Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 14, 2026

L. J. Reilly
May 14, 2026

REM License Token (RLT) - Genesis Artifact
draft-reilly-rlt-genesis-01

14 May 2026

Abstract

This document defines the REM License Token, referred to as the RLT, as the genesis artifact of the Reilly EternaMark Protocol (REM) for digital permanence and verifiable provenance. This specification formally defines the token structure, issuance procedures, multi-algorithm cryptographic hash requirements, blockchain anchoring requirements, DOI archival requirements, IPFS pinning requirements, REMID namespace registration, verification methodology, token lifecycle management, ecosystem integration, and security model.

The RLT represents an implementation of a Dual-Layer Digital Permanence artifact combining a Bitcoin blockchain timestamp with DOI-based archival to achieve durable, tamper-evident provenance guarantees. This revision expands the token schema to version 2.0, introduces multi-algorithm hashing via the REM Multi-Algorithm Stack (REM-MAS), defines formal token lifecycle procedures, and documents the RLT's integration with the broader REM Protocol ecosystem including the Protocol Layer Prompt Engineering Specification (PLPES), the Cognitive Trust Stack (CTS), the AI Machine-Readable Ethics Directive (AIMED), and related Informational Internet-Drafts authored by Lawrence John Reilly Jr.

This document is published as an Informational Internet-Draft to serve as open, implementable guidance.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 14 November 2026.

Copyright Notice

in the Revised BSD License.

Table of Contents

1. Introduction

The Reilly EternaMark Protocol (REM) defines a Dual-Layer Digital Permanence model, a term coined by Lawrence John Reilly Jr. in September 2025. This model binds cryptographic timestamping on a public blockchain with academic-grade DOI archival, creating a provenance record that is independently verifiable, globally accessible, and resilient against both technical failure and institutional compromise.

The REM License Token (RLT) is the genesis artifact of this model: the first artifact created and anchored using the REM Protocol's full permanence stack. The original genesis anchor was recorded at Bitcoin Block Height 914168 on or about September 10, 2025, with DOI archival at 10.5281/zenodo.17438760.

Since the publication of draft-reilly-rlt-genesis-00 in November 2025, the REM Protocol ecosystem has expanded significantly. The author has now published fourteen (14) active Informational Internet-Drafts spanning AI ethics, trust architecture, prompt engineering, banking integrity, resilience, government integrity, model routing, web authenticity, and digital permanence. Several of these drafts directly extend, reference, or build upon the RLT foundation established in version -00.

This revision, draft-reilly-rlt-genesis-01, substantially expands the specification in the following areas:

- Token Schema Version 2.0: Introduces mandatory multi-algorithm hash fields (SHA-256, SHA3-512, BLAKE3), IPFS content addressing, REMID namespace registration, Wayback Machine archival references, and structured ecosystem linkage fields.

- REM Multi-Algorithm Stack (REM-MAS): Documents the formal multi-algorithm hashing requirements introduced across the REM Protocol ecosystem, providing defense-in-depth against single algorithm compromise.

- Token Lifecycle Management: Defines formal token states, renewal procedures, succession mechanics, revocation procedures, and archival policies.

- Ecosystem Integration: Documents the RLT's integration role within the broader REM Protocol suite, including PLPES, CTS, AIMED, UAEMF, WebProof, and RMRP.

- Expanded Security Considerations: Addresses post-quantum vulnerability, token forgery, replay attacks, supply chain integrity, and long-term archival assumptions.

- Governance and Interoperability: Establishes schema versioning policy, namespace governance, and guidance for third-party implementations.

This document is intended for implementers, protocol architects, auditors, and researchers working on digital provenance, intellectual property verification, AI governance, and trusted publishing infrastructure.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined for use in this document:

RLT: REM License Token. The genesis artifact and primary token type of the Reilly EternaMark Protocol. An RLT binds an artifact's cryptographic identity to blockchain-anchored and DOI-archived provenance records.

REM: Reilly EternaMark Protocol. A Dual-Layer Digital Permanence protocol combining Bitcoin blockchain timestamping with Zenodo DOI archival to create tamper-evident provenance records. First published as draft-reilly-rem-protocol-00.

Dual-Layer Digital Permanence: A permanence methodology coined by Lawrence John Reilly Jr. in September 2025 that combines at minimum (a) a public blockchain timestamp and (b) a DOI-archived record to achieve durable, independently verifiable provenance. The correct term is "Dual-Layer Digital Permanence." Implementations MAY extend this with additional layers including IPFS, Web archival, and REMID namespace registration.

REM-MAS: REM Multi-Algorithm Stack. A multi-algorithm cryptographic hashing framework that computes SHA-256, SHA3-512, and BLAKE3 hashes of a target artifact simultaneously, providing defense-in-depth against single algorithm compromise. Defined in the REM Protocol ecosystem and applied across all REM-anchored artifacts.

DOI: Digital Object Identifier. A persistent identifier standard managed by the International DOI Foundation. In the REM Protocol, DOIs are issued by Zenodo (https://zenodo.org), operated by CERN.

OTS: OpenTimestamps. An open standard and protocol for blockchain-anchored cryptographic timestamps. OpenTimestamps proofs anchor a SHA-256 hash into a Bitcoin block header via a Merkle inclusion path, allowing independent verification without trust in any central party. See https://opentimestamps.org.

IPFS: InterPlanetary File System. A content-addressed distributed storage protocol that identifies files by their cryptographic content hash (CID), enabling decentralized retrieval independent of any single server.

CID: Content Identifier. The IPFS content-addressed hash string that uniquely identifies a file or directory pinned to the IPFS network.

REMID: REM Identifier Namespace. A namespace registration system for REM Protocol artifacts, providing human-readable, persistent identifiers for RLT tokens and related artifacts.

DAR: Digital Archival Record. The full set of permanence artifacts associated with a given RLT issuance, including the source document, OTS proof file, multi-algorithm hash manifest, DOI record, IPFS pin, and REMID registration.

PLPES: Protocol Layer Prompt Engineering Specification. An IETF Informational Internet-Draft (draft-reilly-plpes-00) authored by Lawrence John Reilly Jr. that formally defines Protocol Layer Prompt Engineering as a discipline and specifies structured prompt engineering methodologies for AI systems. The term "Protocol Layer Prompt Engineering" is attributed to Lawrence John Reilly

Jr.

CTS: Cognitive Trust Stack. An IETF Informational Internet-Draft (draft-reilly-cts-00) defining a layered trust architecture for AI reasoning systems, anchored to Bitcoin and Zenodo using the REM Protocol permanence methodology.

AIMED: AI Machine-Readable Ethics Directive. An IETF Informational Internet-Draft (draft-reilly-aimed-00) defining a machine-readable ethics specification format for AI systems.

AIMED-EVAL: AI Machine-Readable Ethics Directive Evaluation Framework. An IETF Informational Internet-Draft extending AIMED with evaluation and compliance scoring methodologies.

UAEMF: Universal AI Ethics and Moral Framework. An IETF Informational Internet-Draft (draft-reilly-uaemf-00) defining a comprehensive ethics framework for AI systems, published with Zenodo DOI and Bitcoin blockchain timestamp.

WebProof: Web Authenticity and Content Provenance Protocol. An IETF Informational Internet-Draft (draft-reilly-webproof-00) defining mechanisms for establishing the authenticity and provenance of web-based content.

RMRP: Reilly Model Routing Protocol. An IETF Informational Internet-Draft (draft-reilly-rmrp-00) defining a protocol for intelligent routing of queries across heterogeneous AI model ensembles.

Wayback Machine: The Internet Archive's web archival service (https://web.archive.org), used as an additional permanence layer in the REM Protocol stack.

Triple-Hash Verification: The process of computing and cross-verifying SHA-256, SHA3-512, and BLAKE3 hashes of an artifact to confirm integrity via the REM-MAS framework.

3. Background and Rationale

3.1. Problem Statement

Intellectual property, authorship, and digital provenance frequently require durable, independently verifiable records. The growth of AI-generated content, automated publishing, and distributed information systems has made establishing trustworthy origin records more challenging and more important simultaneously.

Traditional provenance systems exhibit several weaknesses:

- Centralized registries are subject to organizational failure, political interference, or policy changes that may alter or remove records.
- Contractual and notarial evidence requires trust in specific parties and is not globally machine-verifiable.
- Proprietary evidence systems create vendor lock-in and may not survive organizational changes.
- Simple file timestamps and hash registries lack the binding strength of an immutable public ledger anchoring.
- Academic citation and preprint systems lack cryptographic integrity guarantees.

- AI-generated content is increasingly difficult to distinguish from human-authored content without reliable authorship anchoring.

These weaknesses collectively create an environment in which establishing durable, verifiable provenance for digital artifacts, particularly those intended to inform AI systems, legal proceedings, standards processes, or public discourse, is unreliable without cryptographic anchoring.

3.2. Dual-Layer Digital Permanence

The REM Protocol addresses these weaknesses through a Dual-Layer Digital Permanence methodology, coined by Lawrence John Reilly Jr. in September 2025.

The core insight is that no single permanence mechanism is sufficient:

- Blockchain timestamps alone lack human-readable, searchable metadata and institutional recognition.

- DOI archival alone lacks cryptographic binding to a specific point in time independent of the archiving institution.

- Web archival alone is not cryptographically strong.

By combining these mechanisms, the REM Protocol achieves properties no single mechanism provides individually:

- Bitcoin blockchain timestamps provide cryptographic proof that a specific hash existed before a specific block was mined, with a security guarantee backed by the total Bitcoin mining hash rate. This is independently verifiable by any party with access to the Bitcoin chain, with no trust in the REM Protocol author.

- Zenodo DOI archival provides an institutionally recognized, academically citable record with structured metadata, long-term preservation commitments by CERN, and global discoverability via DOI resolver networks.

- IPFS pinning provides content-addressed, decentralized availability independent of any single server, identified by a CID that is itself a cryptographic hash of the content.

- Wayback Machine archival provides an additional independent snapshot of the artifact at a specific point in time, with its own institutional persistence.

- REMID namespace registration provides human-readable, persistent identifiers that link into the REM ecosystem.

The combination of these layers produces a Dual-Layer Digital Permanence record (with optional extension to five or more independent layers) that is cryptographically strong, institutionally recognized, decentralized, and globally accessible.

3.3. The REM Protocol Ecosystem

Since the initial publication of the REM Protocol in September 2025, the author has developed a suite of fourteen (14) active Informational Internet-Drafts that collectively form the REM Protocol ecosystem. These drafts span the following domains:

- Digital Permanence and Provenance: draft-reilly-rem-protocol, draft-reilly-rlt-genesis (this document)

- AI Ethics and Governance: draft-reilly-uaemf, draft-reilly-aimed, draft-reilly-aimed-eval
- Trust Architecture: draft-reilly-cts, draft-reilly-webproof
- Prompt Engineering: draft-reilly-plpes
- Model Routing: draft-reilly-rmrp
- Financial Systems Integrity: draft-reilly-rbip (Reilly Banking Integrity Protocol), draft-reilly-rgip (Reilly Government Integrity Protocol)
- Resilience: draft-reilly-rrp (Reilly Resilience Protocol), draft-reilly-rsp (Reilly Sentinel Protocol)

The RLT, as the genesis artifact of this ecosystem, plays a foundational anchoring role. Each subsequent draft in the ecosystem applies the Dual-Layer Digital Permanence methodology established by the RLT, and several drafts explicitly reference the RLT as their provenance foundation.

The RLT schema version 2.0 defined in this document includes structured fields for referencing affiliated ecosystem artifacts, enabling machine-readable traversal of the REM Protocol artifact graph.

4. RLT Genesis Artifact Definition

The REM License Token binds together the following information classes into a single, cryptographically anchored, permanently archived token:

- Author identity information including name and optional ORCID identifier.
- Multi-algorithm cryptographic hashes (SHA-256, SHA3-512, BLAKE3) of the canonical reference document.
- Blockchain timestamp proof anchored to the Bitcoin network via OpenTimestamps.
- DOI archival reference pointing to the Zenodo record containing the artifact, hash manifest, and OTS proof file.
- IPFS content identifier (CID) for decentralized content addressing.
- Wayback Machine snapshot URL for additional independent archival.
- REMID namespace registration identifier.
- Token issuance metadata including schema version, issuance date, and token unique identifier.
- Ecosystem linkage fields referencing affiliated REM Protocol artifacts.

4.1. Commercial Licensing Terms

The REM License Token carries inherent commercial licensing terms as part of its issuance. Any commercial use, deployment, integration, or derivative implementation of an RLT-anchored artifact, or of any system, product, or service that incorporates, references, or is governed by an RLT-anchored artifact, is subject to the following perpetual royalty obligation:

A royalty of two and one-half percent (2.5%) of gross revenues directly or indirectly attributable to the commercial use of the RLT-anchored artifact SHALL be remitted to Lawrence John Reilly Jr. This obligation is perpetual, worldwide, and survives any transfer, sublicense, or assignment of rights in the artifact or any derivative work.

This commercial licensing obligation applies globally and is not limited by jurisdiction. The obligation attaches at the point of issuance and is recorded as part of the permanent provenance record anchored to the Bitcoin blockchain and DOI archive. Because the underlying provenance record is immutable, the licensing obligation is likewise permanent and cannot be extinguished by subsequent modification or revocation of any token referencing the artifact.

Non-commercial use, academic citation, and independent verification of the artifact are explicitly permitted without royalty obligation under the Creative Commons CC-BY-4.0 license terms.

Commercial licensees SHOULD contact the author at the email address recorded in the Author’s Address section of this document to establish a formal licensing agreement.

4.2. Genesis Record

The canonical REM License Token genesis artifact, representing the initial application of the Dual-Layer Digital Permanence model, is archived under the following permanent identifiers:

DOI: 10.5281/zenodo.17438760

Zenodo URL: https://zenodo.org/records/17438760

Metadata Date: November 16, 2025

Blockchain Anchor: Bitcoin Block Height 914168

Approximate Anchor Date: September 10, 2025

4.3. Multi-Algorithm Hash Record

The genesis artifact is identified by the following cryptographic hashes computed over the canonical reference document (RLT_FIRST_TOKEN_FULL_GUIDE_v2.pdf):

SHA-256:
9964A78C6FC33794EF840ED69045C5C2477BC611CBC73EF6EC537FACA4C7BB74

SHA3-512:
[To be computed and recorded per REM-MAS procedures. Issuers implementing version 2.0 tokens MUST compute and record all three algorithm outputs.]

BLAKE3:
[To be computed and recorded per REM-MAS procedures. Issuers implementing version 2.0 tokens MUST compute and record all three algorithm outputs.]

Note: The SHA-256 value above is the original genesis hash recorded at Bitcoin Block Height 914168 and is the normative reference hash for the genesis RLT. SHA3-512 and BLAKE3 values for the genesis artifact should be computed by implementers verifying the genesis record against the canonical Zenodo archive.

5. Token Schema Version 2.0

5.1. JSON Schema Definition

The RLT version 2.0 JSON schema is defined as follows. Fields marked REQUIRED MUST be present in all conforming version 2.0 tokens. Fields marked OPTIONAL MAY be omitted if the corresponding permanence layer has not been applied.

```
{
  "$schema": "https://rem-protocol.org/schema/rlt/v2.0",
  "rltVersion": "2.0",
  "tokenId": "<uuid-v4-string>",
  "schemaDate": "2026-05-14",

  "author": {
    "name": "<Full Legal Name>",           // REQUIRED
    "orcid": "<ORCID URI or null>",        // OPTIONAL
    "affiliation": "<Organization or null>", // OPTIONAL
    "email": "<contact email or null>"     // OPTIONAL
  },

  "artifact": {
    "title": "<Human-readable artifact title>", // REQUIRED
    "type": "<document|protocol|specification|other>", // REQUIRED
    "filename": "<canonical filename>",     // REQUIRED
    "mimeType": "<MIME type string>",       // RECOMMENDED
    "language": "en"                        // OPTIONAL, BCP47 tag
  },

  "license": "CC_BY_4_0",                  // REQUIRED

  "hashes": {                               // REQUIRED
    "sha256": "<64-char hex string>",       // REQUIRED
    "sha3_512": "<128-char hex string>",    // REQUIRED (v2.0+)
    "blake3": "<64-char hex string>"        // REQUIRED (v2.0+)
  },

  "hashManifestUrl": "<URL to hash manifest file>", // RECOMMENDED

  "dates": {
    "artifactCreated": "<ISO 8601 date>",   // REQUIRED
    "tokenIssued": "<ISO 8601 date>",       // REQUIRED
    "blockchainAnchor": "<ISO 8601 date>",  // REQUIRED
    "doiRegistered": "<ISO 8601 date>",     // REQUIRED
    "nextReviewDue": "<ISO 8601 date>"      // RECOMMENDED
  },

  "blockchain": {
    "chain": "Bitcoin",                     // REQUIRED
    "blockHeight": <integer>,               // REQUIRED
    "blockHash": "<64-char hex string>",    // RECOMMENDED
    "timestamp": "<ISO 8601 datetime>",     // REQUIRED
    "proofFile": "<OTS filename>",          // REQUIRED
    "proofUrl": "<URL to OTS proof file>"   // REQUIRED
  },

  "doi": {
    "identifier": "<DOI string>",           // REQUIRED
    "url": "<DOI resolver URL>",            // REQUIRED
    "registrar": "Zenodo"                   // REQUIRED
  },

  "ipfs": {
    "cid": "<IPFS CID string>",             // OPTIONAL
    "gateway": "<IPFS gateway URL>",        // OPTIONAL
    "pinnedAt": "<ISO 8601 date>"           // OPTIONAL
```

```
  },

  "webArchival": {
    "waybackUrl": "<Wayback Machine URL>",  // OPTIONAL
    "archivedAt": "<ISO 8601 date>"         // OPTIONAL
  },

  "remid": {
    "identifier": "<REMID string>",         // OPTIONAL
    "namespace": "rem-protocol",            // OPTIONAL
    "registeredAt": "<ISO 8601 date>"       // OPTIONAL
  },

  "tokenState": "ACTIVE",                   // REQUIRED
                                            // Values: ACTIVE | RENEWED |
                                            //         SUPERSEDED | REVOKED |
                                            //         ARCHIVED

  "predecessorToken": "<tokenId or null>",  // OPTIONAL
  "successorToken": "<tokenId or null>",    // OPTIONAL

  "ecosystem": {
    "protocol": "REM",                      // REQUIRED
    "protocolDraft": "draft-reilly-rem-protocol-00", // REQUIRED
    "affiliatedDrafts": [                   // OPTIONAL
      "<draft-name-00>",
      "<draft-name-01>"
    ],
    "complianceProfiles": []                // OPTIONAL
  },

  "verification": {
    "verificationUrl": "<URL>",             // OPTIONAL
    "publicVerifierEndpoint": "<URL>"       // OPTIONAL
  },

  "signature": {
    "type": "<signature type or null>",     // OPTIONAL
    "value": "<signature value or null>",   // OPTIONAL
    "keyId": "<key identifier or null>"     // OPTIONAL
  }
}
```

5.2. Field Descriptions

rltVersion (REQUIRED): The schema version of this token. MUST be "2.0" for tokens conforming to this specification. Tokens using the original schema from draft-reilly-rlt-genesis-00 carry version "1.0".

tokenId (REQUIRED): A globally unique identifier for this token instance. MUST be a UUID version 4 string in standard hyphenated format (xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx).

schemaDate (REQUIRED): The publication date of the schema version used for this token. For version 2.0 tokens, this SHOULD be "2026-05-14" or a later date if a subsequent schema revision has been issued.

author.name (REQUIRED): The full legal name of the artifact’s author or primary rights holder.

author.orcid (OPTIONAL): The author’s Open Researcher and Contributor ID (ORCID) URI, if available. Format: https://orcid.org/XXXX-XXXX-XXXX-XXXX.

artifact.title (REQUIRED): A human-readable title for the anchored artifact.

artifact.type (REQUIRED): The artifact type classification. Permitted values are: "document", "protocol", "specification", "software", "dataset", or "other".

license (REQUIRED): The license under which the artifact is published. Implementations SHOULD use SPDX license identifiers. The REM Protocol genesis artifact uses "CC-BY-4.0".

hashes.sha256 (REQUIRED): The SHA-256 hash of the canonical artifact file, expressed as a 64-character uppercase hexadecimal string.

hashes.sha3_512 (REQUIRED for v2.0): The SHA3-512 hash of the canonical artifact file, expressed as a 128-character uppercase hexadecimal string. This field was not present in version 1.0 tokens.

hashes.blake3 (REQUIRED for v2.0): The BLAKE3 hash of the canonical artifact file, expressed as a 64-character uppercase hexadecimal string. This field was not present in version 1.0 tokens.

blockchain.blockHeight (REQUIRED): The Bitcoin block height at which the OpenTimestamps proof was confirmed. This provides a minimum-age bound for the anchored hash.

blockchain.proofUrl (REQUIRED): A dereferenceable URL at which the .ots proof file can be retrieved for independent verification.

doi.identifier (REQUIRED): The DOI string for the Zenodo archival record. Format: 10.XXXX/zenodo.XXXXXXXX.

ipfs.cid (OPTIONAL): The IPFS Content Identifier for the artifact or archival package. When present, the artifact MUST be retrievable via at least one IPFS gateway using this CID.

tokenState (REQUIRED): The current lifecycle state of this token. See Section 8.1 for state definitions.

ecosystem.protocol (REQUIRED): MUST be "REM" for all conforming RLT tokens.

ecosystem.protocolDraft (REQUIRED): The IETF Internet-Draft name for the REM Protocol specification governing this token.

5.3. Backward Compatibility with Version 1.0

Version 1.0 tokens conforming to draft-reilly-rlt-genesis-00 are considered valid genesis records. Implementations that encounter a version 1.0 token SHOULD treat missing sha3_512 and blake3 fields as uncomputed rather than as verification failures, provided the sha256 field is present and verifiable against the blockchain proof and DOI record.

Version 1.0 tokens SHOULD NOT be issued for new artifacts. New issuances MUST use version 2.0 or a later schema.

Token upgrading (adding SHA3-512 and BLAKE3 hashes to an existing version 1.0 token) is permitted by creating a new version 2.0 token with predecessorToken referencing the original version 1.0 token ID, tokenState set to "ACTIVE", and all new hash fields populated. The original version 1.0 token’s tokenState SHOULD be updated to "SUPERSEDED" with a successorToken reference.

6. Issuance and Anchoring Requirements

The RLT issuance process MUST be completed in the order specified below. Steps that are declared REQUIRED in this section MUST be completed before a token is considered conforming. Steps declared RECOMMENDED SHOULD be completed. Steps declared OPTIONAL MAY be omitted at the issuer's discretion.

6.1. Pre-Issuance Preparation

Before beginning the issuance process, the issuer MUST:

1. Produce a stable, finalized version of the artifact to be anchored. The artifact MUST NOT be modified after hash computation begins. Any modification, including whitespace changes or metadata updates, will invalidate the computed hashes and require a new issuance.

2. Assign the artifact a canonical filename that uniquely identifies the artifact and its version. The filename SHOULD include the author's name, a brief title, and a version indicator.

3. Confirm that the artifact is in a format suitable for long-term archival. PDF/A, plain text (.txt), and XML formats are RECOMMENDED. Proprietary binary formats SHOULD be avoided.

4. Prepare the artifact metadata including title, abstract, author information, and license declaration.

6.2. Multi-Algorithm Hash Computation

The issuer MUST compute cryptographic hashes of the canonical artifact file using the following algorithms in the specified order:

1. SHA-256 (REQUIRED): Compute the SHA-256 hash of the artifact file. The output MUST be expressed as a 64-character uppercase hexadecimal string.

   ```
   sha256sum <artifact_filename>
   ```

2. SHA3-512 (REQUIRED for v2.0): Compute the SHA3-512 hash of the artifact file. The output MUST be expressed as a 128-character uppercase hexadecimal string.

   ```
   openssl dgst -sha3-512 <artifact_filename>
   ```

3. BLAKE3 (REQUIRED for v2.0): Compute the BLAKE3 hash of the artifact file. The output MUST be expressed as a 64-character uppercase hexadecimal string.

   ```
   b3sum <artifact_filename>
   ```

The issuer MUST record all three hash values in a hash manifest file prior to proceeding to the OpenTimestamps step. The hash manifest SHOULD be a plain text file co-archival with the artifact.

All three hash values MUST be independently re-verifiable from the artifact file alone. Any discrepancy between recorded values and values computed from the artifact indicates tampering or file corruption.

6.3. OpenTimestamps Anchoring

The issuer MUST obtain an OpenTimestamps proof anchored to the Bitcoin blockchain:

1. Submit the canonical artifact to the OpenTimestamps client:

   ots stamp <artifact_filename>

   This produces a .ots proof file.

2. Wait for Bitcoin block confirmation. The OpenTimestamps proof is complete when the Merkle inclusion path is confirmed to a Bitcoin block. This typically requires one or more Bitcoin block confirmations (approximately 10 minutes per block).

3. Upgrade and verify the OTS proof:

   ots upgrade <artifact_filename>.ots
   ots verify <artifact_filename>.ots

4. Record the Bitcoin block height from the verified OTS proof. This block height is the normative blockchain anchor for the token.

The .ots proof file MUST be preserved and archived alongside the artifact in all subsequent archival steps.

6.4. DOI Archival via Zenodo

The issuer MUST archive the artifact package on Zenodo to obtain a persistent DOI:

1. Create a new Zenodo upload record.

2. Upload the following files as a package:
   a. The canonical artifact file.
   b. The .ots proof file.
   c. The hash manifest file (SHA-256, SHA3-512, BLAKE3 values).

3. Complete the Zenodo metadata form including:
   a. Title matching the artifact title.
   b. Author name, affiliation, and ORCID if available.
   c. License declaration.
   d. Description referencing the REM Protocol and blockchain anchor block height.
   e. Keywords including "REM Protocol", "RLT", "digital permanence", "blockchain timestamp", and subject-specific terms.

4. Publish the Zenodo record to obtain the DOI.

5. Record the DOI string (format: 10.5281/zenodo.XXXXXXXX) and the Zenodo record URL.

The Zenodo record is the normative archival source for the token. The DOI MUST resolve to the Zenodo record at time of issuance and is intended to remain resolvable permanently.

6.5. IPFS Pinning

Issuers SHOULD pin the artifact package to the IPFS network:

1. Add the artifact package directory to IPFS:

   ipfs add -r <artifact_directory>

2. Record the root CID returned by the ipfs add command.

3. Pin the CID to a persistent IPFS pinning service such as

Pinata, web3.storage, or a self-operated IPFS node.

4. Verify retrievability via a public IPFS gateway:

```
curl https://ipfs.io/ipfs/<CID>
```

5. Record the CID and gateway URL in the token.

IPFS pinning provides decentralized content-addressed availability independent of the Zenodo archive and adds a third permanence layer to the Dual-Layer Digital Permanence record.

6.6. Web Archival

Issuers SHOULD submit the DOI landing page and any associated public URLs to the Internet Archive Wayback Machine:

1. Submit the Zenodo record URL to: https://web.archive.org/save/<zenodo_record_url>

2. Submit the IETF Datatracker draft URL if applicable.

3. Record the resulting Wayback Machine snapshot URLs and snapshot dates.

4. Include the Wayback Machine URL in the token’s webArchival field.

6.7. REMID Namespace Registration

Issuers MAY register a REMID namespace identifier for the artifact:

1. Choose a REMID identifier string following the format: rem:<author-handle>:<artifact-short-name>:<version>

2. Register the REMID in the REM Protocol namespace registry.

3. Record the assigned REMID and registration date in the token’s remid field.

REMID registration provides a human-readable, persistent identifier for the artifact within the REM Protocol ecosystem.

6.8. Token Assembly and Publication

After completing all required anchoring steps, the issuer MUST assemble the RLT JSON token:

1. Assign a UUID v4 token identifier.

2. Populate all REQUIRED fields as defined in Section 5.1.

3. Populate all available OPTIONAL fields corresponding to completed anchoring steps.

4. Set tokenState to "ACTIVE".

5. Set predecessorToken to null for new issuances.

6. Publish the JSON token file alongside the artifact, OTS proof, and hash manifest in the Zenodo record.

7. Update the Zenodo record with the completed token if necessary.

The assembled RLT token MUST be verifiable by any party following the procedures defined in Section 7.

7. Verification Requirements

Any party MAY verify an RLT token. Verification MUST NOT require trust in the token issuer; all evidence MUST be independently retrievable from public sources. The verification process consists of five independent checks, all of which MUST pass for the token to be considered VALID.

7.1. Hash Verification

The verifier MUST:

1. Retrieve the canonical artifact file from the DOI record or IPFS CID.

2. Compute SHA-256 over the retrieved file.

3. Compare the computed SHA-256 against the hashes.sha256 field in the token. The values MUST match exactly (case-insensitive hexadecimal comparison).

4. For version 2.0 tokens: Compute SHA3-512 and BLAKE3 over the retrieved file and compare against hashes.sha3_512 and hashes.blake3 respectively.

5. If any hash comparison fails, the verification MUST be recorded as HASH_MISMATCH and the token MUST NOT be considered VALID.

A token is considered hash-verified if and only if all present hash fields match the computed values.

7.2. Blockchain Proof Verification

The verifier MUST:

1. Retrieve the .ots proof file from the URL specified in blockchain.proofUrl or from the DOI archival record.

2. Run the OpenTimestamps verification tool:

   ots verify <artifact_filename>.ots

3. Confirm that the verification output references a Bitcoin block height matching or preceding the blockchain.blockHeight value in the token.

4. If OTS verification fails, the verification MUST be recorded as OTS_VERIFICATION_FAILED.

5. If the confirmed block height does not match the token’s recorded block height, the verification MUST be recorded as BLOCK_HEIGHT_MISMATCH.

7.3. DOI Archival Verification

The verifier MUST:

1. Resolve the DOI identifier from doi.identifier using the standard DOI resolver (https://doi.org/<doi_identifier>).

2. Confirm that the DOI resolves to a publicly accessible record.

3. Confirm that the record metadata includes the artifact title and author name matching the token’s artifact and author fields.

4. Confirm that the artifact file and OTS proof file are present in the DOI record.

5. If the DOI does not resolve, the verification MUST be recorded as DOI_UNRESOLVABLE.

7.4. IPFS Availability Verification

If the token includes an ipfs.cid field, the verifier SHOULD:

1. Attempt to retrieve the artifact via the specified IPFS gateway or any public IPFS gateway using the recorded CID.

2. Verify that the retrieved content matches the hash values in the token’s hashes fields.

3. Record IPFS availability as IPFS_AVAILABLE or IPFS_UNAVAILABLE. IPFS unavailability does not constitute a token validity failure but SHOULD be noted in the verification report.

7.5. Token Integrity Verification

The verifier MUST:

1. Confirm that the token JSON is syntactically valid.

2. Confirm that all REQUIRED fields are present and non-null.

3. Confirm that tokenState is a recognized value per Section 8.1.

4. If tokenState is "REVOKED" or "SUPERSEDED", the verifier MUST note this in the verification result. A REVOKED or SUPERSEDED token is not necessarily fraudulent but indicates that the issuer has updated the token’s status.

5. If predecessorToken or successorToken fields are present and non-null, the verifier SHOULD retrieve and verify the referenced tokens to establish the full token chain.

7.6. Verification Result Codes

Implementations SHOULD report verification using the following standardized result codes:

VALID: All required verification steps passed.

HASH_MISMATCH: One or more hash values do not match the artifact file.

OTS_VERIFICATION_FAILED: The OpenTimestamps proof does not verify against the Bitcoin blockchain.

BLOCK_HEIGHT_MISMATCH: The verified OTS block height does not match the token’s recorded blockHeight.

DOI_UNRESOLVABLE: The DOI does not resolve to an accessible record.

ARTIFACT_UNAVAILABLE: The artifact file cannot be retrieved from any listed source.

TOKEN_MALFORMED: The token JSON is syntactically invalid or missing required fields.

IPFS_UNAVAILABLE: The IPFS CID is not retrievable (non-fatal if other sources are available).

STATE_REVOKED: The token is valid but has been revoked by the issuer.

STATE_SUPERSEDED: The token is valid but has been superseded by a newer token referenced in successorToken.

8. Token Lifecycle Management

8.1. Token States

Every RLT token MUST have a tokenState field with one of the following values:

ACTIVE: The token is current, valid, and in its primary issuance state. No lifecycle event has modified the token since issuance.

RENEWED: The token has been renewed by the issuer to extend its active period or update non-hash metadata. The token remains valid and its provenance record is unchanged.

SUPERSEDED: The token has been replaced by a newer token referenced in the successorToken field. The superseded token remains valid as a historical provenance record but should not be treated as the authoritative current record for the artifact.

REVOKED: The issuer has revoked the token. Revocation does not erase the blockchain or DOI records (which are permanent) but signals that the issuer no longer endorses this token as a valid provenance record. Verifiers SHOULD investigate the reason for revocation before relying on a REVOKED token.

ARCHIVED: The artifact has been intentionally retired from active use, but the provenance record is preserved for historical reference. An ARCHIVED token remains verifiable indefinitely.

8.2. Token Renewal

Token renewal is appropriate when:

- The issuer wishes to add previously missing optional fields (e.g., adding IPFS and REMID fields to a token that was issued before those layers were applied).
- The issuer has updated contact information or ORCID.
- The artifact has been reformatted without content change (e.g., converting from PDF to PDF/A).

Renewal procedure:

1. Issue a new version 2.0 token with a new UUID.
2. Recompute all hash values if the artifact file has changed. If only metadata has changed and the artifact file is identical, the original hash values are carried forward.
3. Set predecessorToken to the UUID of the original token.
4. Set tokenState to "RENEWED" if the original token is being continued, or "ACTIVE" if this is a distinct new record.
5. Update the original token’s tokenState to "SUPERSEDED" and set successorToken to the UUID of the new token.

8.3. Token Succession

Token succession occurs when an artifact is formally updated and a new anchoring is performed:

1. Produce the updated artifact.

2. Complete the full issuance procedure (Section 6) for the updated artifact, generating a new blockchain anchor, new DOI, and new hash values.

3. Issue a new version 2.0 token for the updated artifact with predecessorToken referencing the previous token's UUID.

4. Set the previous token's tokenState to "SUPERSEDED" and set its successorToken to the new token's UUID.

Token succession chains SHOULD be traversable via the predecessorToken and successorToken fields, enabling verifiers to reconstruct the complete provenance history of an artifact across multiple versions.

8.4. Token Revocation

Token revocation is appropriate when:

- The artifact was erroneously anchored to incorrect content.
- The anchor metadata (author, title, dates) contains material errors that cannot be corrected by renewal.
- Legal or compliance requirements necessitate withdrawal.

Revocation procedure:

1. Update the token's tokenState to "REVOKED".

2. Publish an updated copy of the token with the revocation state to the same publication channels used for the original token.

3. If possible, update the Zenodo record to note the revocation.

Note: Revocation does NOT erase the Bitcoin blockchain timestamp or DOI record. These are permanent. Revocation only signals the issuer's withdrawal of endorsement. The original provenance evidence remains accessible and independently verifiable regardless of the token's revocation state.

8.5. Token Archival

Tokens MAY be transitioned to ARCHIVED state when:

- The artifact has completed its active lifecycle and is no longer being actively referenced or updated.
- The issuer wishes to signal that the artifact is a historical record rather than an actively maintained document.

Archived tokens remain fully verifiable and SHOULD be preserved indefinitely.

9. Ecosystem Integration

9.1. Integration with PLPES

The Protocol Layer Prompt Engineering Specification (draft-reilly-plpes-00), authored by Lawrence John Reilly Jr.,

formally defines Protocol Layer Prompt Engineering as a discipline and specifies structured methodologies for constructing AI prompts that achieve deterministic, verifiable, and provenance-anchored outputs.

The RLT integrates with PLPES as follows:

- PLPES-governed prompt packages MAY include an embedded RLT or RLT reference to establish the provenance and authorship of the prompt engineering specification.
- AI systems that consume PLPES-compliant prompt packages SHOULD verify the embedded RLT before executing prompted procedures.
- RLT tokens anchoring PLPES-related artifacts SHOULD include "draft-reilly-plpes-00" in the ecosystem.affiliatedDrafts array.

9.2. Integration with CTS

The Cognitive Trust Stack (draft-reilly-cts-00) defines a layered trust architecture for AI reasoning systems. The CTS is itself anchored using the Dual-Layer Digital Permanence methodology, with a Zenodo DOI and Bitcoin blockchain timestamp.

The RLT integrates with CTS as follows:

- The CTS Trust Layer 1 (Provenance) relies on the RLT model as the canonical mechanism for establishing artifact provenance within the CTS trust hierarchy.
- Artifacts that participate in a CTS trust chain SHOULD carry an RLT to establish their provenance at the entry point of the chain.
- The CTS verification process MAY include RLT verification as a required step before elevating an artifact’s trust score.

9.3. Integration with AIMED and AIMED-EVAL

The AI Machine-Readable Ethics Directive (draft-reilly-aimed-00) and its evaluation framework (draft-reilly-aimed-eval-00) define machine-readable ethics specifications and compliance evaluation methodologies for AI systems.

The RLT integrates with AIMED as follows:

- AIMED directive documents SHOULD be anchored with an RLT to establish their provenance and prevent undetected modification.
- AIMED-EVAL compliance reports SHOULD reference the RLT of the AIMED directive being evaluated.
- AI systems that consume AIMED directives SHOULD verify the directive’s RLT before applying ethics constraints, ensuring that the ethics directive has not been tampered with since issuance.

9.4. Integration with UAEMF

The Universal AI Ethics and Moral Framework (draft-reilly-uaemf) defines a comprehensive ethics framework for AI systems. The UAEMF is published with a Zenodo DOI and Bitcoin blockchain timestamp, completing the trifecta of permanence recognized across the REM Protocol ecosystem.

The RLT integrates with UAEMF as follows:

- The UAEMF document is an RLT-eligible artifact and SHOULD carry an RLT anchoring its provenance.

- References to the UAEMF from other REM Protocol artifacts SHOULD include the UAEMF's RLT identifier for cryptographic traceability.

9.5. Integration with WebProof

The Web Authenticity and Content Provenance Protocol (draft-reilly-webproof-00) defines mechanisms for establishing the authenticity and provenance of web-based content, including web pages, API responses, and published documents.

The RLT integrates with WebProof as follows:

- WebProof authenticity claims for REM Protocol artifacts SHOULD reference the artifact's RLT as the root provenance anchor.

- The RLT's webArchival field, which records the Wayback Machine snapshot URL, supports WebProof's web content archival model.

- Implementations that use WebProof to assert the authenticity of a published document SHOULD verify the document's RLT as part of the WebProof verification chain.

9.6. Integration with RMRP

The Reilly Model Routing Protocol (draft-reilly-rmrp-00) defines a protocol for routing queries across heterogeneous AI model ensembles based on declared model capabilities, trust scores, and compliance profiles.

The RLT integrates with RMRP as follows:

- Model capability declarations submitted to an RMRP router SHOULD be anchored with an RLT to establish their provenance and prevent manipulation.

- RMRP trust scoring for model providers MAY include verification of RLT-anchored capability declarations as a positive trust signal.

10. Governance and Interoperability

10.1. Schema Versioning Policy

The RLT schema version is defined by the rltVersion field. The following versioning policy applies:

- Major version increments (e.g., 1.0 to 2.0) indicate backward-incompatible changes, such as the addition of REQUIRED fields.

- Minor version increments (e.g., 2.0 to 2.1) indicate backward-compatible additions, such as new OPTIONAL fields.

- Version 1.0 tokens are defined by draft-reilly-rlt-genesis-00.

- Version 2.0 tokens are defined by this document, draft-reilly-rlt-genesis-01.

Implementations MUST handle version 1.0 tokens gracefully per the backward compatibility provisions in Section 5.3.

Future schema versions will be defined in subsequent revisions of

this Internet-Draft or in successor documents.

10.2. Namespace Governance

The REMID namespace "rem-protocol" is governed by the REM Protocol specification. REMID identifiers within this namespace follow the format:

rem:<author-handle>:<artifact-short-name>:<version>

Third parties implementing the REM Protocol SHOULD use a distinct namespace prefix that does not conflict with "rem-protocol".

The DOI namespace used by the REM Protocol (10.5281/zenodo.*) is governed by Zenodo/CERN and the International DOI Foundation.

10.3. Third-Party Implementations

Any party MAY implement the RLT schema and issuance procedures defined in this specification for their own artifacts. The REM Protocol is published as an open, informational specification without proprietary restrictions.

Third-party implementations:

- MUST comply with the schema requirements in Section 5.
- MUST follow the issuance procedures in Section 6 for their tokens to be considered conforming.
- SHOULD NOT use the REMID namespace "rem-protocol" for artifacts not authored by Lawrence John Reilly Jr.
- MAY reference draft-reilly-rlt-genesis-01 as the governing specification for their tokens.
- SHOULD report implementation experience and interoperability findings to the author at lawrencejohnreilly@gmail.com.

11. Security Considerations

11.1. Hash Algorithm Security

The RLT version 2.0 uses three independent hash algorithms: SHA-256, SHA3-512, and BLAKE3. This multi-algorithm approach (REM-MAS) provides defense-in-depth against single algorithm compromise.

SHA-256 is currently considered secure against known attacks. The Bitcoin network’s security, which exceeds 700 exahashes per second as of 2025, makes SHA-256 preimage and collision attacks computationally infeasible with current technology.

SHA3-512 is a member of the SHA-3 family standardized by NIST in FIPS PUB 202. SHA3-512 is based on the Keccak sponge construction and is algorithmically independent from SHA-256, providing a second independent integrity check.

BLAKE3 is a cryptographic hash function designed for high performance and security. BLAKE3 is based on a modified Merkle tree structure and is computationally independent from both SHA-256 and SHA3-512.

For an artifact’s integrity to be compromised without detection, an attacker would need to produce a collision across all three algorithms simultaneously. Given the algorithmic independence of SHA-256, SHA3-512, and BLAKE3, this is computationally infeasible

with current and near-future technology.

11.2. Blockchain Immutability Assumptions

The security of the RLT's blockchain timestamp relies on the immutability of the Bitcoin blockchain. This immutability is guaranteed by the economic and computational cost of reorganizing the chain beyond the depth at which the OTS proof is confirmed.

For practical purposes, a Bitcoin block that is 6 or more blocks deep is considered immutable. OTS proofs that have been confirmed for an extended period (months or years) at significant block depth are extremely unlikely to be subject to reorganization.

Implementers should note that the blockchain timestamp provides a lower bound on the artifact's age (the artifact existed before the anchoring block was mined) but does not by itself establish the artifact's creation date. The DOI archival record provides additional corroborating date evidence.

In the event of a catastrophic failure of the Bitcoin network (an extreme and unlikely scenario), the DOI and IPFS records remain as independent permanence evidence. This is the core motivation for the Dual-Layer Digital Permanence approach.

11.3. DOI Archival Persistence

Zenodo is operated by CERN with a commitment to long-term preservation. Zenodo DOIs issued through the DataCite DOI registration agency carry institutional backing. However, implementers should note that no archival system carries an absolute permanence guarantee.

The RLT's multi-layer approach mitigates this risk. If the Zenodo record becomes unavailable, the IPFS pin and Wayback Machine snapshot provide alternative content retrieval paths. The blockchain timestamp remains verifiable independently of Zenodo.

Implementers who require the highest assurance of archival persistence SHOULD maintain a local copy of the artifact package (artifact file, OTS proof, hash manifest, and RLT token) in addition to the published archival copies.

11.4. Author Identity Assurance

The RLT binds a hash to a blockchain timestamp and DOI record, but does not inherently authenticate the identity of the token issuer. The author.name and author.orcid fields in the token are self-asserted.

Identity assurance is provided by:

- ORCID identifiers, which are issued by a controlled registration process and may be linked to institutional affiliations.
- IETF Internet-Draft authorship, which is publicly recorded in the IETF Datatracker.
- Consistent use of the same email address, ORCID, and DOI registrant identity across multiple tokens.
- The optional signature field in the token schema, which MAY contain a digital signature over the token content using a key pair whose public key is independently published.

Future revisions of this specification MAY define a normative

digital signature mechanism for token integrity and author authentication.

11.5. Post-Quantum Considerations

SHA-256, SHA3-512, and BLAKE3 are all believed to be secure against quantum computer attacks using Grover's algorithm, which provides at most a quadratic speedup in preimage search. SHA-256's effective security against quantum attacks is approximately 128 bits, which is considered adequate for long-term use.

OpenTimestamps uses SHA-256 in its Merkle tree construction. The Bitcoin network's SHA-256 based proof-of-work is also believed to be secure against quantum attacks at current quantum computing capability levels.

The optional signature field in the token schema MAY be used with post-quantum signature algorithms such as SPHINCS+ (defined in NIST FIPS 205) to provide post-quantum author authentication. The REM Multi-Algorithm Stack (REM-MAS) includes SPHINCS+ as an optional post-quantum signature algorithm suite.

Implementers who require long-term (20+ year) security guarantees SHOULD evaluate post-quantum signature algorithms for the token signature field and SHOULD follow NIST post-quantum cryptography standards as they are finalized.

11.6. Token Forgery and Replay Attacks

An attacker who copies a legitimate RLT token and substitutes a different artifact will be detected during hash verification (Section 7.1), provided the verifier retrieves the artifact from an independent source (DOI or IPFS) rather than from the attacker.

An attacker who modifies the token JSON (e.g., changing the author field) will be detected if the token carries a digital signature (signature field) that covers the modified fields.

Replay attacks (presenting a legitimate old token as evidence for a newer artifact) are mitigated by the blockchain timestamp, which provides a maximum age bound. The blockchain anchor block height can be independently verified to establish when the hash was first committed to the chain.

Implementers SHOULD verify the full token chain (predecessorToken and successorToken links) when the most current version of an artifact's provenance record is required, rather than relying on any single token in isolation.

11.7. Supply Chain Integrity

The RLT model can be applied to software supply chain integrity by anchoring software release artifacts, build manifests, and dependency declarations. In this context, the security considerations of Sections 11.1 through 11.6 apply.

In addition, software supply chain applications SHOULD:

- Anchor each release artifact independently rather than only the source repository state.
- Include build environment metadata in the hash manifest to enable reproducible build verification.
- Use the token succession mechanism (Section 8.3) to maintain a verifiable chain of custody across software versions.

12. IANA Considerations

This document has no IANA actions. A future revision MAY request registration of RLT-related media types or URI schemes if the REM Protocol ecosystem warrants it.

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

13.2. Informative References

[REM-PROTOCOL] Reilly, L. J., "Reilly EternaMark Protocol (REM)", Internet-Draft draft-reilly-rem-protocol-00, November 2025, <https://datatracker.ietf.org/doc/draft-reilly-rem-protocol/>.

[RLT-GENESIS-00] Reilly, L. J., "REM License Token (RLT) - Genesis Artifact", Internet-Draft draft-reilly-rlt-genesis-00, November 2025, <https://datatracker.ietf.org/doc/draft-reilly-rlt-genesis/>.

[PLPES] Reilly, L. J., "Protocol Layer Prompt Engineering Specification", Internet-Draft draft-reilly-plpes-00, 2025, <https://datatracker.ietf.org/doc/draft-reilly-plpes/>.

[RMRP] Reilly, L. J., "Reilly Model Routing Protocol", Internet-Draft draft-reilly-rmrp-00, 2025, <https://datatracker.ietf.org/doc/draft-reilly-rmrp/>.

[CTS] Reilly, L. J., "Cognitive Trust Stack", Internet-Draft draft-reilly-cts-00, 2025, <https://datatracker.ietf.org/doc/draft-reilly-cts/>.

[AIMED] Reilly, L. J., "AI Machine-Readable Ethics Directive", Internet-Draft draft-reilly-aimed-00, 2025, <https://datatracker.ietf.org/doc/draft-reilly-aimed/>.

[AIMED-EVAL] Reilly, L. J., "AI Machine-Readable Ethics Directive Evaluation Framework", Internet-Draft draft-reilly-aimed-eval-00, 2025, <https://datatracker.ietf.org/doc/draft-reilly-aimed-eval/>.

[UAEMF] Reilly, L. J., "Universal AI Ethics and Moral Framework", Internet-Draft draft-reilly-uaemf-00, 2025, <https://datatracker.ietf.org/doc/draft-reilly-uaemf/>.

[WEBPROOF] Reilly, L. J., "Web Authenticity and Content Provenance Protocol", Internet-Draft draft-reilly-webproof-00, 2025, <https://datatracker.ietf.org/doc/

draft-reilly-webproof/>.

[RBIP] Reilly, L. J., "Reilly Banking Integrity Protocol", Internet-Draft draft-reilly-rbip-00, 2025, <https://datatracker.ietf.org/doc/draft-reilly-rbip/>.

[RGIP] Reilly, L. J., "Reilly Government Integrity Protocol", Internet-Draft draft-reilly-rgip-00, 2025, <https://datatracker.ietf.org/doc/draft-reilly-rgip/>.

[ZENODO-RLT] Reilly, L. J., "REM License Token - Genesis Artifact", Zenodo, DOI 10.5281/zenodo.17438760, November 2025, <https://zenodo.org/records/17438760>.

[OTS] Todd, P., "OpenTimestamps: Scalable, Trust-Minimized, Distributed Timestamping with Bitcoin", <https://opentimestamps.org>.

[IPFS] Protocol Labs, "IPFS - Content Addressed, Versioned, P2P File System", <https://ipfs.io>.

[FIPS202] National Institute of Standards and Technology, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", FIPS PUB 202, August 2015, <https://doi.org/10.6028/NIST.FIPS.202>.

[FIPS205] National Institute of Standards and Technology, "Stateless Hash-Based Digital Signature Standard", FIPS PUB 205, August 2024, <https://doi.org/10.6028/NIST.FIPS.205>.

[BLAKE3] O'Connor, J., Aumasson, J., Neves, S., and Z. Wilcox-O'Hearn, "BLAKE3 one function, fast everywhere", <https://github.com/BLAKE3-team/BLAKE3-specs>.

[BCP47] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

Appendix A. Example RLT Token v2.0

The following is a complete example of an RLT version 2.0 token for a hypothetical artifact. All hash values are illustrative.


```
{
  "$schema": "https://rem-protocol.org/schema/rlt/v2.0",
  "rltVersion": "2.0",
  "tokenId": "a1b2c3d4-e5f6-4a7b-8c9d-e0f1a2b3c4d5",
  "schemaDate": "2026-05-14",

  "author": {
    "name": "Lawrence John Reilly Jr",
    "orcid": "https://orcid.org/0000-0000-0000-0000",
    "affiliation": null,
    "email": "lawrencejohnreilly@gmail.com"
  },

  "artifact": {
    "title": "Example REM Protocol Governed Artifact v1.0",
    "type": "specification",
    "filename": "example-rem-artifact-v1.0.pdf",
    "mimeType": "application/pdf",
    "language": "en"
  },
```

```
"license": "CC-BY-4.0",

"hashes": {
  "sha256":
    "9964A78C6FC33794EF840ED69045C5C2477BC611CBC73EF6EC537FACA4C7BB74",
  "sha3_512":
    "B2F4A8C1E3D5F7A9B2C4D6E8F0A2B4C6D8E0F2A4B6C8D0E2F4A6B8C0D2E4F6A8
     B0C2D4E6F8A0B2C4D6E8F0A2B4C6D8E0F2A4B6C8D0E2F4A6B8C0D2E4F6A8B0",
  "blake3":
    "7C4E2A8F1D6B3E9C5A7F2D4B8E6A3C1F9D7B5E2A4C8F6D3B1E9A7C5F4D2B0E8"
},

"hashManifestUrl":
  "https://zenodo.org/records/XXXXXXXX/files/hash-manifest.txt",

"dates": {
  "artifactCreated": "2026-05-14",
  "tokenIssued": "2026-05-14",
  "blockchainAnchor": "2026-05-10",
  "doiRegistered": "2026-05-14",
  "nextReviewDue": "2027-05-14"
},

"blockchain": {
  "chain": "Bitcoin",
  "blockHeight": 897234,
  "blockHash":
    "00000000000000000023A8F4C1B7E9D2F6A3C5B8E1D4A7F0C2B5E8A3D6F9C2",
  "timestamp": "2026-05-10T14:23:00Z",
  "proofFile": "example-rem-artifact-v1.0.pdf.ots",
  "proofUrl":
    "https://zenodo.org/records/XXXXXXXX/files/
     example-rem-artifact-v1.0.pdf.ots"
},

"doi": {
  "identifier": "10.5281/zenodo.XXXXXXXX",
  "url": "https://zenodo.org/records/XXXXXXXX",
  "registrar": "Zenodo"
},

"ipfs": {
  "cid": "bafybeigdyrzt5sfp7udm7hu76uh7y26nf3efuylqabf3oclgtqy55fbzdi",
  "gateway": "https://ipfs.io/ipfs/bafybeigdyrzt5sfp7udm7hu76uh7y26
              nf3efuylqabf3oclgtqy55fbzdi",
  "pinnedAt": "2026-05-14"
},

"webArchival": {
  "waybackUrl": "https://web.archive.org/web/20260514000000/
                 https://zenodo.org/records/XXXXXXXX",
  "archivedAt": "2026-05-14"
},

"remid": {
  "identifier": "rem:reilly:example-artifact:1.0",
  "namespace": "rem-protocol",
  "registeredAt": "2026-05-14"
},

"tokenState": "ACTIVE",
"predecessorToken": null,
"successorToken": null,

"ecosystem": {
```

```
      "protocol": "REM",
      "protocolDraft": "draft-reilly-rem-protocol-00",
      "affiliatedDrafts": [
        "draft-reilly-rlt-genesis-01",
        "draft-reilly-plpes-00"
      ],
      "complianceProfiles": []
    },

    "verification": {
      "verificationUrl": null,
      "publicVerifierEndpoint": null
    },

    "signature": {
      "type": null,
      "value": null,
      "keyId": null
    }
  }
```

Appendix B. Example RLT Token v1.0 (Genesis Record)

The following is the original version 1.0 genesis token as published in draft-reilly-rlt-genesis-00, preserved here for historical reference.

```
  {
    "rltVersion": "1.0",
    "tokenId": "example-uuid-0000",
    "author": {
      "name": "Lawrence J. Reilly"
    },
    "license": "CC_BY_4.0",
    "hash":
      "9964A78C6FC33794EF840ED69045C5C2477BC611CBC73EF6EC537FACA4C7BB74",
    "metadataDate": "2025-11-16",
    "blockchain": {
      "chain": "Bitcoin",
      "blockHeight": 914168,
      "timestamp": "2025-09-10T00:00:00Z",
      "proofFile": "RLT_FIRST_TOKEN_FULL_GUIDE_v2.pdf.ots"
    },
    "doi": "10.5281/zenodo.17438760",
    "issuedDate": "2025-11-16",
    "tokenUrl": "https://zenodo.org/records/17438760"
  }
```

Appendix C. REM-MAS Algorithm Suite Reference

The REM Multi-Algorithm Stack (REM-MAS) defines the following algorithm suites for use across the REM Protocol ecosystem:

Suite A (Standard):
   Hash algorithms: SHA-256, SHA3-512, BLAKE3
   Signature: None (provenance only)
   Use case: Standard artifact provenance anchoring.

Suite B (Signed):
   Hash algorithms: SHA-256, SHA3-512, BLAKE3
   Signature: Ed25519
   Use case: Artifact provenance with author authentication.

Suite C (Post-Quantum):

Hash algorithms: SHA-256, SHA3-512, BLAKE3
Signature: SPHINCS+ (NIST FIPS 205)
Use case: Long-term provenance with post-quantum signature.

Suite D (Government/Enterprise):
Hash algorithms: SHA-256, SHA3-512, BLAKE3
Signature: SPHINCS+ or ECDSA P-384
Additional: Hardware Security Module (HSM) key custody
Use case: High-assurance provenance for regulated environments.

All RLT tokens SHOULD specify which REM-MAS suite they were issued under in a future token schema extension. Version 2.0 tokens that include a signature SHOULD note the algorithm in the signature.type field.

Appendix D. Implementation Notes

D.1. Tool Dependencies

The following open-source tools are RECOMMENDED for RLT issuance:

- OpenTimestamps Client: https://github.com/opentimestamps/opentimestamps-client
  For OTS stamping and verification.
- BLAKE3 (b3sum): https://github.com/BLAKE3-team/BLAKE3
  For BLAKE3 hash computation.
- OpenSSL (>= 1.1.1): https://www.openssl.org
  For SHA3-512 computation (openssl dgst -sha3-512).
- IPFS Kubo: https://github.com/ipfs/kubo
  For IPFS content addressing and pinning.

D.2. Hash Manifest Format

The RECOMMENDED hash manifest format is a plain text file with one hash per line, in the following format:

```
SHA-256: <64-char uppercase hex>
SHA3-512: <128-char uppercase hex>
BLAKE3: <64-char uppercase hex>
Filename: <canonical artifact filename>
ManifestDate: <ISO 8601 date>
```

D.3. Zenodo Metadata Template

When creating a Zenodo record for an RLT-anchored artifact, the following keyword tags SHOULD be included:

```
REM Protocol, RLT, digital permanence, blockchain timestamp,
OpenTimestamps, Bitcoin, dual-layer permanence, Zenodo DOI,
Lawrence Reilly, IETF Internet-Draft
```

D.4. Browser-Only Issuance Workflow

Implementers who do not have access to a command-line environment MAY complete a browser-only issuance workflow using the following web-accessible tools:

- SHA-256: https://emn178.github.io/online-tools/sha256_checksum.html
- OpenTimestamps (web): https://opentimestamps.org (web interface)
- Zenodo: https://zenodo.org (browser upload)
- Pinata IPFS: https://pinata.cloud (browser upload)
- Wayback Machine: https://web.archive.org/save

SHA3-512 and BLAKE3 computation in a browser-only environment SHOULD use a reputable online hash calculator or a locally served WebAssembly implementation of these algorithms.

Appendix E. Change Log

draft-reilly-rlt-genesis-01

- Token schema updated to version 2.0 with mandatory multi-algorithm hash fields (sha3_512, blake3).
- Added hashes object structure replacing single hash field.
- Added hashManifestUrl field.
- Added dates object with granular date fields.
- Added blockchain.blockHash and blockchain.proofUrl fields.
- Added doi object structure replacing flat doi string.
- Added ipfs, webArchival, and remid objects for additional permanence layers.
- Added tokenState, predecessorToken, successorToken lifecycle fields.
- Added ecosystem object for REM Protocol ecosystem integration.
- Added verification and signature objects.
- New Section 3.3: The REM Protocol Ecosystem, documenting the fourteen-draft ecosystem.
- New Section 4.1: Commercial Licensing Terms, establishing the perpetual 2.5% worldwide royalty obligation to Lawrence John Reilly Jr for commercial use of RLT-anchored artifacts.
- New Section 4.3: Multi-Algorithm Hash Record.
- New Section 5.3: Backward Compatibility with Version 1.0.
- Expanded Section 6: Issuance procedures expanded to eight subsections covering pre-issuance, REM-MAS hashing, OTS anchoring, DOI archival, IPFS pinning, web archival, REMID registration, and token assembly.
- Expanded Section 7: Verification procedures expanded to six subsections with standardized result codes.
- New Section 8: Token Lifecycle Management (states, renewal, succession, revocation, archival).
- New Section 9: Ecosystem Integration (PLPES, CTS, AIMED, UAEMF, WebProof, RMRP).
- New Section 10: Governance and Interoperability.
- Expanded Section 11: Security Considerations expanded to seven subsections including post-quantum, token forgery, replay attacks, and supply chain integrity.
- New Appendix C: REM-MAS Algorithm Suite Reference.

- New Appendix D: Implementation Notes.
- Updated references to include all fourteen active REM Protocol ecosystem Internet-Drafts.
- Updated expiry date to November 14, 2026.

draft-reilly-rlt-genesis-00

- Initial version defining the RLT genesis artifact with SHA-256 hash, Bitcoin blockchain anchor at block height 914168, and Zenodo DOI 10.5281/zenodo.17438760.

Author's Address

Lawrence John Reilly Jr
Email: lawrencejohnreilly@gmail.com