Network Working Group
Internet-Draft
Intended status: Informational
Expires: 20 September 2026

L.J. Reilly
Independent

March 2026

Reilly EternaMark (REM) Protocol - Dual-Layer Digital Permanence Using DOI Archiving and Blockchain Timestamping
draft-reilly-rem-protocol-01

Abstract

The Reilly EternaMark (REM) Protocol defines a dual-layer method for digital permanence through the integration of Digital Object Identifiers (DOIs) and blockchain timestamping. The protocol ensures digital artifacts are permanently identifiable, immutable, and verifiable for both present and future use.

This revision (-01) extends the -00 specification with a formal REM Record structure, a machine-readable artifact manifest format, a verification procedure, implementation guidance, and expanded security and interoperability considerations. It also corrects an erroneous normative reference present in -00.

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 20 September 2026.

Table of Contents

1. Introduction

Existing digital preservation methods are largely centralized and therefore vulnerable to alteration, corruption, administrative failure, or deliberate suppression. Timestamping services dependent on a single authority introduce a single point of failure; DOI registries, while durable, do not alone provide cryptographic proof of a document’s content at a specific moment in time.

The REM Protocol addresses these shortcomings by combining the discoverability and persistence of the DOI system with the cryptographic immutability of blockchain timestamping into a single composable permanence primitive.

The resulting dual-layer record:

* Proves that a specific artifact existed at a specific time, with content integrity verifiable without reliance on any third party.

* Remains discoverable via globally resolvable DOI infrastructure independent of any single hosting provider.

* Produces a machine-readable REM Record linking hash, blockchain attestation, and DOI into a single verifiable unit.

The protocol is intentionally blockchain-agnostic and DOI-provider-agnostic, enabling broad adoption across academic, legal, commercial, and governmental contexts.

This document extends the -00 specification [draft-reilly-rem-protocol-00] with a formal artifact manifest format, a structured verification procedure, implementation considerations, and an expanded security analysis. It also corrects

an erroneous informative reference in -00 where RFC 9162 (Certificate Transparency Version 2.0) was mistakenly cited in place of the OpenTimestamps specification [OpenTimestamps].

2. Terminology

The following terms are used throughout this document.

Artifact:
: Any digital object (document, dataset, source code, model weight file, contract, etc.) whose existence at a given point in time is to be permanently recorded using the REM Protocol.

DOI (Digital Object Identifier):
: A globally unique, resolvable, and persistent identifier for digital objects, standardized in ISO 26324 [ISO-26324] and administered by the International DOI Foundation.

REM Record:
: The machine-readable output of a completed REM Protocol operation, linking a SHA-256 artifact hash, a blockchain TXID or block reference, and a DOI into a single verifiable unit. The canonical serialization format is defined in Section 6.

SHA-256 Hash:
: A cryptographic digest of the artifact computed using the Secure Hash Algorithm 2 (256-bit output) as specified in [FIPS-180-4]. The hash serves as a content-addressable fingerprint of the artifact.

Blockchain Timestamp:
: A proof of existence record anchored to a specific block in a distributed ledger, providing cryptographic attestation that the artifact hash existed no later than the block’s confirmation time.

TXID:
: A transaction identifier on a blockchain network uniquely identifying the transaction carrying the artifact hash or its Merkle-tree aggregation.

OpenTimestamps (OTS):
: An open standard for blockchain timestamping that aggregates multiple document hashes into a Merkle tree and anchors the root hash to the Bitcoin blockchain [OpenTimestamps].

Dual-Layer Permanence:
: The original term, introduced in this protocol suite, describing the combined use of DOI registration and blockchain timestamping to achieve simultaneously discoverable and cryptographically immutable digital permanence.

3. Requirements Language

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.

4. Protocol Overview

The REM Protocol consists of four sequential steps:

1. Hashing - Compute a SHA-256 digest of the digital artifact.

2. Blockchain Timestamping - Submit the hash to a blockchain timestamping service and obtain a confirmed block reference or TXID.

3. DOI Assignment - Register the artifact with a DOI-issuing repository and obtain a persistent DOI.

4. REM Record Publication - Publish the hash, TXID/block reference, and DOI together as a machine-readable REM Record, constituting the permanent dual-layer provenance record.

Steps 1 and 2 MUST be completed before Step 3. The DOI registration metadata SHOULD include the blockchain TXID or block reference as a custom metadata field to enable cross-referencing from within the DOI infrastructure itself.

5. Specification

This section provides normative detail for each of the four protocol steps.

5.1. Step 1: Artifact Hashing

The implementer MUST compute a SHA-256 hash of the canonical binary representation of the artifact prior to any other protocol steps.

For textual documents, the canonical representation is the UTF-8 encoded byte sequence of the document in its final, intended form. No normalization (e.g., line-ending conversion or whitespace removal) MAY be applied after the hash has been computed and recorded, as any such transformation would invalidate the hash.

The resulting digest MUST be encoded as a lowercase hexadecimal string of exactly 64 characters.

Example:

```
  Input:  "Universal AI Ethics & Moral Framework v1.0" (UTF-8 PDF)
  Output: 02439f01ba0b805ba3011aaeb2783ffb096c1c6ad847876e75...
          [truncated for illustration]
```

Implementations SHOULD also record the artifact’s byte length at the time of hashing to assist in detecting accidental truncation or corruption during later transmission.

5.2. Step 2: Blockchain Timestamping

The artifact hash obtained in Section 5.1 MUST be submitted to a blockchain timestamping service. The submission MUST result in a confirmed on-chain record prior to DOI registration.

Implementations SHOULD use the OpenTimestamps protocol [OpenTimestamps], which aggregates multiple hashes into a Merkle tree and anchors the root to the Bitcoin blockchain. This approach provides high security guarantees (inheriting Bitcoin’s proof-of-work) while minimizing on-chain footprint.

Alternatively, direct on-chain transaction embedding (e.g., via Bitcoin OP_RETURN) or timestamping on other sufficiently secure public ledgers MAY be used, provided the chain selection is documented in the REM Record.

Upon confirmation, the implementer MUST record one of the following:

* For OpenTimestamps: the .ots calendar attestation file and the Bitcoin block height at which the attestation was confirmed.

* For direct on-chain embedding: the TXID and the block height.

A minimum confirmation depth of six (6) blocks is RECOMMENDED for Bitcoin-anchored timestamps before proceeding to Step 3, to guard against blockchain reorganization (see Section 10.2).

Example (from the original REM Protocol archiving operation):

```
Blockchain:    Bitcoin
Block Height:  914168
Block Date:    2025-09-10 (EST)
Method:        OpenTimestamps
```

5.3. Step 3: DOI Assignment

The artifact MUST be deposited with a DOI-issuing repository. Zenodo [ZenodoStandards] is the RECOMMENDED repository due to its integration of DataCite DOIs, open-access mandate, and long-term CERN infrastructure backing. Other DataCite or Crossref member repositories MAY be used.

The DOI registration metadata MUST include:

* Artifact title and version.
* Author name(s).
* Publication or deposit date.
* The SHA-256 hash from Section 5.1 (in the description or a custom metadata field).

The DOI registration metadata SHOULD include:

* The blockchain TXID or block reference from Section 5.2.
* The name of the blockchain network used.
* A reference to this specification (draft-reilly-rem-protocol).

The resulting DOI MUST be recorded for inclusion in the REM Record.

Example:

```
Repository:  Zenodo
DOI:         10.5281/zenodo.17096230
Resolver:    https://doi.org/10.5281/zenodo.17096230
```

5.4. Step 4: REM Record Publication

Upon successful completion of Steps 1 through 3, the implementer MUST assemble and publish a REM Record as defined in Section 6.

The REM Record MUST be made publicly accessible via at least one of:

* Inclusion in the DOI-registered artifact or its landing page metadata.

* Publication to a persistent public archive (e.g., Internet Archive Wayback Machine) with the artifact URL noted in the Record.

* Inline embedding within the artifact itself (e.g., as a metadata appendix or document footer).

6. REM Record Format

This section defines the canonical machine-readable format for a REM Record. The format is intentionally minimal to maximize compatibility with existing metadata pipelines.

6.1. JSON Manifest

The RECOMMENDED serialization of a REM Record is a JSON object conforming to the following schema. Fields marked REQUIRED MUST be present; fields marked OPTIONAL MAY be omitted.

```
{
  "rem_version": "1.1",
  "artifact": {
    "title":        <string, REQUIRED>,
    "version":      <string, OPTIONAL>,
    "author":       <string or array of strings, REQUIRED>,
    "description":  <string, OPTIONAL>,
    "byte_length":  <integer, OPTIONAL>
  },
  "hash": {
    "algorithm":    "SHA-256",
    "value":        <64-char lowercase hex string, REQUIRED>
  },
  "blockchain": {
    "network":      <string, REQUIRED>,
    "method":       <"opentimestamps" | "op_return" | "other",
                     REQUIRED>,
    "txid":         <string, OPTIONAL>,
    "block_height": <integer, REQUIRED>,
    "block_date":   <ISO 8601 date string, REQUIRED>,
    "ots_file_uri": <URI string, OPTIONAL>
  },
  "doi": {
    "value":        <DOI string, REQUIRED>,
    "resolver":     <URI, OPTIONAL>,
    "repository":   <string, OPTIONAL>
  },
  "rem_record": {
    "created":      <ISO 8601 datetime, REQUIRED>,
    "protocol_ref": "draft-reilly-rem-protocol-01",
    "archive_uris": [<array of URI strings>, OPTIONAL]
  }
}
```

The "rem_version" field MUST be set to "1.1" for records produced under this (-01) specification. Records produced under -00 MAY use "1.0" for backward-compatibility identification.

The following is a complete example REM Record JSON manifest:

```
{
  "rem_version": "1.1",
  "artifact": {
    "title": "Reilly EternaMark (REM) Protocol",
    "version": "draft-reilly-rem-protocol-01",
```

```
      "author": "Lawrence John Reilly Jr.",
      "description": "Dual-layer digital permanence specification.",
      "byte_length": 48291
    },
    "hash": {
      "algorithm": "SHA-256",
      "value": "02439f01ba0b805ba3011aaeb2783ffb096c1c6ad847876e75"
    },
    "blockchain": {
      "network": "Bitcoin",
      "method": "opentimestamps",
      "block_height": 914168,
      "block_date": "2025-09-10",
      "ots_file_uri": "https://zenodo.org/records/17096230/files/
                       rem-protocol.ots"
    },
    "doi": {
      "value": "10.5281/zenodo.17096230",
      "resolver": "https://doi.org/10.5281/zenodo.17096230",
      "repository": "Zenodo"
    },
    "rem_record": {
      "created": "2026-03-20T00:00:00Z",
      "protocol_ref": "draft-reilly-rem-protocol-01",
      "archive_uris": [
        "https://web.archive.org/web/2025/https://zenodo.org/
         records/17096230"
      ]
    }
  }
```


6.2. Field Definitions

rem_version:
: Identifies the REM Record schema version. Enables parsers to apply the correct field semantics.

artifact.title:
: Human-readable name of the archived artifact.

artifact.version:
: Version string of the artifact, if applicable (e.g., "v1.0", "draft-00").

artifact.author:
: Name or names of the artifact's author(s). MAY be a single string or a JSON array of strings.

artifact.byte_length:
: Size of the artifact in bytes at the moment the hash was computed. Assists in detecting corruption or truncation.

hash.algorithm:
: MUST be "SHA-256" for records conforming to this specification. Future revisions MAY introduce additional algorithm identifiers.

hash.value:
: The full SHA-256 digest in lowercase hexadecimal, exactly 64 characters.

blockchain.network:
: Name of the blockchain network (e.g., "Bitcoin", "Ethereum"). Implementations SHOULD prefer networks with substantial proof-

of-work or proof-of-stake security history.

blockchain.method:
Timestamping mechanism used. "opentimestamps" refers to the OpenTimestamps aggregation and Bitcoin anchoring protocol [OpenTimestamps]. "op_return" refers to direct embedding via a Bitcoin OP_RETURN output. "other" MUST be accompanied by a human-readable description in the artifact.description field.

blockchain.txid:
The transaction identifier, if applicable to the chosen method.

blockchain.block_height:
The confirmed block height at which the timestamp attestation is anchored.

blockchain.block_date:
Calendar date of the confirmed block in ISO 8601 format (YYYY-MM-DD).

blockchain.ots_file_uri:
URI from which the .ots proof file may be retrieved for independent verification.

doi.value:
The DOI string in its canonical form, e.g., "10.5281/zenodo.17096230".

doi.resolver:
The HTTPS URI of the DOI resolver, typically "https://doi.org/10.5281/zenodo.17096230".

doi.repository:
Human-readable name of the repository that issued the DOI, e.g., "Zenodo".

rem_record.created:
ISO 8601 datetime at which this REM Record was assembled.

rem_record.protocol_ref:
MUST contain the draft name of the REM Protocol version under which this record was produced (e.g., "draft-reilly-rem-protocol-01").

rem_record.archive_uris:
Array of URIs at which this REM Record or the artifact itself has been independently archived (e.g., Internet Archive snapshot URLs).

7. Verification Procedure

Any party wishing to verify a REM Record MUST follow the procedure in this section. Verification is fully independent and requires no trust in the original publisher.

7.1. Hash Verification

1. Obtain the original artifact.

2. Compute its SHA-256 hash using the same canonical representation described in Section 5.1.

3. Compare the computed hash to the value in hash.value of the

REM Record.

4. If the hashes match, hash integrity is CONFIRMED. If they do not match, the artifact has been modified after timestamping and the record is INVALID for that artifact.

7.2. Blockchain Verification

For OpenTimestamps records:

1. Retrieve the .ots proof file from blockchain.ots_file_uri.

2. Run the OpenTimestamps verification tool against the artifact hash and the .ots file.

3. Confirm that the tool reports a valid Bitcoin block attestation at or before blockchain.block_height.

For direct on-chain records:

1. Look up blockchain.txid in a public blockchain explorer for the network identified in blockchain.network.

2. Confirm that the transaction output contains a value matching hash.value (or a Merkle path leading to it).

3. Confirm that the transaction is in block blockchain.block_height or earlier.

If the on-chain attestation matches the hash and precedes or equals the stated block height, blockchain attestation is CONFIRMED.

7.3. DOI Resolution Verification

1. Resolve doi.resolver using an HTTPS client.

2. Confirm that the DOI resolves to an accessible landing page or artifact download.

3. If the landing page metadata includes the SHA-256 hash or TXID recorded in the REM Record, cross-referencing is CONFIRMED.

DOI resolution verifies discoverability and persistence; it does NOT independently verify content integrity, which is the role of hash verification in Section 7.1.

7.4. Full Verification Outcome

A REM Record is considered FULLY VERIFIED if and only if:

* Hash verification (Section 7.1) confirms artifact integrity.

* Blockchain verification (Section 7.2) confirms the hash existed no later than the stated block timestamp.

* DOI resolution (Section 7.3) confirms the artifact remains publicly discoverable.

A record may be PARTIALLY VERIFIED if one or two of the three checks pass while the remaining check(s) are inconclusive due to temporary infrastructure unavailability. A PARTIALLY VERIFIED record SHOULD NOT be treated as failed pending resolution of the infrastructure issue.

A record is INVALID if hash verification fails, indicating content modification after the timestamp was established.

8. Implementation Considerations

8.1. Toolchain Recommendations

The following open-source toolchain is RECOMMENDED for REM Protocol implementations:

Hashing:
The sha256sum utility (available on Linux, macOS, and Windows Subsystem for Linux) or any FIPS 180-4 compliant SHA-256 implementation.

Blockchain Timestamping:
The OpenTimestamps client, invoked as:

```
ots stamp <artifact-file>
ots upgrade <artifact-file>.ots
ots verify <artifact-file>.ots
```

The client is available at https://opentimestamps.org.

DOI Registration:
The Zenodo REST API or web deposit interface at https://zenodo.org. Zenodo sandbox at https://sandbox.zenodo.org SHOULD be used for testing prior to production deposits.

REM Record Generation:
Any JSON serializer producing output conforming to the schema in Section 6.1. Implementations SHOULD validate against the schema before publishing.

8.2. Timing and Ordering Constraints

The ordering of operations is not merely procedural; it carries legal and evidentiary significance:

* The blockchain timestamp establishes the legally cognizable earliest possible creation date of the artifact. Performing DOI registration before the blockchain timestamp is confirmed means the DOI record cannot be cited as independently corroborating the timestamp.

* For OpenTimestamps, calendar servers provide immediate attestation but Bitcoin block confirmation requires approximately 10 minutes per block. Implementers MUST wait for on-chain confirmation before treating the timestamp as finalized.

* Zenodo issues DOIs immediately upon deposit; however, the deposit metadata may be updated after issuance. Implementers SHOULD finalize the artifact and its hash before depositing to avoid version discrepancies between the hash and the deposited artifact.

8.3. Versioning and Amendments

If an artifact is revised after a REM Record has been published:

* The original REM Record MUST NOT be modified or deleted.

* A new, independent REM Protocol operation MUST be performed for the revised artifact.

* The new REM Record SHOULD reference the original record's DOI in the rem_record.archive_uris field or in the artifact description, establishing an explicit provenance chain between versions.

This policy ensures that the historical record of all artifact versions remains intact and independently verifiable.

9. Interoperability

9.1. Relationship to Other Reilly Protocol Suite Members

The REM Protocol serves as the foundational permanence primitive for the broader Reilly Protocol Suite. Other suite members reference and build upon REM as follows:

* The Cognitive Trust Stack (CTS) [draft-reilly-cts-00] uses REM as its Layer 3 (Provenance Layer) mechanism for anchoring AI behavioral attestation records, enabling third parties to verify that an AI system's published alignment claims correspond to specific, immutably timestamped behavioral evidence.

* The REM License Token (RLT) Genesis specification [draft-reilly-rlt-genesis] uses REM to timestamp its own specification, creating a self-referential provenance record in which the token that represents REM-anchored rights is itself protected by the REM Protocol.

* The Reilly Sentinel Protocol (RSP), Reilly Resilience Protocol (RRP), Reilly Banking Integrity Protocol (RBIP), and Reilly Government Integrity Protocol (RGIP) all designate REM as their canonical archiving and provenance-anchoring mechanism.

Implementers building on any member of the Reilly Protocol Suite SHOULD implement REM first, as it underpins all cross-suite provenance and integrity guarantees.

9.2. Blockchain Agnosticism

While Bitcoin is the RECOMMENDED blockchain due to its security history, proof-of-work finality, and broad public verifiability, the REM Protocol is explicitly blockchain-agnostic.

Implementers MAY use Ethereum, Polygon, or other public ledgers provided:

* The chosen network has been operational for a minimum of three years at the time of timestamping.

* Block timestamps on the chosen network are publicly verifiable via multiple independent explorers.

* The implementer documents the network name, consensus mechanism, and explorer references in the REM Record.

10. Security Considerations

10.1. Hash Integrity

The SHA-256 algorithm provides 128 bits of collision resistance under current cryptanalytic understanding [FIPS-180-4]. No practical collision attacks against SHA-256 are known at the time of this writing.

Implementers concerned with long-term or post-quantum integrity SHOULD record the artifact under multiple hash algorithms (e.g., SHA-256 and SHA3-256) in parallel and include both values in the REM Record. A future revision of this specification MAY formalize multi-algorithm hashing.

10.2. Blockchain Reorganization Risk

All proof-of-work blockchains are subject to probabilistic finality. A block that appears confirmed may be reorganized out of the canonical chain if a competing chain of greater cumulative work is published. For Bitcoin:

* Reorganizations of depth greater than six blocks are historically exceedingly rare.

* Implementers SHOULD wait for a minimum of six confirmations before treating a Bitcoin-anchored timestamp as final.

* For artifacts of exceptional legal or financial significance, waiting for 100 confirmations (approximately 16.7 hours) is RECOMMENDED.

10.3. DOI Resolver Availability

DOI resolution depends on the availability of the issuing repository and the DOI federation infrastructure. Implementers SHOULD mitigate this risk by:

* Including independent archive URIs (e.g., Internet Archive snapshots) in rem_record.archive_uris.

* Self-hosting a copy of the artifact at a stable URI and including that URI in the REM Record.

10.4. Timestamp Granularity

Bitcoin block timestamps are set by miners and may vary by up to two hours from wall-clock time under network rules. Additionally, OpenTimestamps calendar servers aggregate hashes and submit Merkle roots to Bitcoin in batches, meaning the calendar attestation time may precede the Bitcoin block confirmation by minutes to hours.

Implementers SHOULD record both the calendar attestation datetime (if using OpenTimestamps) and the confirmed block date in the REM Record to preserve full temporal evidence.

10.5. Privacy of Artifact Content

The REM Protocol records a SHA-256 hash, not the artifact content itself. Publishing a REM Record does not expose the artifact's content. However, if the artifact contains sensitive information, implementers SHOULD consider whether the DOI landing page metadata or description fields may inadvertently reveal details about artifact contents that the author did not intend to disclose.

For artifacts that must remain confidential, implementers MAY perform Steps 1 and 2 (hashing and blockchain timestamping) without proceeding to Step 3 (DOI registration), thereby obtaining a blockchain-anchored proof of existence without public disclosure. Such a record is a Partial REM Record and SHOULD be clearly marked as such in any associated documentation.

11. IANA Considerations

This document has no IANA actions.

Future revisions of this specification MAY request registration of a media type (e.g., application/rem-record+json) with IANA to facilitate standardized handling of REM Record JSON manifests. Such registration would follow the procedures in [RFC6838].

12. Applications

The REM Protocol is applicable in any domain requiring durable, independently verifiable proof of the existence and content of a digital artifact at a specific point in time.

Academic Publishing:
: Authors may timestamp preprints and datasets prior to submission, establishing a verifiable priority record independent of the peer-review timeline.

Intellectual Property:
: Prior-art establishment for inventions, protocols, and standards without dependence on patent prosecution timelines or institutional intermediaries.

Regulatory Compliance:
: Finance, healthcare, and logistics operators subject to record-keeping mandates may use REM to produce tamper-evident audit trails. The EU AI Act [EU-AI-Act] and similar instruments requiring documented AI system provenance represent emerging use cases for this application domain.

AI and ML Governance:
: Training datasets, model weights, evaluation benchmarks, and alignment-assessment records may be REM-anchored to establish verifiable provenance chains, supporting the requirements of the CTS framework [draft-reilly-cts-00].

Legal and Evidentiary Contexts:
: Contracts, affidavits, and correspondence may be REM-anchored to produce an independent record of document existence and content at a specific date, suitable for evidentiary purposes.

13. Informative References

[draft-reilly-cts-00]
Reilly, L.J., "Cognitive Trust Stack (CTS): A Protocol Framework for Verifiable AI Behavioral Provenance", Internet-Draft draft-reilly-cts-00, March 2026, <https://datatracker.ietf.org/doc/draft-reilly-cts/>.

[draft-reilly-rlt-genesis]
Reilly, L.J., "REM License Token (RLT) Genesis Specification", Internet-Draft draft-reilly-rlt-genesis-00, 2025,

            <https://datatracker.ietf.org/doc/
            draft-reilly-rlt-genesis/>.

   [EU-AI-Act]
            European Parliament and of the Council, "Regulation
            (EU) 2024/1689 on Artificial Intelligence (EU AI Act)",
            2024, <https://eur-lex.europa.eu/legal-content/EN/TXT/
            ?uri=OJ:L_202401689>.

   [FIPS-180-4]
            National Institute of Standards and Technology, "Secure
            Hash Standard (SHS)", FIPS PUB 180-4, August 2015,
            <https://doi.org/10.6028/NIST.FIPS.180-4>.

   [ISO-26324]
            International DOI Foundation, "DOI Handbook
            (ISO 26324)", 2012,
            <https://www.doi.org/doi_handbook/>.

   [OpenTimestamps]
            Todd, P., "OpenTimestamps: Scalable, Trust-Minimized,
            Distributed Timestamping with Bitcoin", 2016,
            <https://opentimestamps.org/>.

   [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>.

   [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type
            Specifications and Registration Procedures", BCP 13,
            RFC 6838, DOI 10.17487/RFC6838, January 2013,
            <https://www.rfc-editor.org/info/rfc6838>.

   [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>.

   [ZenodoStandards]
            CERN/Zenodo, "Zenodo Archiving Standards", 2025,
            <https://about.zenodo.org/>.

Appendix A. Change Log

   (This appendix to be removed before or upon RFC publication.)

   Changes from -00 to -01:

   * Added Section 3 (Requirements Language) with RFC 2119 / RFC 8174
      normative language boilerplate. The -00 contained no normative
      MUST/SHOULD/MAY requirements; the specification has been updated
      throughout to use RFC 2119 key words consistently.

   * Added Section 6 (REM Record Format) defining the canonical JSON
      manifest schema (rem_version "1.1"), field semantics, and a
      complete worked example.

   * Added Section 7 (Verification Procedure) specifying independent
      hash verification, blockchain verification (both OpenTimestamps

and direct on-chain paths), and DOI resolution verification, with defined outcome states: FULLY VERIFIED, PARTIALLY VERIFIED, and INVALID.

* Added Section 8 (Implementation Considerations) covering recommended toolchain, timing and ordering constraints (including the legal significance of operation ordering), and a versioning and amendment policy requiring that original REM Records not be modified.

* Added Section 9 (Interoperability) documenting REM's role as the foundational permanence primitive for the Reilly Protocol Suite, its relationship to CTS, RLT Genesis, RSP, RRP, RBIP, and RGIP, and criteria for blockchain agnosticism.

* Expanded Section 10 (Security Considerations) with five subsections: hash algorithm agility and post-quantum readiness guidance; blockchain reorganization depth recommendations; DOI resolver availability mitigation; timestamp granularity considerations; and content privacy guidance including the Partial REM Record option for confidential artifacts.

* Corrected erroneous informative reference: RFC 9162 (Certificate Transparency Version 2.0) appeared in -00 as if it described OpenTimestamps; it does not. RFC 9162 has been removed and replaced with the correct OpenTimestamps reference [OpenTimestamps].

* Added informative references to draft-reilly-cts-00, draft-reilly-rlt-genesis, FIPS-180-4, RFC 2119, RFC 6838, RFC 8174, and EU-AI-Act.

* Removed duplicate Author's Address section present in -00.

* Updated expiry date from 5 March 2026 (lapsed) to 20 September 2026.

Author's Address

Lawrence John Reilly Jr.
Independent

Email: lawrencejohnreilly@gmail.com