE and SLSA build fixture, document-release checks, craft tag experiment | At least one valid and one adversarial fixture per adapter; residual limits recorded |
| 5 | Public verifier, status state machine, privacy projection, accessible evidence timeline | End-to-end verification for three synthetic trails; privacy and keyboard tests |
| 6 | Controlled demonstrations, attack exercises, comprehension testing, recovery drill, evidence-gap report | Predeclared measures reported without claim inflation; go/no-go decision for any later pilot |

# 17. Prioritized backlog

## P0 — required for a controlled PoC

1. Freeze event, receipt, error and proof schemas.
1. Build independent RFC 8785 digest fixtures.
1. Implement WebAuthn registration/assertion and atomic replay cache.
1. Implement artifact store, event store, sequence and parent checks.
1. Implement signed acceptance receipts and purpose-separated key IDs.
1. Implement Merkle inclusion/consistency proofs and one external checkpoint method.
1. Implement terminal close records.
1. Build public privacy projection and state machine.
1. Complete data protection and participant-governance gates.
1. Create adversarial fixtures before controlled demonstrations.

## P1 — required for three-track evidence

1. Dual-device app/session and cross-challenge capture.
1. Controlled editor/export adapter.
1. C2PA 2.4 supported-media adapter.
1. in-toto/DSSE and SLSA build adapter.
1. Manual craft QR/NFC and inspection adapter.
1. Correction, supersession, revocation and historical verification UI.
1. Accessibility and comprehension harness.

### P2 — evaluate after evidence gaps are known

1. CAWG identity assertion integration.
1. Additional independent witnesses or public-log interoperability.
1. PAdES B-LT/B-LTA document signatures and document C2PA embedding.
1. Cryptographic NFC variants and attachment sensors.
1. Offline receipt bundles and archival validator packaging.
1. Optional Steam account/distribution experiment with no sensor-trust claim.

## 18. Dependencies and operational ownership

| Dependency | Decision needed | Owner role |
|---|---|---|
| Identity-vetting prerequisite | Provider, level, mapping and correction process | Product/privacy lead |
| WebAuthn fleet | Browsers, authenticators, attestation and recovery policy | Security lead |
| Receipt/log key service | KMS/HSM, rotation, audit and emergency revocation | Security/operations lead |
| External anchor | Witness, monitor or TSA and outage policy | Security/operations lead |
| C2PA tooling | Writable formats, certificate source, trust policy and validator compatibility | Media engineering lead |
| Evidence storage | Region, encryption, versioning, retention and deletion behaviour | Operations/privacy lead |
| Mobile capture | Device support, app signing, platform attestation and update policy | Mobile engineering lead |
| Legal/privacy | Controller, lawful basis, DPIA, participant notice and rights handling | Qualified human reviewer |
| Accessibility | Test standard, assistive-technology matrix and alternative ceremony | Accessibility lead |
| Publication signing | Organisation-controlled PAdES/C2PA identity | everwished innovation |

## 19. Go/no-go gates

### Gate G0 — publication fixture

Proceed only if the Markdown, permission-restricted PDF, manifest and checksums are internally consistent; Annex A is included; forbidden overclaims are absent; and current technical statements are cited.

### Gate G1 — synthetic lab trial

Proceed only if canonicalization, replay, signature, sequence, parent, terminal-record, Merkle and privacy-projection tests pass; key separation is documented; and no production personal data is required.

### Gate G2 — controlled participant trial

Proceed only after qualified human approval of DPIA/lawful-basis work, participant notice, retention/deletion, incident response, accessibility alternatives, device/admin failure-domain classification and operator access controls.

### Gate G3 — public verification pilot

Proceed only if at least one external checkpoint method is operating, current status and revocation are visible, all three adapters expose their limitations, recovery exercises pass, and independent legal/privacy/security/accessibility review actions are resolved or accepted by named owners.

No gate authorizes broader identity, employment, biometric or legal-admissibility claims.

## 20. Assumptions

1. A-01: A six-week controlled PoC has people, devices, budget and participant time available. The product owner must confirm this before kickoff.
2. A-02: Two capture devices can use distinct keys and meaningfully documented failure domains. If they share operator, administrator, app supply chain or cloud account, the verifier must show that dependency.
3. A-03: At least one target media format can be written and validated with the selected C2PA 2.4 toolchain. Format support must be demonstrated with fixtures.
4. A-04: The selected WebAuthn browsers and authenticators expose the required user-verification and attestation behaviour. Assurance tiers change if they do not.
5. A-05: A suitable KMS/HSM, external witness or timestamp provider can be selected for a later pilot. The lab may use isolated test keys but must label them as such.
6. A-06: Participants are consenting adults in a controlled cohort. Children, safeguarding-sensitive work, right-to-work, criminal-record and regulated decision uses are outside this document.
7. A-07: Public opaque identifiers and commitments can be made sufficiently resistant to practical linkage for the approved use. The privacy lead must test this assumption.
8. A-08: everwished innovation has not supplied an organisation-controlled signing identity for this release. The v0.1 release therefore uses permission restrictions and unsigned checksums rather than a document signature.
9. A-09: HumanVerified sites remain static publication surfaces. A live evidence service would use a separate protected origin and expose only privacy-reviewed data.
10. A-10: No named independent human reviewer has recorded approval of this document at publication. Its review state must remain AI-generated and Human-directed.
11. A-11: AI service providers may not provide a cryptographically verifiable receipt for every output. The protocol records exact provider identifiers only when accurately captured.
12. A-12: A physical-to-digital binding cannot be closed by QR/NFC alone. The PoC will report tag and inspection evidence separately.
13. A-13: Public retention of a receipt commitment has a documented purpose and lawful basis even after private evidence deletion. This requires human privacy/legal determination.
14. A-14: Current UK DVS trust-framework status and any provider registration will be rechecked before public use. HumanVerified makes no status claim in this guide.

## 21. Source notes

All web sources below were accessed 3 September 2026. Standards and service programmes may change; implementation must pin the version used and recheck status at each go/no-go gate.

1 C2PA Technical Specification 2.4. Defines signed claims/assertions, actions, ingredients, AI disclosure, repository receipts, supported embedding models and the limits of validation. The specification states that trust in assertions depends on the signer credential and that the framework does not make a value judgement about provenance data. https://spec.c2pa.org/specifications/specifications/2.4/specs/C2PA_Specification.html

2 C2PA specifications index, version 2.4. Identifies the current 2.4 publication set used by this guide. https://spec.c2pa.org/specifications/specifications/2.4/index.html

3 C2PA conformance programme. Describes conformance and official trust-list arrangements. HumanVerified does not claim programme status. https://c2pa.org/conformance/

4 C2PA human and organisational identity guidance. Recommends CAWG identity assertions for human/organisation identity beyond core machine identity. https://spec.c2pa.org/specifications/specifications/2.4/identity/identity.html

5 CAWG Identity Assertion 1.3. Defines identity assertions as binding an actor's control of a digital identity to assertions or an asset; it cautions against treating that as attribution or ownership and documents recontextualization risk. https://cawg.io/identity/1.3/

6 CAI SDK supported formats. The current support table lists PDF reading while specification capability is broader; actual write support is therefore a fixture gate. https://opensource.contentauthenticity.org/docs/c2pa-node/docs/supported-formats/

7 W3C Web Authentication Level 3. Defines RP-scoped public-key credentials, challenge ceremonies, user presence, user verification, backup-state flags, attestation and signature-counter handling. It distinguishes local user verification from concrete identity. https://www.w3.org/TR/webauthn-3/

8 FIDO Metadata Service. Provides authenticator metadata and status information used by an attestation policy. https://fidoalliance.org/metadata/

9 NIST SP 800-63B-4. Distinguishes authenticator properties including non-exportable hardware-backed credentials and syncable authenticators. https://pages.nist.gov/800-63-4/sp800-63b/authenticators/

10 FIDO CTAP 2.2. Defines authenticator protocol behaviour, including user-presence interaction semantics. https://fidoalliance.org/specs/fido-v2.2-rd-20241003/fido-client-to-authenticator-protocol-v2.2-rd-20241003.html

11 Android Key Attestation. Describes validation of hardware-backed key properties and certificate chains. It is used here as platform/key evidence, not scene evidence. https://developer.android.com/privacy-and-security/security-key-attestation

12 Apple DeviceCheck and App Attest. Describes app/device integrity signals and their role in fraud-risk assessment. It is not treated as sensor truth. https://developer.apple.com/documentation/devicecheck

13 RFC 8785, JSON Canonicalization Scheme. Defines deterministic JSON serialization used for event and receipt hashing. https://www.rfc-editor.org/rfc/rfc8785

14 FIPS 180-4, Secure Hash Standard. Defines SHA-256 used for artifact, event and log commitments. https://csrc.nist.gov/pubs/fips/180-4/upd1/final

15 RFC 9162, Certificate Transparency Version 2.0. Defines Merkle inclusion and consistency proof concepts used as the transparency-log model. https://www.rfc-editor.org/rfc/rfc9162

16 Sigstore transparency logging overview and security model. Describe append-only signature transparency and the need for monitoring in the long-term trust model. https://docs.sigstore.dev/logging/overview/ and https://docs.sigstore.dev/about/security/

17 RFC 3161, Time-Stamp Protocol. Defines trusted timestamps over message imprints. https://www.rfc-editor.org/rfc/rfc3161

18 RFC 9921, COSE Header Parameter for RFC 3161 Timestamp Tokens. Defines two ways to combine RFC 3161 timestamp tokens with COSE signatures, including signature-then-timestamp and timestamp-then-signature constructions. https://www.rfc-editor.org/rfc/rfc9921

19 ETSI EN 319 142-1, PAdES digital signatures. Defines PAdES baseline signature levels including validation material and archival timestamps. https://www.etsi.org/deliver/etsiEN/319100319199/31914201/01.02.0160/en31914201v010201p.pdf

20 in-toto Attestation Statement v1. Defines subject digests and typed predicates used for code/process statements. https://github.com/in-toto/attestation/blob/main/spec/v1/statement.md

21 DSSE specification and scope. Defines signatures for typed payloads and notes that identity, confidentiality, key management and payload truth are outside its scope. https://github.com/secure-systems-lab/dsse and https://github.com/secure-systems-lab/dsse/blob/master/governance/02-scope.md

22 SLSA Provenance 1.2 and build track. Define software build provenance and assurance levels; they are not human-authorship specifications. https://slsa.dev/spec/v1.2/provenance and https://slsa.dev/spec/v1.2/build-track-basics

23 ICO pseudonymisation guidance. Explains that pseudonymized information remains personal data for a party holding the additional identifying information. https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-sharing/anonymisation/pseudonymisation/

24 ICO data minimisation guidance. Requires personal data to be adequate, relevant and limited to what is necessary. https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-protection-principles/a-guide-to-the-data-protection-principles/data-minimisation/

25 ICO DPIA guidance. Describes assessment requirements for processing likely to result in high risk. https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/accountability-and-governance/data-protection-impact-assessments-dpias/

26 ICO biometric-recognition compliance guidance. Covers data-protection obligations where biometric recognition is proposed. https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/lawful-basis/biometric-data-guidance-biometric-recognition/how-do-we-demonstrate-our-compliance-with-our-data-protection-obligations/

27 UK digital verification services trust framework v1.0. The GOV.UK collection identifies version 1.0 as published 9 June 2026 and describes commencement as contingent on first conformity-assessment-body accreditation, no earlier than 1 September 2026. This guide makes no claim about HumanVerified or provider status; the source must be rechecked before any public status statement. https://www.gov.uk/government/collections/uk-digital-verification-services-trust-framework

28 GPG45: How to prove and verify someone's identity, version 1.0. Provides current UK government identity-proofing guidance relevant to any selected identity-vetting prerequisite. https://www.gov.uk/government/publications/how-to-check-someones-identity-1-0

29 Register of digital identity and attribute services. The current register is the source to check before stating that a service is registered. https://www.gov.uk/guidance/find-registered-digital-identity-and-attribute-services

# Annex A — AI generation and use disclosure

## A.1 Status

This document was generated using OpenAI Codex under human direction for everwished innovation on 3 September 2026. The AI-generated material includes this annex. “Responsible-AI disclosure” describes the disclosure practice; it is not an external certification or assurance mark.

This document describes a proposed proof of concept. Its publication is not evidence that the proposed system has been implemented, independently audited, or shown to achieve any stated outcome.

## A.2 Human inputs and direction

Human input supplied the problem statement, intended audiences, approved technical thesis, delivery constraints, existing HumanVerified materials, and required claim boundaries. Human direction is not the same as independent human validation or approval.

## A.3 AI contribution

AI systems assisted with research retrieval, architecture development, threat analysis, drafting, editing, diagram specification, consistency review, document generation, and release-quality checks. The source list identifies the external materials relied upon. Tool or model identifiers are stated only where they were captured accurately; none has been inferred.

## A.4 Review state

At publication, no claim should be described as human-reviewed unless a named reviewer has completed and recorded that review. AI-based cross-checking is not independent human review. Legal, privacy, identity-assurance, cryptographic, accessibility, and operational decisions require qualified human assessment before a live pilot.

## A.5 Fact and assumption controls

Externally verifiable claims are cited to primary sources where available. Assumptions, proposals, untested hypotheses, and limitations are identified as such. Citations show the basis for a statement; they do not guarantee that the statement is complete or suitable for a particular implementation.

## A.6 Known limitations

Generative AI can omit context, misstate sources, produce plausible but incorrect details, and conceal uncertainty in fluent language. The proposed evidence system would provide bounded provenance signals, not semantic truth, guaranteed authorship, absence of AI contribution, legal ownership, safety, or resistance to every alteration. Cryptographic validation establishes only the properties defined by the relevant keys, records, trust anchors, and verification policy.

## A.7 Document provenance and integrity

The editable Markdown is the publication source. Each PDF contains a visible document identifier, page count, annex marker, and source-hash prefix. The external release manifest records the complete source and final-PDF hashes. PDF permission controls deter editing in conforming software but can be bypassed by other tools. A changed copy is detectable only when compared with a trusted checksum or signature.

This release is not claimed as PDF/UA conformant. It provides selectable text, embedded fonts, `en-GB` language metadata and permission for accessibility extraction, but it is not a fully tagged PDF. Tagged-PDF remediation and independent accessibility review remain required before a live publication decision.

## A.8 Privacy

The publication records input categories and source references rather than a private prompt transcript. It must not contain identity documents, raw participant evidence, secrets, credentials, device identifiers, private locations, or signing material. Public ledgers should contain only privacy-reviewed opaque identifiers and cryptographic commitments; sensitive evidence belongs in access-controlled, retention-limited storage.

## A.9 Change control

This package is an unpublished release candidate until the owner deliberately publishes it and its checksum manifest through a trusted channel. Candidate rebuilds in this workspace may replace local v0.1 files and necessarily create new hashes; the builder requires an explicit release-candidate replacement flag when prior output exists.

After a package has been distributed, cited or otherwise relied upon, any byte change must use a new version or build identifier and preserve the earlier package and manifest. Publication must use a versioned, package-atomic deployment or equivalent rollback-controlled process. The local document builder uses atomic replacement per file but does not claim package-level atomic deployment.