PUBLIC FRAMEWORK SPECIFICATION FOR CREATIVE-WORK EVIDENCE

Universal Creative Evidence (UCE)

Evidentiary Standards & Verification Framework v1.0

Version	1.0 Public Draft
Date	May 2026
Framework Name	Universal Creative Evidence (UCE)
Prepared by	CbyUCE
Document Type	Technical Evidentiary Framework / Public Specification Draft
Documentation License	CC BY 4.0; UCE mark and trademark rights reserved separately

About the UCE name. *In the UCE name, creative refers to creative works and the people who make them. Evidence means the technical data and recorded statements associated with that work. Examples include photographs, illustrations, music, video, writing, and software.* ***Read the introduction to UCE evidence for creative works.***

Publication Notice. Universal Creative Evidence (UCE) is an implementation-independent evidentiary framework for preserving and verifying technical evidence associated with creative works. 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. **This framework does not constitute legal advice, a judicial determination, a government registration system, or a guarantee of copyright ownership.**

No affiliation or endorsement. UCE is independent and is not affiliated with, sponsored by, approved by, certified by, or endorsed by WIPO or any national copyright office. UCE evidence is not government copyright registration. Locate official national or regional offices through WIPO's official directory.

1. Executive Summary

Universal Creative Evidence (UCE) is a public-draft, implementation-independent evidentiary framework designed to preserve technical evidence and support independent verification of file and manifest integrity, chronology references, signatures, storage references, and recorded assertions 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:

- Does a presented file match the cryptographic hash contained in the recorded evidence package?
- What platform-recorded and network-reported chronology references are present, and what does each source actually support?
- 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; and 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, and 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; or 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 the separate live operational implementation and current reference architecture. UCE does not itself operate the CbyUCE commercial platform. 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 published 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 Creative 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.
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.

Term	Definition
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.
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, and 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; and 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; and 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, and 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; and 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; and 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; and 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`, and `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.

Current implementation note. *The CbyUCE production manifest schema predates this reference category model and organizes some equivalent or related data under different field names. It is not an export of the illustrative Appendix A structure, and the representative hosted record is not presented as a formal UCE v1 conformance determination. Appendix C maps the current production fields and observed verification scope.*

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; and 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; and hash algorithm identifier.

10.2 Manifest Hashing Requirements

A conforming implementation MUST compute the manifest hash over the canonicalized manifest. The manifest SHOULD include the manifest hash, hash algorithm, and 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; or 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; and 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; and 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; and 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; and 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, and 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 Distinct Chronology Sources

UCE evidence packages may contain several chronology values with different sources and evidentiary meanings: claimant-supplied creation date; a platform-recorded upload date; manifest creation or signing time; transaction submission time; network-reported block or storage-inclusion dates; verification time; and optional external timestamp-authority records. Interfaces and verification reports MUST label these sources distinctly and MUST NOT collapse them into one undifferentiated "timestamp."

12.2 Platform-Recorded Upload Date

In the current CbyUCE reference implementation, the **Platform-Recorded Upload Date** is the date CbyUCE recorded receiving the upload. That value is contained in the evidence manifest subsequently submitted to Arweave. It is a platform-recorded value and MUST NOT be described as Arweave-confirmed, an authoritative creation time, or a network-reported block date.

12.3 Separate File and Manifest Transactions

The current CbyUCE reference implementation submits the completed file and the signed Evidence Manifest as separate Arweave transactions. Each transaction may have its own network-reported block date. A verifier SHOULD report the file and manifest block dates separately when they are available and SHOULD identify unavailable or pending values without inferring them.

12.4 Complete UCE Record Date

When both the file and manifest Arweave block dates are available, the later calendar date MAY be displayed as the **Complete UCE Record Date**. This derived date indicates when both components had network-reported block dates; it does not replace either underlying value and does not convert the platform-recorded upload date into an Arweave-confirmed time.

12.5 Plain-Language and Technical Presentation

The primary plain-language utility claim for the current CbyUCE implementation is **proof of upload date**. Calendar-date chronology is often the most practically relevant public-facing claim, while exact UTC values SHOULD remain available for technical examination. Any no-later-than assertion MUST identify the specific network or storage reference on which it relies.

12.6 Limits of Chronology

UCE does not assert that a claimant-supplied creation date is independently true; that no earlier version existed elsewhere; that no competing claimant exists; that the claimant was the first creator; that chronology alone proves authorship or ownership; or that a work is original, non-infringing, or eligible for copyright.

12.7 No Qualified Timestamp Claim

Neither UCE nor CbyUCE is described by this framework as an RFC 3161 timestamp authority, qualified electronic timestamp provider, eIDAS trust service, governmental authority, or authoritative time source. A future implementation MAY include an external timestamp-authority reference only if that integration is actually implemented, documented, and labeled separately.

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; and third-party explorer data.

13.3 Verification Status Categories

A verification system SHOULD distinguish at least the following outcomes:

Status	Meaning
Verified (reported scope)	Every check expressly reported as performed succeeded. The result MUST identify the checks performed, unavailable inputs, and checklist items not tested; it MUST NOT imply that every possible verification or conformance check succeeded.
File Mismatch	The submitted file hash does not match the manifest entry.

Status	Meaning
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.

Current hosted-result note. *On the representative CbyUCE record observed August 24, 2026, the hosted* ***VERIFIED*** *result reported a matching manifest hash and a valid platform signature. It did not report a successful public file-byte comparison, file retrieval result, File Arweave Block Date, or Complete UCE Record Date. Appendix C records that scope explicitly.*

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; and 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 Technical Verification Determines

A conforming verification result determines only the checks it actually performs and reports. Depending on the available inputs, it can determine whether: a presented file's bytes produce the recorded content hash; the manifest produces the recorded manifest hash under the declared canonicalization method; the platform signature is valid for the referenced key and signing payload; storage references resolve to the checked data; recorded transaction references and network-reported chronology values are internally consistent; and particular claimant or rights statements are present in the evidence package.

A successful result therefore means that the checked technical relationships are consistent at the time of verification. It does not transform recorded statements into independently verified facts.

14.2 Recorded Assertions

Creator name, claimant identity, authorship, ownership basis, creation date, publication status, rights authority, AI-use declarations, and other claim details are recorded assertions unless a separately identified attestation process actually verifies the relevant fact. A platform signature authenticates the platform origin and integrity of the signed manifest; it is not, by itself, the claimant's cryptographic signature or a validation that the claimant's statements are true.

14.3 What Verification Does Not Independently Determine

Verification does not independently determine: legal ownership; the truth of identity or authorship assertions; exclusivity of authorship; originality; copyright eligibility; nonexistence of prior creators or competing claims; non-infringement; validity or scope of a license; enforceability of an AI opt-out declaration; admissibility or evidentiary weight; or absence of fraud, coercion, impersonation, or unauthorized upload.

14.4 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; and 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

```
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; and 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 and 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, and 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, and 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; and 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, and 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; and 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; and 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, or 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; and 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.

20.7 Abuse, Disputes, and Correction Notices

Implementations SHOULD publish procedures for reporting unauthorized uploads, impersonation, disputed authorship or ownership, competing claims, legal requests, and other abuse. Because an immutable record may not be removable from the underlying storage network, a correction, dispute, or takedown outcome MAY need to be represented through a linked notice or status record rather than deletion of the original artifact. Such a notice MUST NOT silently alter the historical manifest.

20.8 Framework Principles and Operational Procedures

This section states framework principles. It does not claim that every implementation has completed each operational procedure. CbyUCE-specific reporting, legal-request, deletion, and account processes are governed by CbyUCE's published policies and operational documentation.

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; and 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; and 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; and 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; a tamper-evident evidence record; designed for independent verification; evidence of asserted chronology; and evidence-layer infrastructure. They SHOULD NOT describe UCE as guaranteed ownership; government copyright registration; conclusive proof; impossible to dispute; unforgeable proof; or 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; and 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 Present Governance

The organizational maintainer of the UCE Framework v1.0 Public Draft is CbyUCE, an alternate name of 5 Race Street LLC. The current public governance status, change record, feedback channel, contribution process, disclosure contact, and unresolved governance decisions are documented on the UCE Governance page.

23.6 Change Control

Changes to a published framework version MUST be identified in a public change record. Breaking technical changes require a new major version; backward-compatible additions require a minor version; and clarifications or corrections may use a patch version. Historical versions SHOULD remain available so a verifier can apply the version declared by an evidence package.

23.7 Pending Governance Decisions

An individually named editor, formal conflict-of-interest policy, independent review body, and external conformance authority have not yet been adopted. The framework MUST NOT imply that those structures, or any WIPO or standards-body approval, currently exist.

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; and 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; and 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; and 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; and 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; and 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; or a substitute for judicial findings.

26.3 Preferred Terminology

Preferred terminology includes tamper-evident; cryptographically verifiable integrity and chronology; designed for independent verification; immutable evidence manifest; recorded assertions; proof of upload date; and evidence-layer infrastructure. Full provider-independent reproducibility SHOULD NOT be claimed until the necessary tools, test vectors, key-history material, and validation evidence are published.

26.4 Public Explanation

> *UCE defines a cryptographically verifiable evidence record linking an exact file hash, chronology references, platform signatures, storage references, and the creator or claimant assertions recorded at that time. Technical verification determines whether the checked relationships are consistent; it does not determine authorship, ownership, originality, or legal effect. UCE complements, but does not replace, national 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; and 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; and corroborating evidence.

28. Conclusion

Universal Creative Evidence (UCE) is intended to provide a publicly documented 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, recorded rights declarations, and human-readable evidence summaries to create evidence records designed for independent audit.

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; and 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. It is not an export of the current CbyUCE production manifest and does not assert that every illustrated field is implemented today. Exact field names and implementation details may evolve; Appendix C provides the current production crosswalk.

```
{
  "schema": "uce.evidence.manifest",
  "schemaVersion": "1.0.0",
  "framework": {
    "name": "Universal Creative Evidence",
    "abbreviation": "UCE",
    "version": "1.0"
  },
  "implementation": {
    "name": "CbyUCE",
    "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": {
    "platformRecordedUploadDate": "2026-05-10",
    "platformRecordedUploadDateNotice": "Recorded by the implementation and included in the manifest; not an
Arweave-confirmed time.",
    "manifestCreatedAt": "2026-05-10T00:00:00Z",
    "fileStorageSubmittedAt": "2026-05-10T00:01:00Z",
    "manifestStorageSubmittedAt": "2026-05-10T00:02:00Z",
    "fileArweaveBlockDate": "<network-reported-file-block-date-if-available>",
    "manifestArweaveBlockDate": "<network-reported-manifest-block-date-if-available>",
    "completeUceRecordDate": "<later-file-or-manifest-block-date-if-both-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
- Platform-Recorded Upload Date, labeled as a platform value
- File Arweave Block Date, when independently retrieved and available
- Manifest Arweave Block Date, when independently retrieved and available
- Complete UCE Record Date, only when both block dates are available and verified
- File hash
- Manifest hash
- Signature status
- Public key reference
- Separate file and manifest 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?

Current CbyUCE Production Crosswalk

This informative crosswalk records the representative production example observed on August 24, 2026. It is a time-bounded description of one hosted record, not a schema guarantee, formal conformance certification, independent audit, or claim that every checklist item has succeeded. Dynamic network results should be rechecked at the cited production URLs.

Production item	Current field or value	What it determines

Schema identifiers	`schema` = `uce.evidence.manifest`
	`schemaVersion` = `1.0.0`
	`manifestVersion` = `1`
		Identifies the production manifest format. It does not establish formal UCE framework conformance.

Platform-Recorded Upload Date	Production field: `registrationTimestamp` Value: `2026-04-26T23:02:31.809Z`	The field name is retained in the production schema, but the public label is **Platform-Recorded Upload Date**. It is a CbyUCE-recorded value included in the signed manifest, not an Arweave-confirmed time or legal creation date.
Recorded claimant and work assertions	`work.title`, `work.authorName`, `work.creationDate`, `identity`, `policy`, and `attestations`	Shows the claims, declarations, assurance metadata, and acknowledgments recorded in the package. It does not independently determine authorship, ownership, originality, or legal effect.

Production item	Current field or value	What it determines
File fingerprint	`files[0].sha256` = `4591bc72cc30a1fc97fd768ec8e0a1b0b271d8a5b0a1a6c62f166b3f545d5f44`	Identifies the exact bytes claimed for the submitted file. A verifier needs those file bytes to recompute and compare this hash.
Manifest hash and canonicalization	`hashes.manifestHash` = `5d476448cf9f05d1fd5d3b863f8a212732f1593d90dc63b292f557c28775a48f` `hashes.algorithm` = `sha256` `hashes.canonicalization` = `RFC8785`	The hosted result reported `hashMatches: true` for the manifest hash under the declared canonicalization method.
Platform signature	JWS algorithm: `ES256` `signatures.platformKeyId` = `platform-classical-1761279471109` `signatures.platformPublicKeyRef` = `8IQIcijOSMO0dtKrNsVPTFbJBGiIaL6x9EsUW4UdApQ`	The hosted result reported `sigValid: true`. This authenticates the platform signature and signed payload; it is not the claimant's personal signature or proof that the claimant's statements are true. Published key-history and revocation validation remain separate diligence items.

Two Transactions and Chronology

Component	Current production reference	Observed status and limitation
Completed file transaction	`anchors.fileStorage.txId` = `_D99Kv2qnhY0XDu-k105M_tqVpK_8Qbgrmcd_lfqeuc`	The raw manifest records this separate file transaction. At observation, the hosted data reported zero confirmations and no File Arweave Block Date; public preview was unavailable. Submission or a transaction reference alone is not a network-reported block date.
Signed Evidence Manifest transaction	`Oagba5o2yoEn-JT1C1RbVr0VVKdcxH797tHk9Xao8kg`	The hosted record and its URL identify the manifest transaction. The stored raw manifest's own `anchors.arweave.txId` value is empty; the production artifacts reviewed here do not establish why. Verifiers must not infer that the raw manifest self-contains this value.
Manifest Arweave Block Date	`2026-04-26T23:07:55.000Z` Block height: `1905830`	The hosted data reported the manifest transaction confirmed and displayed April 26, 2026 as the Manifest Arweave Block Date.
File Arweave Block Date	Not present in the observed hosted result	Must remain absent until independently retrieved and available. It must not be inferred from the platform-recorded upload date, submission event, or manifest block date.

Component	Current production reference	Observed status and limitation
Complete UCE Record Date	`uceRecordBlockTimestamp` = `null` `uceRecordDateUtc` = `null`	Not available because both file and manifest block dates were not available and verified. No Complete UCE Record Date should be displayed for this observed state.

Current Verification Routes and Scope

- **Hosted verification display:** `https://cbyuce.com/verify/5d476448cf9f05d1fd5d3b863f8a212732f1593d90dc63b292f557c28775a48f`
- **Hosted JSON result:** the same URL with `?format=json`. This is current implementation output and should not be mistaken for the raw manifest.
- **Raw manifest:** `https://arweave.net/Oagba5o2yoEn-JT1C1RbVr0VVKdcxH797tHk9Xao8kg`
- **Third-party explorer:** `https://viewblock.io/arweave/tx/Oagba5o2yoEn-JT1C1RbVr0VVKdcxH797tHk9Xao8kg`. Explorer availability is supplementary and may vary by client or access policy.
- **Lens browser verification:** UCE Evidence Lens provides profile-limited manifest-hash and ES256 signature checks, private local-file comparison, signing-key provenance, chronology inspection, and a public demonstration file with its manifest.
- **WebMCP inspection:** eight registered, read-only tools expose public-record inspection and verification to compatible browser agents, including unsigned dated inspection reports. These reports are snapshots of completed checks and limitations, not UCE Certificates or independent audit reports.

The reviewed Extended 1.1.0 profile supports independent manifest-hash recomputation and critical-profile ES256 signature checks. Standard 1.0.0 supports reviewed ES256 checks and, on an exact match, a narrow public hash projection; older non-matches can remain unresolved where original preimage timestamps are unavailable. Hybrid 2.0.0 and 2.1.0 records are readable, but their signatures and manifest-hash recomputation are not independently verified in the released runtime. A match covers only the stated profile fields and does not establish the truth of assertions.

The hosted CbyUCE route and live Lens browser checks are available now. Lens runs supported cryptographic checks locally, while live retrieval and network chronology require connectivity. A complete versioned standalone/offline provider-independent release, packaged success and failure vectors, full public-key history and revocation records, and documented outside reproduction remain assurance milestones. An independent security audit is not claimed. UCE remains **designed for independent verification**, with supported browser checks available today; complete provider-independent continuity is not yet claimed. See Verification and Conformance Status for current coverage and limits.

Appendix D - Recommended Public Language

Preferred Language

Cryptographically verifiable evidence
Tamper-evident evidence record
Designed for 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 and is designed to support third-party verification of 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
- Completion of the provider-independent verifier package and documented outside reproduction

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 -