Universal Creation Evidence (UCE)

Evidentiary Standards & Verification Framework v1.0

Version: 1.0 Public Draft
Date: May 2026
Framework Name: Universal Creation Evidence (UCE)
Prepared by: Copyright by UCE (CbyUCE)
Document Type: Technical Evidentiary Framework / Public Specification Draft

---

Publication Notice

Universal Creation Evidence (UCE) is an implementation-independent evidentiary framework for preserving and verifying technical evidence associated with creative works. Copyright by UCE (CbyUCE) is one operational implementation and reference architecture for the UCE framework.

This document is intended to support public understanding, third-party review, technical implementation, legal-tech interoperability, and standards-oriented discussion. It does not constitute legal advice, a judicial determination, a government registration system, or a guarantee of copyright ownership.

Table of Contents

1. Executive Summary
2. Purpose, Scope, and Non-Goals
3. Normative Language and Document Status
4. Definitions
5. Framework Architecture
6. Architectural Principles
7. Core Evidence Components
8. Manifest Data Model
9. Canonicalization Requirements
10. Cryptographic Requirements
11. Cryptographic Agility and Algorithm Migration
12. Timestamping and Chronology Model
13. Verification Procedures

14. Verification Outcomes and Evidence Weight
15. Rights Declarations
16. Public Verification Infrastructure
17. Chain-of-Custody Considerations
18. Identity Assertions and Identity Attestation
19. Threat Model
20. Privacy, Security, and Operational Controls
21. Interoperability and Open Standards
22. Conformance Requirements
23. Governance, Versioning, and Evolution
24. Limitations
25. Recommended Best Practices
26. Evidentiary Positioning and Public Communications
27. Potential Evidentiary Use Cases
28. Conclusion
29. Appendix A — Reference Manifest Structure
30. Appendix B — Simplified Public Certificate Fields
31. Appendix C — Public Verification Checklist
32. Appendix D — Recommended Public Language
33. Appendix E — Suggested Future Research Areas
34. Appendix F — Reference Standards and Technical Materials

---

1. Executive Summary

Universal Creation Evidence (UCE) is a cryptographically verifiable evidentiary framework designed to preserve and independently verify technical evidence concerning the existence, integrity, asserted authorship, chronology, provenance, and rights declarations associated with creative works.

UCE is not a copyright registration system and does not replace registration with the United States Copyright Office or any national copyright authority. Instead, UCE functions as an evidence-layer architecture intended to support creators, counsel, adjudicators, investigators, licensing entities, platforms, educational institutions, archives, and commercial counterparties by preserving technical evidence relevant to questions of chronology, integrity, attribution, and provenance.

A UCE evidence record is designed to answer limited but important technical questions:

- Did a specific digital file, matching a specific cryptographic hash, exist no later than a verifiable timestamp?
- Does the file presented for verification match the file identified in the evidence manifest?
- Does the evidence manifest remain unchanged from the manifest that was originally recorded?

- Was the manifest signed by a recognized signing key associated with the implementation that created the evidence record?
- What creator, claimant, metadata, rights declarations, and storage references were asserted at the time the evidence record was created?

UCE does not, by itself, answer broader legal or factual questions such as whether the claimant is the true legal owner, whether the work infringes another work, whether competing claims exist, or how an adjudicator should weigh the evidence in a particular dispute.

The framework combines:

- Cryptographic hashing
- Deterministic manifest canonicalization
- Public-key digital signatures
- Immutable or durable decentralized storage
- Public verification infrastructure
- Multi-gateway retrieval and redundancy
- Human-readable evidence certificates
- Machine-readable rights declarations
- Implementation-independent verification procedures

The intended result is a verifiable evidence package capable of independent audit by third parties without requiring reliance on a centralized proprietary database.

---

2. Purpose, Scope, and Non-Goals

2.1 Purpose

The purpose of UCE is to create a standardized evidentiary methodology for documenting and verifying:

- The existence of a digital work
- The integrity of a digital work
- The asserted identity of a creator, claimant, or submitting party
- The chronology of submission, publication, storage, or evidence creation
- Rights declarations associated with the work
- The relationship between a work, its evidence manifest, and its storage references
- The verifiability of an evidence package through open technical procedures

2.2 Scope

UCE is designed to support evidentiary and operational workflows involving:

- Creative authorship disputes

- Copyright chronology evidence
- U.S. Copyright Claims Board proceedings
- DMCA-related evidence preservation
- AI training opt-out declarations
- Licensing verification
- Chain-of-custody documentation
- Catalog management
- Provenance verification
- Platform moderation review
- Archive preservation
- Educational institution evidence programs
- Entertainment-industry asset management
- Commercial due diligence
- Rights-management and licensing ecosystems

2.3 Non-Goals

UCE is not intended to:

- Determine legal ownership conclusively
- Replace judicial or administrative findings
- Replace statutory registration systems
- Grant copyright protection
- Transfer or assign rights
- Adjudicate competing factual claims
- Determine whether a work is original
- Determine whether a work infringes another work
- Function as digital rights management or copy protection
- Guarantee enforceability of rights declarations
- Guarantee that no earlier work or competing claimant exists

2.4 Relationship Between UCE and CbyUCE

UCE is the framework and evidentiary methodology. CbyUCE is an implementation of that framework.

The validity of a conforming UCE evidence package should not depend solely on the continued operation of CbyUCE or any other single commercial entity. A conforming implementation should produce evidence records that can be independently verified using publicly documented procedures, open cryptographic standards, and accessible storage references.

---

3. Normative Language and Document Status

3.1 Normative Terms

The key words **MUST**, **MUST NOT**, **REQUIRED**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **MAY**, and **OPTIONAL** are to be interpreted in the standards-document sense.

- **MUST / REQUIRED** indicates a mandatory requirement for conformance.
- **MUST NOT** indicates a prohibited behavior for conforming implementations.
- **SHOULD / RECOMMENDED** indicates a strong recommendation, where valid reasons may exist to vary from the recommendation, but the implications should be understood and documented.
- **MAY / OPTIONAL** indicates a permissible implementation choice.

3.2 Normative and Informative Material

Sections that define required data structures, verification steps, cryptographic requirements, canonicalization requirements, conformance requirements, or versioning rules are normative unless expressly identified as illustrative or informative.

Examples, implementation notes, explanatory commentary, and simplified public communications guidance are informative unless expressly incorporated into conformance requirements.

3.3 Public Draft Status

This version is a public draft intended for review, implementation feedback, legal-tech discussion, and interoperability planning. Future revisions may refine schema definitions, verification requirements, governance processes, rights declarations, identity attestations, and cryptographic migration paths.

---

4. Definitions

Term	Definition
UCE	Universal Creation Evidence, an evidentiary framework for preserving and verifying technical evidence associated with creative works.
UCE Evidence Package	The complete set of evidence components associated with a work, including file hash, manifest, signature, storage references, verification references, and certificate data.

Term	Definition
Evidence Manifest	A structured machine-readable document containing hashes, metadata, signatures, storage references, rights declarations, and verification information associated with a work.
Content Hash	A cryptographic digest of the submitted file bytes, typically using SHA-256 in UCE v1.
Manifest Hash	A cryptographic digest of the canonicalized evidence manifest.
Canonicalization	A deterministic serialization process used before hashing structured data.
Verification URL	A public web endpoint associated with a manifest or manifest hash that provides a human-readable verification interface.
Public Key Reference	A reference to a public verification key used to validate a digital signature.
Immutable Storage Layer	A storage infrastructure designed to preserve data durably and resist later alteration, such as Arweave Layer 1 storage.
Storage Anchor	A transaction ID, content address, URI, or other durable reference identifying where evidence data is stored.
Rights Declaration	A machine-readable expression of rights, restrictions, permissions, or policy assertions associated with the work.
Gateway	A network endpoint used to retrieve decentralized or content-addressed data.
Verifier	A person, organization, system, or software process that evaluates a UCE evidence package.
Claimant	The person or entity asserting authorship, ownership, control, or rights in relation to a work.
Creator Assertion	Metadata supplied by or on behalf of a claimant concerning authorship, title, creation date, ownership, rights, or related facts.
Reference Implementation	A publicly documented implementation intended to demonstrate or test conformance with the framework.

Term	Definition
Conforming Implementation	An implementation that satisfies the mandatory requirements of this framework version.

---

5. Framework Architecture

5.1 Architectural Overview

A UCE evidence package connects four categories of information:

1. **The work** — the original file or files submitted for evidence preservation.
2. **The evidence manifest** — a canonicalized machine-readable record describing the work, hashes, metadata, rights declarations, signatures, and storage anchors.
3. **The cryptographic layer** — hashing, canonicalization, and digital signatures that allow third parties to detect tampering and verify integrity.
4. **The verification layer** — public procedures and interfaces that allow independent review of the evidence package.

5.2 Implementation Independence

A conforming UCE implementation MAY be operated by CbyUCE or by another party. The framework is intended to permit independent implementation, provided that the implementation satisfies conformance requirements and publicly documents its verification procedures.

5.3 Storage-Layer Independence

UCE v1 uses Arweave as the current reference storage architecture for immutable evidence anchoring. However, the UCE framework is intended to be storage-layer aware but not permanently storage-layer captive. Future conforming implementations MAY support additional decentralized, content-addressed, archival, or ledger-based storage systems, provided they preserve independent verifiability, durability, and tamper evidence.

5.4 Reference Architecture

The current reference architecture includes:

- SHA-256 file hashing
- RFC 8785 JSON canonicalization
- SHA-256 manifest hashing
- ES256 digital signatures
- Public verification keys

- Arweave Layer 1 or Arweave-compatible durable storage anchoring
- Multi-gateway data retrieval
- Public verification URLs
- Human-readable certificates
- Machine-readable rights declarations

---

6. Architectural Principles

UCE is built around the following principles.

6.1 Verifiability

Critical evidence elements MUST be independently verifiable using publicly documented cryptographic methods.

6.2 Transparency

Verification procedures SHOULD be public, reproducible, and sufficiently documented to allow competent third parties to audit evidence packages without relying on private databases.

6.3 Vendor Independence

The evidentiary value of a UCE evidence package SHOULD NOT depend on the ongoing operation of a single commercial platform, hosted web interface, proprietary database, or closed verifier.

6.4 Tamper Evidence

Modification of protected evidence components MUST produce detectable cryptographic inconsistencies.

6.5 Decentralized Durability

Evidence records SHOULD persist independently of any single hosting provider.

6.6 Standards Compatibility

The framework SHOULD use broadly accepted cryptographic, internet, serialization, and identity standards whenever feasible.

6.7 Minimal Legal Overclaiming

UCE implementations MUST avoid representing UCE evidence as conclusive proof of ownership, government copyright registration, or legal adjudication.

6.8 Human and Machine Readability

UCE evidence should support both human-readable review and machine-readable processing.

6.9 Cryptographic Agility

UCE implementations SHOULD be capable of migrating to stronger or different cryptographic algorithms over time while preserving backward verification of historical evidence records where reasonably possible.

---

7. Core Evidence Components

A standard UCE evidence package includes the following components.

7.1 Original Work

The digital file or files submitted by the creator, claimant, or authorized representative.

Examples include:

- Audio recordings
- Images
- Video files
- Documents
- Source code
- Literary works
- Design files
- 3D assets
- Project archives
- Mixed-media works

7.2 Content Hash

The submitted work MUST be hashed using a declared cryptographic hash algorithm.

For UCE v1, the reference hash algorithm is:

- SHA-256

The content hash is computed over the exact submitted file bytes.

Properties:

- Deterministic
- Collision-resistant under current cryptographic assumptions
- Sensitive to byte-level changes
- Independently reproducible

A change to even a single byte of the submitted file should produce a different content hash.

7.3 Evidence Manifest

The evidence manifest is a structured machine-readable document containing evidence information associated with the work.

A UCE manifest SHOULD include:

- Schema identifier
- Schema version
- Work metadata
- Creator or claimant assertions
- File metadata
- Content hashes
- Timestamp references
- Rights declarations
- Storage anchors
- Public key references
- Digital signature references
- Verification references
- Implementation metadata
- Canonicalization method
- Hash and signature algorithm identifiers

7.4 Manifest Hash

The manifest MUST be canonicalized before hashing.

The manifest hash binds the following into a single tamper-evident digest:

- Work metadata
- File references
- Content hashes
- Rights declarations
- Storage anchors
- Public key references
- Signature references
- Verification references

- Structural organization

7.5 Digital Signature

The manifest hash, or a defined signing payload derived from the manifest, MUST be digitally signed using a declared asymmetric signature algorithm.

Current reference implementation:

- ES256, meaning ECDSA using the P-256 curve and SHA-256

Future-compatible pathways may include:

- ML-DSA
- Hybrid classical/post-quantum signatures
- Multi-signature workflows
- Institutional co-signatures

7.6 Public Verification Key

The corresponding public verification key MUST be made available to verifiers. The public key SHOULD be available through a durable reference, public key set, decentralized storage anchor, or other documented verification mechanism.

A public key reference SHOULD include:

- Key identifier
- Algorithm
- Key type
- Public key material or durable reference
- Publication or activation date
- Revocation or deprecation status, if applicable

7.7 Storage Anchor

Evidence records SHOULD be anchored to durable, immutable, decentralized, content-addressed, archival, or ledger-based infrastructure.

The current reference architecture uses:

- Arweave Layer 1 storage

A conforming implementation MUST document the storage layer used and explain how verifiers can retrieve and validate the evidence data.

7.8 Human-Readable Certificate

A UCE evidence package MAY include a human-readable certificate summarizing:

- Work title
- Claimed author or creator
- File name
- File hash
- Manifest hash
- Timestamp references
- Verification URL
- Storage references
- Rights declarations
- Framework and implementation version

The certificate is a summary of the evidence package. The certificate itself SHOULD NOT be treated as the sole source of evidentiary truth if the manifest and storage anchors are available.

---

8. Manifest Data Model

8.1 General Requirements

A conforming UCE manifest MUST be machine-readable and deterministically canonicalizable.

The manifest SHOULD be encoded as JSON unless a future version specifies additional supported formats.

A conforming manifest MUST identify:

- The schema name
- The schema version
- The hash algorithm used for file hashes
- The canonicalization method used for manifest hashing
- The signature algorithm used
- The evidence storage references
- The verification procedure or verification URL

8.2 Required Manifest Categories

A UCE v1 manifest SHOULD contain the following top-level categories:

- `schema`
- `schemaVersion`
- `framework`
- `implementation`
- `work`

- `claimant`
- `files`
- `rights`
- `timestamps`
- `storage`
- `hashes`
- `signatures`
- `verification`

A minimal conforming manifest MAY omit optional fields, but it MUST include enough information for independent verification of file hash, manifest hash, signature, and storage references.

8.3 Creator and Claimant Assertions

Creator, claimant, and rights metadata are assertions recorded by or on behalf of the submitting party. A UCE implementation MUST NOT describe such assertions as independently verified unless the implementation actually performs and documents a corresponding identity or rights attestation process.

8.4 Rights Declaration Fields

Rights declarations SHOULD be clearly identified as assertions, policy expressions, or permissions statements. They SHOULD NOT be presented as self-executing enforcement mechanisms.

8.5 Implementation Metadata

Implementation metadata SHOULD identify the software, platform, or service that generated the manifest. However, implementation metadata SHOULD NOT make the evidence package dependent on a proprietary database for verification.

---

9. Canonicalization Requirements

9.1 Purpose

Canonicalization is required to ensure deterministic hashing of structured manifest data across environments.

Without canonicalization, two semantically identical JSON documents could produce different hash values because of differences in property ordering, whitespace, encoding, or serialization behavior.

9.2 Current Implementation Standard

The current UCE v1 canonicalization method is:

- RFC 8785 JSON Canonicalization Scheme (JCS)

9.3 Canonicalization Requirements

A conforming UCE v1 implementation MUST:

- Use deterministic property ordering
- Use UTF-8 encoding
- Remove serialization ambiguity
- Preserve stable number, string, array, and object representation according to the declared canonicalization method
- Compute the manifest hash over the canonicalized representation
- Record the canonicalization method in the manifest

9.4 Canonicalization Errors

If a verifier cannot canonicalize a manifest according to the declared method, verification MUST fail or return an indeterminate result with a clear explanation.

---

10. Cryptographic Requirements

10.1 Hashing Requirements

A conforming UCE v1 implementation MUST compute a content hash over the exact bytes of each submitted file.

The current reference algorithm is SHA-256.

Each file entry SHOULD include:

- File name
- File size in bytes
- MIME type or content type, if known
- SHA-256 hash
- Hash algorithm identifier

10.2 Manifest Hashing Requirements

A conforming implementation MUST compute the manifest hash over the canonicalized manifest.

The manifest SHOULD include:

- Manifest hash
- Hash algorithm
- Canonicalization method

If the manifest hash is embedded inside the manifest, the implementation MUST define the pre-hash representation unambiguously to avoid circularity. A common approach is to compute the manifest hash over a manifest representation that excludes the final manifest hash field or uses a defined placeholder.

10.3 Signature Requirements

A conforming implementation MUST digitally sign a defined signing payload associated with the manifest.

The signing payload SHOULD be one of:

- The manifest hash
- A detached JWS payload containing the manifest hash and metadata
- A canonicalized signing object derived from the manifest

The signature object SHOULD identify:

- Signature algorithm
- Key identifier
- Public key reference
- Signing time, if available
- Signature value
- Signature encoding

10.4 Key Publication Requirements

Public verification keys MUST be available to verifiers.

Implementations SHOULD maintain:

- Public key history
- Activation timestamps
- Key rotation records
- Revocation or compromise notices, if applicable

10.5 Key Rotation

Implementations SHOULD support key rotation without invalidating prior evidence records. Historical evidence records SHOULD remain verifiable using the public key that was valid at the time of signing.

10.6 Key Compromise

Implementations SHOULD maintain a documented key compromise response process, including:

- Prompt disclosure of affected key identifiers
- Publication of revocation or compromise status
- Preservation of historical verification data
- Clear distinction between pre-compromise and post-compromise signatures
- Migration to new signing keys

---

11. Cryptographic Agility and Algorithm Migration

11.1 Principle

UCE evidence packages are intended to remain useful over long time horizons. Cryptographic algorithms that are considered secure at one point in time may become weakened or obsolete in the future.

For that reason, UCE implementations SHOULD be cryptographically agile.

11.2 Algorithm Identification

UCE manifests MUST explicitly identify algorithms used for:

- File hashing
- Manifest hashing
- Canonicalization
- Digital signatures
- Optional encryption
- Optional identity attestation

11.3 Migration Strategy

Future UCE versions MAY introduce:

- Additional hash algorithms
- Stronger signature schemes
- Hybrid signature schemes

- Post-quantum signature schemes
- Multi-signature attestations
- Cross-storage anchoring
- Re-anchoring or renewal records

11.4 Backward Verification

Newer implementations SHOULD preserve the ability to verify older evidence packages according to the algorithms and procedures declared at the time of evidence creation, unless a specific algorithm is later determined to be critically unsafe.

11.5 Renewal and Re-Attestation

A future UCE revision MAY define renewal or re-attestation workflows, allowing historical evidence records to be re-hashed, cross-signed, or re-anchored using newer algorithms without altering the original evidence record.

---

12. Timestamping and Chronology Model

12.1 Timestamp Sources

UCE timestamps may derive from multiple sources, including:

- Manifest creation timestamp
- Transaction submission timestamp
- Decentralized ledger block timestamp
- Storage inclusion timestamp
- Verification timestamp
- Optional external timestamping authority records

12.2 Ledger Inclusion Chronology

The strongest chronology assertion in a UCE evidence package generally derives from durable storage or ledger inclusion.

UCE establishes evidence that a specific file matching a specific hash existed no later than the relevant storage or ledger timestamp associated with the evidence record.

12.3 Chronology Assertions

UCE may support the following assertion:

A file matching the recorded content hash existed no later than the relevant verified storage, ledger, or inclusion timestamp associated with the evidence package.

12.4 Limits of Chronology

UCE does not assert:

- That no earlier version existed elsewhere
- That no competing claimant exists
- That chronology alone proves ownership
- That the recorded creation date is independently true
- That the claimant was the first creator
- That the work is original or non-infringing

12.5 Timestamp Conflicts

Where multiple timestamps exist, verifiers SHOULD distinguish between:

- User-supplied creation date
- System-generated manifest creation time
- Transaction submission time
- Confirmed ledger or storage inclusion time
- Time of verification

User-supplied timestamps SHOULD be treated as assertions unless independently attested.

---

13. Verification Procedures

13.1 Standard Verification Workflow

A verifier SHOULD perform the following steps:

1. Obtain the original file or candidate file.
2. Compute the file's SHA-256 hash or declared content hash.
3. Retrieve the evidence manifest from its storage anchor.
4. Confirm the file hash appears in the manifest.
5. Canonicalize the manifest according to the declared canonicalization method.
6. Compute the manifest hash according to the declared hash algorithm.
7. Compare the computed manifest hash to the recorded manifest hash.
8. Validate the digital signature against the signed payload.
9. Retrieve or validate the public key reference.
10. Confirm that the public key was valid for the relevant signing context.
11. Confirm decentralized or durable storage references resolve correctly.

12. Confirm timestamp references, including storage or ledger inclusion where available.
13. Review creator, claimant, work, and rights metadata as assertions.
14. Record the verification result, date, verifier, and any exceptions.

13.2 Verification Inputs

A verifier may use:

- The original file
- A UCE certificate
- The evidence manifest
- Storage transaction IDs
- Manifest hash
- Verification URL
- Public key reference
- Public gateway responses
- GraphQL or ledger queries
- Third-party explorer data

13.3 Verification Status Categories

A verification system SHOULD distinguish at least the following outcomes:

Status	Meaning
Verified	The file, manifest, hash, signature, and storage references are consistent.
File Mismatch	The submitted file hash does not match the manifest entry.
Manifest Mismatch	The manifest hash does not match the canonicalized manifest.
Signature Invalid	The digital signature cannot be validated against the referenced public key.
Key Unavailable	The public key reference cannot be retrieved or validated.
Storage Unavailable	The storage reference cannot currently be retrieved.
Indeterminate	Verification cannot be completed because of insufficient data or unresolved errors.
Assertion Review Required	Technical verification succeeded, but identity, ownership, or rights assertions require independent review.

13.4 Verification Interface

A human-readable verification interface MAY summarize verification results, but it SHOULD NOT obscure the underlying cryptographic status. Where feasible, a verification interface SHOULD expose:

- File hash
- Manifest hash
- Signature status
- Public key reference
- Storage references
- Timestamp references
- Rights declarations
- Verification date
- Framework version

13.5 Reference Verification Implementation

A reference verifier MAY be maintained to support interoperability testing and public review. Evidentiary validity SHOULD NOT depend on a single verifier, hosted website, or proprietary database.

Independent implementations SHOULD be able to verify UCE evidence packages using the procedures and data structures documented in this framework.

---

14. Verification Outcomes and Evidence Weight

14.1 What Verification May Establish

Verification may establish:

- File integrity consistency
- Manifest integrity consistency
- Signature validity
- Public key consistency
- Timestamp consistency
- Storage retrieval consistency
- Relationship between a file and a manifest
- Relationship between a manifest and a public verification record
- Recorded rights declarations and creator assertions

14.2 What Verification Does Not Independently Prove

Verification does not independently prove:

- Legal ownership
- Truthfulness of identity assertions
- Exclusivity of authorship
- Nonexistence of prior creators
- Non-infringement
- Validity of a license
- Enforceability of an AI opt-out declaration
- Absence of fraud or coercion

14.3 Evidence Weight

The probative value, admissibility, and evidentiary weight of a UCE evidence package depend on surrounding factual circumstances, corroborating evidence, applicable procedural rules, jurisdiction-specific standards, and adjudicator discretion.

UCE is designed to preserve technical facts that may be relevant to an evidentiary inquiry. It does not determine how those facts will be weighed in a legal, administrative, commercial, or platform-specific proceeding.

---

15. Rights Declarations

15.1 Purpose

UCE supports machine-readable rights declarations so that creators, platforms, licensing entities, AI systems, archives, and verification tools can identify asserted rights preferences or restrictions associated with a work.

Examples include:

- AI training prohibited
- Commercial use restricted
- Attribution required
- Licensing contact required
- Usage permitted under stated conditions
- No derivative works declared
- Rights inquiry route specified

15.2 Declaration Status

Rights declarations are evidentiary assertions and policy expressions. They are not self-executing enforcement mechanisms.

Machine-readable declarations do not, by themselves, create or guarantee enforceable contractual obligations absent applicable law, platform rules, license terms, or separate agreements.

15.3 Example Rights Tags

Illustrative examples:

```
AI-Training-Prohibited: true
UCE-AI-Training-Prohibited: true
Commercial-License-Required: true
Attribution-Required: true
Rights-Contact-Required: true
```

15.4 Rights Declaration Requirements

Rights declarations SHOULD be:

- Machine-readable
- Human-understandable
- Clearly labeled as assertions
- Linked to the relevant evidence manifest
- Versioned when schema changes occur
- Interoperable where possible with external rights-management systems

15.5 Mutable Rights Routing

Some rights-related information, such as contact routing, may need to change over time. Implementations SHOULD distinguish between:

- Immutable evidence declarations recorded at evidence creation
- Mutable contact or licensing routes maintained for operational convenience

Mutable rights-routing data SHOULD NOT silently alter the historical evidence manifest.

---

16. Public Verification Infrastructure

16.1 Redundancy and Decentralization

UCE verification architecture SHOULD support retrieval redundancy and avoid dependence on a single gateway, website, storage host, or explorer.

16.2 Multi-Gateway Retrieval

Verification systems SHOULD support retrieval across multiple gateways where the storage layer permits it.

Example Arweave-compatible retrieval infrastructure may include:

- `arweave.net`
- `permagate.io`
- `ar-io.dev`
- `g8way.io`
- `ardrive.net`

The specific gateway list MAY evolve over time.

16.3 Failure Tolerance

Verification systems SHOULD retry across gateways on:

- 404 responses
- 5xx responses
- Timeout failures
- Indexing inconsistencies
- Temporary network failures
- Gateway-specific caching delays

A temporary retrieval failure from one gateway SHOULD NOT automatically be treated as evidence invalidity if other valid retrieval paths exist.

16.4 Public Verification URLs

Human-readable verification URLs may supplement direct storage references.

Example:

```
https://cbyuce.com/verify/<manifestHash>
```

Such URLs are convenience interfaces. The underlying evidentiary value derives from cryptographic verification, storage references, and independently reproducible verification procedures, not from the web interface alone.

16.5 Third-Party Explorers

Third-party explorers MAY provide useful human-readable confirmation of storage transactions. They SHOULD be treated as supplementary convenience tools, not as the sole authority for verification.

---

17. Chain-of-Custody Considerations

17.1 Cryptographic Evidence Layer

The primary UCE evidence layer consists of content hashes, canonicalized manifests, signatures, storage anchors, and verification procedures.

17.2 Operational Supplements

UCE implementations MAY support additional chain-of-custody elements, including:

- Upload timestamps
- Account identifiers
- Payment records
- Audit trails
- Access logs
- Verification logs
- Version histories
- Administrative actions
- IP logs where legally appropriate
- User consent records
- Institutional cohort records

17.3 Treatment of Operational Logs

Operational logs may strengthen evidentiary context but SHOULD be distinguished from cryptographic evidence. Logs may be subject to retention limits, privacy laws, platform policies, authentication issues, or operational errors.

17.4 Preservation

Organizations relying on UCE evidence SHOULD preserve relevant operational records according to their legal, compliance, privacy, and records-retention obligations.

---

18. Identity Assertions and Identity Attestation

18.1 Asserted Identity

UCE records asserted identity information supplied by or on behalf of a claimant. Unless otherwise stated and independently documented, UCE does not inherently validate legal identity.

18.2 Identity Claims

Identity-related manifest fields SHOULD be treated as claimant assertions unless supported by additional attestation.

Examples of identity-related assertions include:

- Author name
- Legal name
- Organization name
- Account identifier
- Email address
- Wallet address
- Institutional affiliation
- Representative capacity

18.3 External Attestation

Future UCE implementations MAY support external identity attestations, including:

- Institutional identity verification
- Government-issued credential verification
- Decentralized identifiers
- Verifiable credentials
- KYC or KYB checks
- Organization-controlled signing keys
- Multi-party attestation workflows

18.4 Separation of Technical Verification and Identity Truth

A technically verified UCE record means the cryptographic and manifest relationships are consistent. It does not necessarily mean the claimant's identity assertion is true.

Verification interfaces SHOULD make this distinction clear.

---

19. Threat Model

19.1 Threats UCE Is Designed to Mitigate

UCE is designed to mitigate or expose:

- Silent file tampering
- Manifest alteration
- Metadata substitution
- Undetectable file replacement

- Post-publication timestamp manipulation
- Centralized database alteration
- Single-host dependency
- Gateway retrieval inconsistency
- Later denial that a recorded file existed
- Later denial that a specific rights declaration was recorded

19.2 Threats UCE Does Not Eliminate

UCE does not eliminate risks associated with:

- False creator assertions
- Fraudulent uploads
- Credential compromise
- Signing-key compromise
- Social engineering
- Prior undiscovered works
- Competing claimants
- Off-chain coercion
- Misrepresentation of legal rights
- Unauthorized upload of another person’s work
- Loss of private account credentials
- Incomplete operational logs

19.3 Key Compromise

A compromised signing key may undermine trust in records signed after compromise. Implementations SHOULD maintain key histories, activation periods, revocation indicators, and compromise disclosure procedures.

A key compromise does not automatically invalidate all prior evidence records if it can be shown that the signatures were created before compromise and that the relevant public key was valid at the time of signing. However, evidentiary weight may depend on surrounding facts.

19.4 Gateway Censorship or Selective Availability

A gateway may fail to serve valid data because of indexing delays, caching state, outage, policy filtering, censorship, or technical failure. Verification systems SHOULD use multiple retrieval paths and SHOULD distinguish storage unavailability from cryptographic invalidity.

19.5 Replay and Duplication

A malicious or careless actor may submit duplicate or copied works. UCE can show that a particular file was recorded at a particular time by a particular claimant or implementation. It does not prevent others from submitting the same or similar content.

19.6 Hash Collision Risk

UCE v1 relies on SHA-256 collision resistance. Current cryptographic practice treats SHA-256 as suitable for this use. Future revisions SHOULD address migration if practical risk conditions change.

19.7 Quantum and Future Cryptographic Risks

Future advances in cryptanalysis or quantum computing may affect digital signatures and other cryptographic assumptions. UCE should support algorithm migration and renewal strategies.

---

20. Privacy, Security, and Operational Controls

20.1 Public Evidence Records

Implementations MUST clearly disclose whether evidence manifests, file data, metadata, or rights declarations are publicly accessible.

20.2 Sensitive Metadata

Creators and organizations SHOULD avoid placing unnecessary sensitive personal information into immutable public records.

20.3 Data Minimization

Implementations SHOULD use data minimization principles when designing manifests, certificates, and public verification interfaces.

20.4 Access-Controlled Works

If an implementation supports private or restricted files, the manifest SHOULD still enable verification of hashes and evidence metadata without unnecessarily exposing confidential content.

20.5 Operational Security

Implementations SHOULD maintain appropriate security controls, including:

- Secure key management
- Access control
- Audit logging
- Incident response procedures
- Backup and recovery procedures

- Monitoring and alerting
- Secure deployment practices
- Change management

20.6 Privacy Law Considerations

Because immutable storage may conflict with deletion or correction expectations under some privacy regimes, implementations SHOULD carefully consider what information is permanently recorded.

---

21. Interoperability and Open Standards

21.1 Interoperability Objectives

The UCE framework is intended to support interoperability through:

- Publicly documented verification methods
- Portable manifest formats
- Open cryptographic standards
- Vendor-independent validation
- Third-party implementation compatibility
- Machine-readable rights declarations
- Durable storage references

21.2 Standards Compatibility

UCE v1 aligns with or may reference the following types of standards and technical materials:

- JSON canonicalization standards
- Cryptographic hash standards
- Public-key signature standards
- JSON Web Signature and JSON Web Key standards
- Decentralized storage protocols
- Content-addressing models
- Provenance and authenticity frameworks
- Rights metadata systems

21.3 Long-Term Interoperability Goals

Long-term goals may include interoperability with:

- Legal-tech platforms
- Copyright registration support tools

- Licensing platforms
- AI provenance systems
- Platform moderation systems
- Archival repositories
- Educational institutions
- Media authenticity systems
- Rights-management ecosystems
- Institutional identity systems

---

22. Conformance Requirements

22.1 Conforming UCE v1 Evidence Package

A UCE v1 evidence package conforms to this framework if it satisfies the following minimum requirements:

1. It includes a machine-readable evidence manifest.
2. It identifies the schema and schema version.
3. It computes a cryptographic content hash for each protected file.
4. It identifies the hash algorithm used.
5. It canonicalizes the manifest using a declared canonicalization method.
6. It computes a manifest hash over the canonicalized manifest or a clearly defined pre-hash representation.
7. It digitally signs the manifest hash or an equivalent defined signing payload.
8. It identifies the signature algorithm used.
9. It provides a public key reference sufficient for verification.
10. It provides storage references sufficient for retrieval or audit.
11. It identifies relevant timestamp references.
12. It distinguishes technical verification from legal ownership conclusions.
13. It does not represent the evidence package as government copyright registration.

22.2 Conforming UCE v1 Verifier

A UCE v1 verifier conforms to this framework if it can:

1. Accept or retrieve a UCE manifest.
2. Canonicalize the manifest according to the declared method.
3. Compute and compare the manifest hash.
4. Compute and compare file hashes when candidate files are provided.
5. Validate the digital signature against the referenced public key.
6. Retrieve or assess storage references.
7. Report verification outcomes clearly.
8. Distinguish technical verification failures from identity, ownership, and legal questions.

22.3 Conforming Public Communications

Public communications associated with UCE SHOULD avoid overclaiming. They SHOULD describe UCE as:

- Cryptographically verifiable evidence
- Tamper-evident evidence record
- Independent verification framework
- Evidence of asserted chronology
- Evidence-layer infrastructure

They SHOULD NOT describe UCE as:

- Guaranteed ownership
- Government copyright registration
- Conclusive proof
- Impossible to dispute
- Unforgeable proof
- Replacement for judicial findings

---

23. Governance, Versioning, and Evolution

23.1 Framework Evolution

The framework may evolve over time to address new technical, legal, institutional, and interoperability requirements.

Future revisions may address:

- Post-quantum cryptography
- Expanded rights declarations
- Additional canonicalization methods
- Identity attestation systems
- Multi-signature workflows
- Institutional attestations
- Cross-storage anchoring
- Extended audit structures
- Formal conformance test suites
- External standards-body engagement

23.2 Versioning Policy

UCE SHOULD use semantic versioning principles where feasible:

- Major version changes indicate breaking changes to core verification requirements.
- Minor version changes indicate backward-compatible additions.
- Patch version changes indicate clarifications, corrections, or non-breaking refinements.

23.3 Backward Compatibility

Backward compatibility SHOULD be preserved where reasonably possible. Historical evidence records SHOULD remain verifiable according to the framework version and algorithms declared at the time of creation.

23.4 Schema Evolution

Schema changes SHOULD be documented. Deprecated fields SHOULD be identified before removal where feasible. Implementations SHOULD avoid silent interpretation changes that would alter the meaning of historical manifests.

23.5 Governance Process

Future governance may include:

- Public specification releases
- Change logs
- Implementation notes
- Conformance tests
- Technical advisory review
- Legal and evidentiary review
- Community or standards-body feedback

---

24. Limitations

UCE is subject to operational, factual, legal, and technical limitations.

These include:

- Reliance on cryptographic assumptions
- Dependence on durable network accessibility
- Gateway indexing inconsistencies
- Human metadata accuracy limitations
- Public-key management risks
- Long-term software compatibility risks
- Risk of fraudulent or unauthorized uploads
- Limits of identity assertion without external attestation
- Jurisdiction-specific evidentiary rules

- Future cryptographic obsolescence
- Incomplete external adoption of rights declarations

No evidentiary system is perfect or immune to future technological change.

UCE should be understood as a technical evidence framework that strengthens preservation, chronology, integrity, and provenance analysis. It does not eliminate the need for legal judgment, factual investigation, corroborating evidence, or statutory registration where appropriate.

---

25. Recommended Best Practices

25.1 For Creators

Creators SHOULD:

- Preserve original source files
- Retain UCE certificates and upload receipts
- Maintain local backups
- Register commercially significant works with the U.S. Copyright Office or relevant national authority when appropriate
- Use consistent creator identity information
- Preserve version histories
- Document licensing agreements separately
- Keep account credentials secure
- Avoid uploading works they do not have authority to submit
- Avoid placing unnecessary sensitive information into public immutable records

25.2 For Organizations

Organizations implementing UCE-style systems SHOULD:

- Maintain public verification procedures
- Preserve cryptographic transparency
- Avoid proprietary lock-in
- Preserve auditability
- Use open standards whenever possible
- Maintain key rotation and compromise procedures
- Maintain conformance documentation
- Maintain incident response procedures
- Use multi-gateway or multi-path retrieval where feasible
- Distinguish immutable evidence from mutable operational routing

25.3 For Verifiers

Verifiers SHOULD:

- Recompute hashes independently
- Confirm manifest canonicalization
- Validate digital signatures
- Retrieve public keys from documented references
- Check multiple storage gateways where appropriate
- Distinguish technical verification from legal conclusions
- Preserve verification logs when used in disputes
- Review surrounding evidence before drawing ownership conclusions

---

26. Evidentiary Positioning and Public Communications

26.1 Proper Positioning

UCE should be understood as:

- A technical evidence framework
- A provenance framework
- A chronology framework
- A cryptographic integrity framework
- A preservation and verification framework
- An evidence layer complementary to copyright registration

26.2 Improper Positioning

UCE should not be described as:

- A government copyright registration
- Conclusive proof of ownership
- Legally determinative evidence
- Unforgeable proof
- Impossible to dispute
- A replacement for legal counsel
- A substitute for judicial findings

26.3 Preferred Terminology

Preferred terminology includes:

- Tamper-evident
- Cryptographically verifiable
- Independently auditable

- Immutable evidence record
- Verifiable chronology evidence
- Evidence of asserted authorship
- Evidence-layer infrastructure

26.4 Public Explanation

A concise public explanation may be:

UCE creates a cryptographically verifiable evidence record showing that a specific digital work, matching a specific file hash, was recorded no later than a verifiable timestamp, together with the creator or claimant assertions and rights declarations recorded at that time. UCE complements, but does not replace, copyright registration or legal adjudication.

---

27. Potential Evidentiary Use Cases

Potential use cases include:

- Copyright Claims Board proceedings
- DMCA disputes
- Licensing negotiations
- Platform moderation reviews
- AI provenance disputes
- AI training opt-out evidence
- Entertainment-industry catalog management
- Commercial due diligence
- Archive preservation
- Creative workflow preservation
- Educational creator-protection programs
- Institutional portfolio management
- Chain-of-custody documentation

The admissibility and weight of UCE evidence remain subject to:

- Applicable procedural rules
- Judicial discretion
- Adjudicator discretion
- Platform policies
- Contract terms
- Jurisdiction-specific standards
- Corroborating evidence

---

28. Conclusion

Universal Creation Evidence (UCE) is intended to provide a standardized framework for preserving and verifying cryptographic evidence associated with creative works.

The framework combines modern cryptographic methods, deterministic manifest canonicalization, durable decentralized storage references, public verification procedures, rights declarations, and human-readable evidence summaries to create independently auditable evidence records.

UCE is designed to complement, not replace, existing copyright registration systems and legal adjudication processes.

Its primary function is evidentiary preservation:

- Preserving chronology evidence
- Preserving integrity evidence
- Preserving provenance assertions
- Preserving rights declarations
- Enabling independent verification

As digital creation scales globally and AI systems increasingly interact with creative works, interoperable evidence frameworks will become increasingly important infrastructure for creators, platforms, adjudicators, archives, educational institutions, licensing systems, and rights ecosystems.

UCE aims to provide one such framework: technically rigorous, legally restrained, implementation-independent, and capable of independent verification.

Appendix A — Reference Manifest Structure

The following example is illustrative. It is intended to show a richer reference structure than a minimum conforming manifest. Exact field names and implementation details may evolve.

```
{
  "schema": "uce.evidence.manifest",
  "schemaVersion": "1.0.0",
  "framework": {
    "name": "Universal Creation Evidence",
    "abbreviation": "UCE",
    "version": "1.0"
  },
  "implementation": {
    "name": "Copyright by UCE",
    "abbreviation": "CbyUCE",
```

```
    "implementationVersion": "1.0",
    "implementationRole": "reference-implementation"
  },
  "work": {
    "title": "Example Work",
    "workType": "audio",
    "description": "Example description",
    "declaredCreationDate": "2026-05-01",
    "language": "en"
  },
  "claimant": {
    "displayName": "Example Creator",
    "claimantType": "individual",
    "identityStatus": "asserted",
    "assertionNotice": "Claimant identity is recorded as asserted unless
separately attested."
  },
  "files": [
    {
      "filename": "example.mp3",
      "contentType": "audio/mpeg",
      "bytes": 12345678,
      "hashes": {
        "sha256": "<file-sha256-hex>"
      },
      "storage": {
        "storageLayer": "arweave",
        "transactionId": "<file-transaction-id>",
        "uri": "ar://<file-transaction-id>"
      }
    }
  ],
  "rights": {
    "rightsStatus": "asserted",
    "aiTrainingProhibited": true,
    "commercialLicenseRequired": true,
    "attributionRequired": true,
    "rightsContactRequired": true,
    "declarationNotice": "Rights declarations are assertions and policy
expressions, not self-executing enforcement mechanisms."
  },
  "timestamps": {
    "manifestCreatedAt": "2026-05-10T00:00:00Z",
    "fileStorageSubmittedAt": "2026-05-10T00:01:00Z",
    "manifestStorageSubmittedAt": "2026-05-10T00:02:00Z",
    "ledgerTimestamp": "<ledger-or-block-timestamp-if-available>"
  },
  "storage": {
    "manifestStorageLayer": "arweave",
    "manifestTransactionId": "<manifest-transaction-id>",
    "manifestUri": "ar://<manifest-transaction-id>",
    "retrievalGateways": [
      "https://arweave.net/<manifest-transaction-id>",
      "https://permagate.io/<manifest-transaction-id>",
      "https://ar-io.dev/<manifest-transaction-id>",
      "https://g8way.io/<manifest-transaction-id>"
    ]
```

```
  },
  "hashes": {
    "contentHashAlgorithm": "sha256",
    "manifestHashAlgorithm": "sha256",
    "canonicalization": "RFC8785-JCS",
    "manifestHash": "<manifest-sha256-hex>"
  },
  "signatures": {
    "platformSignature": {
      "algorithm": "ES256",
      "keyId": "<key-id>",
      "publicKeyRef": "<public-key-reference>",
      "signatureFormat": "JWS-detached",
      "signatureValue": "<signature>"
    }
  },
  "verification": {
    "verificationUrl": "https://cbyuce.com/verify/<manifestHash>",
    "verificationMethod": "UCE-v1-standard-verification",
    "verificationNotice": "Technical verification does not independently
prove legal ownership, identity truthfulness, originality, or absence of
competing claims."
  }
}
```

Appendix B — Simplified Public Certificate Fields

A public UCE certificate may summarize:

- UCE certificate number or identifier
- Work title
- Work type
- File name
- Claimed creator or claimant
- Declared creation date
- Evidence creation date
- File hash
- Manifest hash
- Signature status
- Public key reference
- Storage transaction references
- Verification URL
- Rights declarations
- Framework version
- Implementation name
- Legal/evidentiary limitation notice

Recommended certificate notice:

This certificate summarizes a cryptographically verifiable evidence record. It does not constitute government copyright registration, legal ownership determination, or a judicial finding.

Appendix C — Public Verification Checklist

A public verifier should be able to answer:

1. Does the file hash match the manifest?
2. Does the canonicalized manifest produce the recorded manifest hash?
3. Does the digital signature validate against the referenced public key?
4. Can the public key reference be retrieved or otherwise validated?
5. Can the manifest storage reference be retrieved?
6. Can the file storage reference be retrieved, if the file is publicly available?
7. What timestamp references are available?
8. What creator or claimant assertions were recorded?
9. What rights declarations were recorded?
10. What limitations apply to this verification result?

Appendix D — Recommended Public Language

Preferred Language

- “Cryptographically verifiable evidence”
- “Tamper-evident evidence record”
- “Independent verification”
- “Evidence of asserted creation chronology”
- “Immutable evidence manifest”
- “Evidence-layer framework”
- “Technical provenance record”
- “Verifiable file integrity record”

Avoid

- “Guaranteed ownership”
- “Unforgeable proof”

- “Legal copyright registration”
- “Conclusive proof”
- “Impossible to dispute”
- “Court-proof ownership”
- “Automatic copyright enforcement”
- “AI-proof protection”

Recommended One-Sentence Description

UCE is a cryptographically verifiable evidence framework that records a work’s file hash, manifest, timestamp references, creator assertions, and rights declarations so third parties can independently verify integrity and chronology.

Recommended Short Disclaimer

UCE evidence complements, but does not replace, copyright registration, legal advice, or adjudication of ownership disputes.

Appendix E — Suggested Future Research Areas

- Post-quantum signature migration
- Decentralized identity integration
- Verifiable credentials for creator identity
- AI provenance interoperability
- Cross-chain and cross-storage evidence anchoring
- Court-admissibility studies
- Copyright Claims Board workflow studies
- Educational institutional workflows
- Machine-readable licensing standards
- Media authenticity interoperability
- Multi-party co-signature workflows
- Institutional repository integrations
- Rights-routing interoperability
- Public conformance test suites
- Reference verifier publication

Appendix F — Reference Standards and Technical Materials

The following materials are relevant to UCE implementation and review. This list is informative and may be expanded in future versions.

- RFC 8785 — JSON Canonicalization Scheme (JCS)
- FIPS 180-4 — Secure Hash Standard, including SHA-256
- FIPS 186-5 — Digital Signature Standard
- RFC 7515 — JSON Web Signature (JWS)
- RFC 7517 — JSON Web Key (JWK)
- RFC 7518 — JSON Web Algorithms (JWA)
- NIST post-quantum cryptography materials, including ML-DSA standardization materials
- Arweave protocol and developer documentation
- ar.io gateway and retrieval documentation
- W3C Verifiable Credentials materials
- W3C Decentralized Identifiers materials
- W3C PROV provenance model materials
- Content authenticity and provenance ecosystem materials

End of Document