Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 21 September 2026

L.J. Reilly
Independent

21 March 2026

Reilly Banking Integrity Protocol (RBIP)
draft-reilly-banking-integrity-01

Abstract

This document defines version 01 of the Reilly Banking Integrity Protocol (RBIP), a compliance-grade architecture for generating immutable, auditor- and regulator-verifiable evidence trails in banking operations. RBIP combines cryptographic anchoring (via blockchain timestamping) with archival DOI issuance to produce permanent, tamper-evident records across three core domains: Proof-of-Reserves & Liquidity (PRL), Loan Origination & Collateral Chain (LOC), and KYC/AML Evidence Ledger (KAL).

RBIP is designed to help financial institutions satisfy requirements from Basel III/IV, SOX, BSA/AML, DORA, MiCA, ISO/IEC 42001:2023, and other applicable regulations while preserving privacy, accountability, and auditability.

This document and its foundational specification are protected under triple-layer digital permanence: (1) DOI archival via a persistent archival provider (DOI: 10.5281/zenodo.17114424, published September 13, 2025, initially archived at Zenodo); (2) blockchain timestamping via a public proof-of-work blockchain timestamping service (initially anchored via OpenTimestamps to the Bitcoin blockchain); and (3) IETF Internet-Draft submission (draft-reilly-banking-integrity-00, submitted September 27, 2025). This triple-layer permanence establishes cryptographically verifiable prior art with an immutable public timestamp predating any subsequent implementation, derivative work, or regulatory mandate.

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 21 September 2026.

Copyright Notice

Table of Contents

---

1. Introduction

1.1. Motivation

Financial regulators and auditors require trustworthy, tamper-evident audit trails of banking operations. Conventional centralized logs and database snapshots are susceptible to undetectable alteration, retrospective deletion, or insider manipulation. High-profile failures -- from fraudulent reserve attestations to undocumented collateral substitution -- demonstrate that cryptographic guarantees are no longer optional; they are a structural prerequisite for systemic trust in financial

infrastructure.

Existing audit frameworks rely on periodic sampling, self-reported attestations, and trusted-third-party certifications. None of these mechanisms produce independently verifiable, continuous, tamper-evident records that auditors can validate without institutional cooperation. RBIP closes this gap by combining two orthogonal mechanisms:

- Deep archival of evidence artifacts under persistent, globally resolvable identifiers (DOIs), ensuring permanent accessibility and citation permanence independent of any single institution.

- Cryptographic anchoring of artifact digests to a public blockchain, establishing a chronologically ordered, tamper-evident ledger of events that cannot be rewritten without detectable proof-of-work invalidation.

Together, these mechanisms produce audit trails that are permanent, tamper-evident, and regulator-verifiable -- properties that no existing proprietary compliance product achieves simultaneously and at open-protocol scale.

1.2. Scope

RBIP is a protocol specification for the generation, archival, anchoring, and verification of banking integrity evidence. It does NOT prescribe any particular blockchain platform, DOI provider, or financial core system. Implementations MUST conform to the cryptographic requirements and data models specified herein but MAY choose conforming backends.

RBIP covers:

- Proof-of-Reserves and Liquidity attestations (PRL module)
- Loan origination, underwriting, and collateral chain events (LOC module)
- KYC/AML identity verification and case management (KAL module)
- Bundle formation, DOI archival, and blockchain anchoring workflows
- Regulator, auditor, and public transparency APIs
- Security, privacy, and regulatory compliance requirements

RBIP does NOT cover:

- Core banking transaction processing
- Payment clearing and settlement protocols
- Smart contract execution logic
- Specific blockchain consensus mechanisms

1.3. Triple-Layer Permanence Declaration

The RBIP specification is itself protected and timestamped under the author’s Reilly EternaMark (REM) Protocol [I-D.draft-reilly-rem-protocol], applying triple-layer digital permanence:

Layer 1 -- DOI Archival:
The foundational RBIP whitepaper was published and DOI-archived at a persistent archival provider on September 13, 2025. The assigned DOI is permanently resolvable via the Handle System and indexed by DataCite and OpenAIRE:
DOI: 10.5281/zenodo.17114424
URL: https://zenodo.org/records/17114424
NOTE: Zenodo (operated by CERN) was used as the initial archival provider for this specification. RBIP implementations MAY use any DataCite-member DOI registration agency or equivalent persistent identifier infrastructure (e.g., Zenodo, Figshare,

institutional repositories, or national data archives). The requirement is a globally resolvable, persistent identifier -- not any specific platform.

Layer 2 -- Blockchain Timestamping:
The specification artifact was submitted to a public blockchain timestamping service, producing a cryptographic proof-of-existence attestation with a block timestamp that is immutable by design of proof-of-work consensus. This timestamp PRECEDES any subsequent regulatory mandate, derivative implementation, or competing claim. NOTE: OpenTimestamps anchored to the Bitcoin blockchain was used as the initial timestamping mechanism for this specification. RBIP implementations MAY use any public, proof-of-work blockchain with sufficiently deep finality (RECOMMENDED: 6+ confirmations) and a compatible open timestamping protocol. Other suitable implementations include Ethereum-based anchoring services, Hedera Hashgraph, or any IETF-compatible distributed timestamping mechanism conforming to [RFC3161]. The requirement is a publicly verifiable, tamper-evident chronological commitment -- not any specific blockchain platform.

Layer 3 -- IETF Internet-Draft Submission:
The protocol was submitted to the IETF as an Internet-Draft on September 27, 2025 (draft-reilly-banking-integrity-00), recorded in the IETF Datatracker at:
https://datatracker.ietf.org/doc/draft-reilly-banking-integrity/
IETF submission timestamps are publicly logged, independently verifiable, and archived by the IETF document repository.

The combination of these three independent, redundant archival and timestamping mechanisms constitutes triple-layer digital permanence. Any attempt to dispute authorship, priority, or provenance of this specification must contend with three independently operated, cryptographically verifiable systems.

---

2. 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.

---

3. Terminology

This section defines terms used throughout this document.

Evidence Item:
The atomic unit of RBIP. A structured record capturing a discrete banking event, including cryptographic linkage to prior events. Analogous to a journal entry in a tamper-evident ledger.

Evidence Bundle:
A collection of Evidence Items aggregated for a defined interval (e.g., daily, hourly). Bundles are the unit of DOI archival and blockchain anchoring.

Bundle Root Digest:
A single SHA-256 (or SHA3-256) hash representing the Merkle root of all Evidence Items in a bundle. This is the value submitted to the blockchain anchoring service.

DOI (Digital Object Identifier):
A persistent, globally unique identifier assigned to a published artifact. RBIP uses DOIs to ensure long-term, institution-independent accessibility of Evidence Bundles.

Blockchain Anchor:
A transaction on a public blockchain that embeds or commits to the Bundle Root Digest, producing a tamper-evident proof-of-existence for the bundle. The transaction's block timestamp constitutes chronological proof that the bundle existed at or before that time. Any public, proof-of-work or proof-of-stake blockchain with sufficient finality and public verifiability (e.g., Bitcoin, Ethereum) MAY be used as the anchoring chain.

Triple-Layer Permanence:
The RBIP-defined property of an artifact that has been simultaneously (1) DOI-archived, (2) blockchain-timestamped, and (3) submitted to a public standards body. Each layer is independently verifiable and redundant.

PRL (Proof-of-Reserves & Liquidity):
The RBIP module capturing reserve attestations, liquidity stress test outcomes, and regulatory capital snapshots.

LOC (Loan Origination & Collateral Chain):
The RBIP module capturing the full lifecycle of loan events from application through default resolution.

KAL (KYC/AML Evidence Ledger):
The RBIP module capturing identity verification, transaction monitoring alerts, SAR filings, and AML case lifecycle events.

Canonical Form:
A deterministic, byte-for-byte reproducible serialization of an Evidence Item. RBIP mandates JCS (JSON Canonicalization Scheme) [RFC8785] for JSON serialization and CBOR Deterministic Encoding [RFC8949] for CBOR serialization.

prev_digest:
The SHA-256 digest of the immediately preceding Evidence Item in the same module chain. Creates an unforgeable chain linking all events in chronological order.

evidence_id:
A globally unique identifier for an Evidence Item, formatted as a UUID v4 [RFC9562] prefixed with the module code.
Example: "PRL-550e8400-e29b-41d4-a716-446655440000"

HSM (Hardware Security Module):
A tamper-resistant hardware device used to generate, store, and manage cryptographic keys. RBIP REQUIRES HSM-backed key operations for bundle signing.

RBIP Node:
A software component deployed within a financial institution that implements the RBIP Evidence Generation, Bundling, and Anchoring workflows as specified in this document.

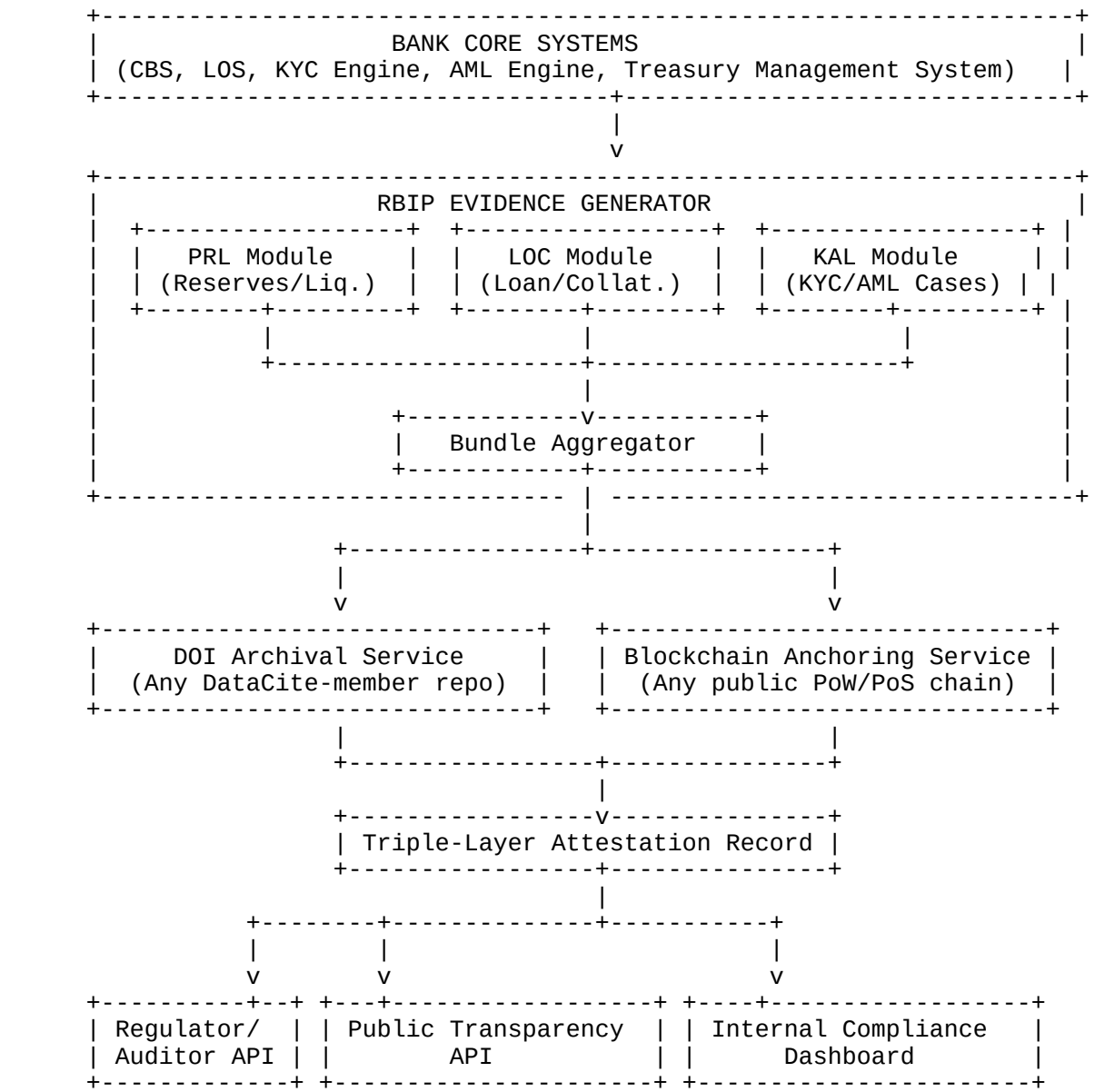
Auditor Interface:
The RBIP API endpoint exposed to authorized regulators and auditors for evidence retrieval, Merkle path verification, and anchor validation.

---

4. Design Overview

4.1. Architecture and Modules

RBIP is composed of three cooperating compliance modules and a shared anchoring infrastructure layer:



The RBIP Node integrates with the financial institution’s core systems via an event adapter layer. All events captured by the three modules flow through the Evidence Generator, are aggregated into Bundles, and simultaneously submitted to the DOI Archival and Blockchain Anchoring services.

4.2. Threat Model

RBIP is designed to resist the following adversarial scenarios:

T1 -- Retrospective Record Modification:
An internal actor or attacker with database access attempts to alter historical event records. RBIP counters this via cryptographic chaining (prev_digest) and blockchain anchoring. Any modification breaks the digest chain and invalidates the blockchain anchor.

T2 -- Record Deletion:
An actor attempts to delete records covering a specific period. RBIP counters this via DOI archival (bundles are permanently archived) and Merkle proofs (absence of a bundle produces a gap in the chain that auditors can detect).

T3 -- Event Reordering:
An actor attempts to reorder events to alter the apparent sequence of decisions. RBIP counters this via prev_digest chaining and bundle timestamp ordering anchored to the blockchain.

T4 -- Insider Collusion with Log Manager:
Operators collude to replace archived bundles. RBIP counters this via the blockchain anchor: the bundle's Merkle root is publicly anchored, so any replacement bundle would produce a different root digest, invalidating the anchor.

T5 -- Customer Data Exposure:
An attacker or unauthorized auditor accesses sensitive PII or financial data via the RBIP APIs. RBIP counters this via selective encryption, role-based access control on the Auditor API, and zero-knowledge proofs for public transparency (MAY be implemented per Section 7.3).

T6 -- Timestamp Fraud:
An actor backdates an event to claim it occurred before a regulatory deadline. RBIP counters this via blockchain anchoring: the block timestamp is determined by network consensus and cannot be retroactively altered without invalidating the proof-of-work or proof-of-stake chain from that point forward.

T7 -- API Replay Attacks:
An attacker replays a valid API request to duplicate evidence submissions. RBIP counters this via idempotency keys, nonce requirements, and TLS with mutual authentication.

4.3. Trust Hierarchy

RBIP defines the following trust hierarchy:

Level 1 -- RBIP Node (Institution-Controlled):
Generates Evidence Items. Trusted for event capture and serialization. MUST be deployed in an HSM-backed environment.

Level 2 -- DOI Archival Provider:
Archives Evidence Bundles. Trusted for long-term preservation. SHOULD be a DataCite-member repository or equivalent persistent identifier infrastructure that guarantees long-term, institution-independent accessibility.

Level 3 -- Blockchain Anchoring Service:
Embeds the Bundle Root Digest on-chain. Trusted for chronological ordering. MUST use a public blockchain with sufficient finality and publicly verifiable transaction history. At least 6 block confirmations are REQUIRED before treating an anchor as final (see Section 9.2).

Level 4 -- Independent Auditors and Regulators:
External verifiers. Trust nothing except the cryptographic proofs. RBIP enables trustless verification: auditors MUST be able to verify all claims without institutional cooperation.

---

5. Data Model and Canonical Serialization

5.1. Evidence Item Structure

Each Evidence Item MUST be serialized in canonical JSON form per [RFC8785] (JCS -- JSON Canonicalization Scheme) before hashing. The canonical form MUST NOT include insignificant whitespace. Implementations supporting CBOR MUST use CBOR Deterministic Encoding [RFC8949] Section 4.2.

Evidence Item JSON Schema:

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://rbip.blackseedlabs.io/schema/evidence-item-v1",
  "type": "object",
  "required": [
    "evidence_id", "schema_version", "module", "event_type",
    "institution_id", "timestamp", "prev_digest", "data_digest",
    "data_fields", "signature"
  ],
  "properties": {
    "evidence_id": {
      "type": "string",
      "pattern": "^(PRL|LOC|KAL)-[0-9a-f]{8}-[0-9a-f]{4}-4[0-9a-f]{3}-[89ab][0-9a-f]{3
}-[0-9a-f]{12}$",
      "description": "Module-prefixed UUID v4 [RFC9562] uniquely identifying this Evid
ence Item."
    },
    "schema_version": {
      "type": "string",
      "const": "rbip-01",
      "description": "RBIP schema version.  MUST be 'rbip-01' for this specification."
    },
    "module": {
      "type": "string",
      "enum": ["PRL", "LOC", "KAL"],
      "description": "The RBIP module that generated this item."
    },
    "event_type": {
      "type": "string",
      "description": "One of the event type strings defined in Sections 5.4-5.6."
    },
    "institution_id": {
      "type": "string",
      "description": "LEI [ISO 17442] or BIC [ISO 9362] of the reporting institution."
    },
    "timestamp": {
      "type": "string",
      "format": "date-time",
      "description": "ISO 8601 [RFC3339] UTC timestamp of event occurrence. MUST be UT
C."
    },
    "prev_digest": {
      "type": "string",
      "pattern": "^sha256:[0-9a-f]{64}$",
      "description": "SHA-256 hex digest of the preceding Evidence Item in this module
 chain. 'sha256:0000...0000' (64 zeros) for the genesis item."
    },
    "data_digest": {
      "type": "string",
      "pattern": "^sha256:[0-9a-f]{64}$",
      "description": "SHA-256 hex digest of the canonical serialization of data_fields
."
    },
    "data_fields": {
```

```
        "type": "object",
        "description": "Module-specific event payload.  Content varies by event_type (se
e Sections 5.4-5.6).  MAY be encrypted per Section 10."
      },
      "metadata": {
        "type": "object",
        "description": "Optional key-value metadata for indexing, tagging, and audit ann
otation.",
        "properties": {
          "regulatory_ref": { "type": "string" },
          "jurisdiction":   { "type": "string" },
          "tags":           { "type": "array", "items": { "type": "string" } },
          "rbip_node_id":   { "type": "string" },
          "operator_id":    { "type": "string" }
        }
      },
      "signature": {
        "type": "string",
        "description": "Base64url-encoded JWS Compact Serialization [RFC7515] signing th
e canonical Evidence Item (excluding the signature field itself) using the institution's
RBIP signing key."
      }
    }
  }
```

The data_digest field MUST be computed over the canonical serialization of data_fields only (not the full Evidence Item). The signature field MUST be computed over the full canonical Evidence Item with signature set to the empty string "", ensuring the signed payload is stable and reproducible.

5.2. Evidence Bundle

Evidence Items are aggregated into Evidence Bundles for DOI archival and blockchain anchoring. Bundles MAY be formed on a time-based schedule (e.g., every 15 minutes, hourly) or on an event-count threshold (e.g., every 1000 Evidence Items). Implementations MUST form and anchor at least one bundle per 24-hour UTC day per active module.

Evidence Bundle JSON Schema:

```
{
  "bundle_id":        "<UUID v4>",
  "schema_version":   "rbip-01",
  "institution_id":   "<LEI or BIC>",
  "modules_included": ["PRL", "LOC", "KAL"],
  "interval_start":   "<ISO 8601 UTC datetime>",
  "interval_end":     "<ISO 8601 UTC datetime>",
  "item_count":       <integer>,
  "merkle_root":      "sha256:<64 hex chars>",
  "merkle_tree":      [
    { "index": 0, "evidence_id": "<id>", "digest": "sha256:<...>" },
    ...
  ],
  "prev_bundle_digest": "sha256:<64 hex chars>",
  "doi":              "<DOI URI or null if not yet archived>",
  "blockchain_anchor": {
    "chain":          "<chain identifier, e.g., bitcoin, ethereum>",
    "tx_id":          "<transaction hash>",
    "block_height":   <integer>,
    "block_timestamp": "<ISO 8601 UTC datetime>",
    "proof":          "<Base64-encoded timestamping proof blob>"
  },
  "bundle_digest":    "sha256:<64 hex chars>",
  "bundle_signature": "<JWS Compact Serialization over bundle_digest>"
```

```
}
```

The merkle_root field MUST be computed as the root of a binary Merkle tree [RFC9162] constructed from the ordered list of data_digest values of all Evidence Items in the bundle. Odd-count trees MUST duplicate the last leaf per the convention in [RFC9162] Section 2.1.

The bundle_digest field MUST be computed over the canonical serialization of the entire bundle object with bundle_digest and bundle_signature set to empty strings "".

5.3. Merkle Anchoring Structure

RBIP employs a binary Merkle tree for bundle integrity. The Merkle tree enables auditors to verify a single Evidence Item's inclusion in a bundle without downloading the full bundle, which is important for privacy-preserving partial disclosure.

Merkle Proof Object (returned by Auditor API for single-item verification):

```
{
  "evidence_id":   "<id>",
  "bundle_id":     "<id>",
  "leaf_digest":   "sha256:<...>",
  "proof_path": [
    { "direction": "left",  "digest": "sha256:<...>" },
    { "direction": "right", "digest": "sha256:<...>" },
    ...
  ],
  "merkle_root":   "sha256:<...>",
  "doi":           "10.5281/zenodo.<record>",
  "blockchain_anchor": { ... }
}
```

Auditors MUST be able to reconstruct the Merkle root from the leaf_digest and proof_path, then compare it against both the DOI-archived bundle and the blockchain anchor, without any involvement from the reporting institution.

5.4. PRL Module Events

The PRL module captures reserve, liquidity, and capital adequacy events. All monetary amounts MUST be expressed as strings in the format "<amount> <ISO 4217 currency code>" (e.g., "1500000000.00 USD") to avoid floating-point precision errors.

5.4.1. ReserveSnapshot

Triggered at a regular cadence (RECOMMENDED: daily at close-of-business) or on demand by a regulator request.

data_fields:

```
{
  "snapshot_type":       "daily_close" | "regulator_requested" | "stress_triggered",
  "reporting_date":      "<ISO 8601 date>",
  "reserve_assets": [
    {
      "asset_class":     "cash" | "hqla_l1" | "hqla_l2a" | "hqla_l2b" | "other",
      "asset_id":        "<internal asset reference or ISIN>",
      "market_value":    "<amount> <CCY>",
      "haircut_pct":     "<decimal 0.00-100.00>",
      "post_haircut_value": "<amount> <CCY>"
    }
  ],
```

```
    "total_hqla":          "<amount> <CCY>",
    "lcr_ratio":           "<decimal>",
    "nsfr_ratio":          "<decimal>",
    "tier1_capital_ratio": "<decimal>",
    "attestation_officer": "<name and LEI or employee ID>",
    "attestation_method":  "automated_system" | "manual_review" | "external_audit"
  }
```

5.4.2. LiquidityStressTest

```
  data_fields:
  {
    "stress_scenario":     "regulatory_standard" | "adverse" | "severely_adverse" | "ins
titution_defined",
    "scenario_id":         "<identifier>",
    "horizon_days":        <integer>,
    "baseline_lcr":        "<decimal>",
    "stressed_lcr":        "<decimal>",
    "survival_days":       <integer>,
    "assumptions": [
      { "parameter": "<string>", "value": "<string>", "basis": "<string>" }
    ],
    "model_version":       "<semver string>",
    "run_timestamp":       "<ISO 8601 UTC datetime>"
  }
```

5.4.3. ReserveAttestation

A signed certification by an authorized officer or external auditor that the ReserveSnapshot accurately reflects the institution's assets.

```
  data_fields:
  {
    "attests_to_snapshot_id": "<evidence_id of the ReserveSnapshot>",
    "attestor_name":          "<full legal name>",
    "attestor_role":          "CFO" | "CRO" | "external_auditor" | "regulator",
    "attestor_institution":   "<LEI>",
    "attestation_scope":      "<free text, e.g., Basel III LCR compliance, Q4 2025>",
    "pgp_signature":          "<armored PGP signature over the attested snapshot digest>
",
    "certificate_ref":        "<X.509 certificate serial number or thumbprint>"
  }
```

5.5. LOC Module Events

The LOC module tracks the full lifecycle of loan origination, underwriting, and collateral management.

5.5.1. LoanApplication

```
  data_fields:
  {
    "application_id":     "<internal loan application ID>",
    "application_date":   "<ISO 8601 date>",
    "loan_type":          "residential_mortgage" | "commercial_real_estate" | "auto" | "
personal" | "sme" | "syndicated" | "other",
    "requested_amount":   "<amount> <CCY>",
    "requested_term_months": <integer>,
    "applicant_hash":     "sha256:<hash of anonymized applicant identifier>",
    "origination_channel":"branch" | "digital" | "broker" | "wholesale",
    "jurisdiction":       "<ISO 3166-1 alpha-2 country code>"
  }
```

NOTE: applicant_hash MUST be the SHA-256 of a stable internal applicant identifier. The identifier itself MUST NOT appear in

any RBIP evidence record to comply with privacy requirements (Section 10).

5.5.2. UnderwritingDecision

```
data_fields:
{
  "application_id":     "<ref to LoanApplication>",
  "decision":           "approved" | "declined" | "referred" | "conditionally_approved",
  "decision_date":      "<ISO 8601 date>",
  "decision_rationale_hash": "sha256:<hash of rationale document>",
  "approved_amount":    "<amount> <CCY>",
  "approved_rate":      "<decimal as percentage>",
  "approved_term_months": <integer>,
  "conditions": [
    { "condition_id": "<id>", "description_hash": "sha256:<...>", "deadline": "<date>" }
  ],
  "decision_model_id":  "<model version reference>",
  "underwriter_hash":   "sha256:<hash of underwriter ID>"
}
```

5.5.3. CollateralBinding

```
data_fields:
{
  "loan_id":            "<approved loan ID>",
  "collateral_id":      "<unique collateral item ID>",
  "collateral_type":    "residential_property" | "commercial_property" | "securities" | "vehicle" | "equipment" | "cash_deposit" | "guarantee" | "other",
  "collateral_ref":     "<ISIN, title number, VIN, or other external reference>",
  "valuation":          "<amount> <CCY>",
  "valuation_method":   "automated_valuation" | "desktop_appraisal" | "full_appraisal" | "market_price",
  "valuation_date":     "<ISO 8601 date>",
  "ltv_ratio":          "<decimal>",
  "lien_position":      "first" | "second" | "subordinate",
  "binding_date":       "<ISO 8601 date>",
  "perfection_ref":     "<UCC filing number or land registry reference>"
}
```

5.5.4. CollateralRevaluation

```
data_fields:
{
  "collateral_id":      "<ref to CollateralBinding>",
  "prior_valuation":    "<amount> <CCY>",
  "new_valuation":      "<amount> <CCY>",
  "revaluation_date":   "<ISO 8601 date>",
  "trigger":            "scheduled_review" | "market_event" | "borrower_request" | "regulator_order" | "impairment_indicator",
  "revaluation_method": "automated_valuation" | "desktop_appraisal" | "full_appraisal" | "market_price",
  "updated_ltv_ratio":  "<decimal>",
  "margin_call_triggered": <boolean>
}
```

5.5.5. LoanDisbursement

```
data_fields:
{
  "loan_id":            "<id>",
  "disbursement_id":    "<unique disbursement ID>",
  "disbursement_date":  "<ISO 8601 UTC datetime>",
  "amount":             "<amount> <CCY>",
```

```
    "disbursement_method":"wire_transfer" | "ach" | "check" | "internal_credit",
    "destination_hash":   "sha256:<hash of destination account reference>",
    "conditions_satisfied": [{ "condition_id": "<id>", "satisfied_date": "<date>" }]
  }
```

5.5.6. PaymentReceipt

```
  data_fields:
  {
    "loan_id":            "<id>",
    "payment_id":         "<unique payment ID>",
    "payment_date":       "<ISO 8601 UTC datetime>",
    "amount":             "<amount> <CCY>",
    "principal_component":"<amount> <CCY>",
    "interest_component": "<amount> <CCY>",
    "fees_component":     "<amount> <CCY>",
    "days_past_due":      <integer, 0 if current>,
    "payment_channel":    "ach" | "wire" | "internal" | "cash" | "check"
  }
```

5.5.7. DefaultEvent / RecoveryEvent

```
  data_fields:
  {
    "loan_id":            "<id>",
    "event_subtype":      "default_declaration" | "recovery_partial" | "recovery_full" |
 "charge_off" | "workout_agreement",
    "event_date":         "<ISO 8601 date>",
    "days_past_due_at_default": <integer>,
    "outstanding_balance":"<amount> <CCY>",
    "recovery_amount":    "<amount> <CCY>",
    "lgd_estimate":       "<decimal, loss given default, 0.00-1.00>",
    "resolution_timeline_days": <integer>
  }
```

5.6. KAL Module Events

The KAL module captures identity lifecycle and AML case events. All personally identifiable information MUST be encrypted or hashed before inclusion in RBIP Evidence Items (see Section 10).

5.6.1. IdentitySubmission

```
  data_fields:
  {
    "submission_id":      "<unique ID>",
    "submission_date":    "<ISO 8601 UTC datetime>",
    "customer_type":      "individual" | "entity" | "correspondent_bank" | "pep",
    "identity_doc_type":  "passport" | "national_id" | "drivers_license" | "entity_regis
tration" | "other",
    "doc_hash":           "sha256:<hash of identity document image or data>",
    "liveness_check_result": "pass" | "fail" | "not_required",
    "submission_channel": "branch" | "digital_onboarding" | "api_partner"
  }
```

5.6.2. IdentityVerificationResult

```
  data_fields:
  {
    "submission_id":      "<ref to IdentitySubmission>",
    "verification_date":  "<ISO 8601 UTC datetime>",
    "result":             "verified" | "failed" | "referred" | "escalated",
    "verification_method":"automated_ocr_biometric" | "manual_review" | "third_party_bur
eau" | "video_interview",
    "risk_score":         <integer 0-1000>,
    "risk_tier":          "low" | "medium" | "high" | "prohibited",
```

```
    "sanctions_checked":  <boolean>,
    "pep_checked":        <boolean>,
    "adverse_media_checked": <boolean>,
    "next_review_date":   "<ISO 8601 date>"
  }
```

5.6.3. TransactionMonitorAlert

```
  data_fields:
  {
    "alert_id":           "<unique alert ID>",
    "alert_generated_at": "<ISO 8601 UTC datetime>",
    "alert_type":         "structuring" | "layering" | "unusual_velocity" | "sanctions_m
atch" | "pep_activity" | "high_risk_jurisdiction" | "other",
    "rule_id":            "<AML rule or model ID that triggered>",
    "customer_hash":      "sha256:<hash of customer ID>",
    "transaction_count":  <integer>,
    "aggregate_amount":   "<amount> <CCY>",
    "risk_score":         <integer 0-1000>,
    "auto_disposition":   "escalated_to_analyst" | "auto_cleared" | "pending"
  }
```

5.6.4. SARSubmission

```
  data_fields:
  {
    "sar_reference_hash": "sha256:<hash of SAR filing reference number>",
    "submission_date":    "<ISO 8601 UTC datetime>",
    "jurisdiction":       "<ISO 3166-1 alpha-2>",
    "fiu_agency":         "FinCEN" | "NCA" | "FINTRAC" | "AUSTRAC" | "other",
    "activity_type":      "money_laundering" | "terrorist_financing" | "fraud" | "struct
uring" | "other",
    "subject_count":      <integer>,
    "reporting_period_start": "<ISO 8601 date>",
    "reporting_period_end":   "<ISO 8601 date>",
    "filing_status":      "initial" | "corrective" | "continuing"
  }
```

5.6.5. CaseEscalation

```
  data_fields:
  {
    "case_id":            "<unique case ID>",
    "escalation_date":    "<ISO 8601 UTC datetime>",
    "from_tier":          "l1_analyst" | "l2_investigator" | "l3_senior",
    "to_tier":            "l2_investigator" | "l3_senior" | "compliance_officer" | "law_
enforcement",
    "escalation_reason":  "<free text, hashed if sensitive>",
    "linked_alert_ids":   ["<alert_id>", ...],
    "linked_sar_hashes":  ["sha256:<...>", ...]
  }
```

5.6.6. CaseClosure

```
  data_fields:
  {
    "case_id":            "<ref to CaseEscalation>",
    "closure_date":       "<ISO 8601 UTC datetime>",
    "closure_outcome":    "no_action" | "sar_filed" | "account_closed" | "law_enforcemen
t_referral" | "regulatory_disclosure",
    "investigation_duration_days": <integer>,
    "disposition_hash":   "sha256:<hash of case disposition document>"
  }
```

---

6. Protocol Workflow

6.1. Evidence Generation

The following steps MUST be executed for every banking event captured by RBIP:

Step 1 -- Event Capture:
The RBIP Node receives an event notification from the core banking system via the Internal Bank Systems Interface (Section 7.1).

Step 2 -- Field Population:
The RBIP Node populates the Evidence Item structure per Section 5.1. Sensitive fields MUST be encrypted or hashed per Section 10 before populating data_fields.

Step 3 -- Prev-Digest Linking:
The RBIP Node retrieves the data_digest of the most recently committed Evidence Item in the same module chain and sets prev_digest to that value. For the genesis item, prev_digest MUST be set to "sha256:" followed by 64 zeros.

Step 4 -- Data Digest Computation:
Serialize data_fields to canonical JSON per [RFC8785]. Compute SHA-256 of the canonical bytes. Set data_digest.

Step 5 -- Full Item Serialization:
Serialize the full Evidence Item (with signature set to "") to canonical JSON per [RFC8785].

Step 6 -- Signing:
Using the institution’s RBIP signing key (HSM-backed, see Section 8.3), compute a JWS Compact Serialization [RFC7515] signature over the canonical Evidence Item bytes. Set the signature field.

Step 7 -- Emission:
Emit the signed Evidence Item to the Bundle Aggregator. The item MUST be persisted atomically before the event notification is acknowledged.

6.2. Bundle Formation

Step 1 -- Collection:
The Bundle Aggregator collects Evidence Items per the configured bundling policy (time-based or count-based).

Step 2 -- Ordering:
Items are sorted ascending by timestamp, then by evidence_id lexicographically to resolve same-timestamp ties.

Step 3 -- Merkle Tree Construction:
Construct a binary Merkle tree over the ordered list of data_digest values. Compute the merkle_root.

Step 4 -- Bundle Digest:
Serialize the full Bundle object (with bundle_digest and bundle_signature set to "") to canonical JSON per [RFC8785]. Compute SHA-256 to obtain bundle_digest.

Step 5 -- Bundle Signing:
Sign bundle_digest using the institution’s RBIP bundle signing key (MAY be the same as the item signing key). Set bundle_signature.

6.3. DOI Archival Step

RBIP does not mandate any specific DOI archival provider. Conforming implementations MAY use any DataCite-member DOI registration agency or equivalent persistent identifier infrastructure (e.g., institutional repositories, national data archives, or certified research data repositories). The requirement is a globally resolvable, persistent identifier that guarantees long-term accessibility independent of any single institution.

Step 1 -- Submission:
The RBIP Node submits the complete Evidence Bundle (excluding blockchain_anchor fields, which are populated post-anchoring) to the configured DOI archival provider via its published deposit API.

Step 2 -- Metadata:
The submission MUST include metadata conforming to the DataCite Metadata Schema 4.5 [DataCite4.5]:

- title: "RBIP Evidence Bundle - <institution_id> - <interval_start> to <interval_end>"
- creator: <institution legal name and LEI>
- publicationYear: <YYYY>
- resourceType: "Dataset"
- description: "RBIP-01 evidence bundle: <modules> events, merkle_root: <value>"
- identifier: <DOI assigned by provider>
- relatedIdentifier: <prior bundle DOI>

Step 3 -- DOI Recording:
Upon receiving the DOI from the archival provider, the RBIP Node updates the bundle record’s doi field.

Step 4 -- DOI Verification (RECOMMENDED):
The RBIP Node SHOULD re-resolve the DOI via the Handle System (https://doi.org/<doi>) within 30 minutes to confirm successful archival before triggering blockchain anchoring.

6.4. Blockchain Anchoring and Timestamping

RBIP does not mandate any specific blockchain platform or timestamping service. Conforming implementations MAY use any public, proof-of-work or proof-of-stake blockchain with sufficient finality guarantees, provided the anchoring produces a publicly verifiable, tamper-evident chronological commitment. Suitable implementations include, but are not limited to, open timestamping protocols compatible with public blockchains and distributed timestamping mechanisms conforming to [RFC3161].

Step 1 -- Anchor Data Preparation:
Compute the Anchor Payload as:

```
anchor_payload = sha256(bundle_digest || doi_uri_bytes)
```

where || denotes concatenation. If doi is null (DOI archival failed), anchor_payload = bundle_digest.

Step 2 -- Submission to Anchoring Service:
Submit anchor_payload to the configured blockchain anchoring service. The service MUST return:

a. A transaction identifier (tx_id) on the target chain.
b. A proof artifact (e.g., a Merkle inclusion proof or timestamping proof blob) sufficient for independent offline verification.
c. The block height and block timestamp upon confirmation.

Step 3 -- Confirmation Waiting:

RBIP REQUIRES at least 6 block confirmations on the target chain before treating the anchor as final. 10 confirmations are RECOMMENDED for high-value bundles (e.g., daily PRL snapshots). The required number of confirmations MAY be increased by institutional policy based on the finality characteristics of the chosen chain.

Step 4 -- Anchor Recording:
Record in blockchain_anchor:
* chain: <chain identifier string, e.g., "bitcoin", "ethereum">
* tx_id: <transaction hash>
* block_height: <block number>
* block_timestamp: <block header timestamp, ISO 8601 UTC>
* proof: <Base64-encoded timestamping proof blob>

Step 5 -- Bundle Update and Re-sign:
Update the bundle with the populated blockchain_anchor fields. Recompute bundle_digest and bundle_signature over the updated canonical bundle. Archive the final bundle to the DOI provider as a new version.

6.5. Triple-Layer Permanence Attestation

After Steps 6.3 and 6.4 are both complete, the RBIP Node MUST generate a Triple-Layer Permanence Attestation (TLPA) record for the bundle:

```
{
  "tlpa_version": "rbip-01",
  "bundle_id": "<id>",
  "bundle_digest": "sha256:<...>",
  "layer_1_doi": {
    "doi": "<DOI assigned by archival provider>",
    "doi_url": "https://doi.org/<doi>",
    "archived_at": "<ISO 8601 UTC datetime>",
    "provider": "<name of DOI archival provider>"
  },
  "layer_2_blockchain": {
    "chain": "<chain identifier, e.g., bitcoin, ethereum>",
    "tx_id": "<tx hash>",
    "block_height": <integer>,
    "block_timestamp": "<ISO 8601 UTC datetime>",
    "proof_sha256": "sha256:<hash of timestamping proof blob>",
    "confirmations_at_finality": <integer>
  },
  "layer_3_ietf": {
    "draft_name": "draft-reilly-banking-integrity-01",
    "ietf_datatracker_url": "https://datatracker.ietf.org/doc/draft-reilly-banking-int
egrity/",
    "submission_date": "2025-09-27",
    "this_version_date": "2026-03-21"
  },
  "tlpa_digest": "sha256:<hash of canonical TLPA object with tlpa_digest=’’>",
  "tlpa_signature": "<JWS over tlpa_digest>"
}
```

The TLPA MUST be stored alongside the Evidence Bundle and MUST be returned to auditors upon request (Section 7.2).

6.6. Evidence Verification and Audit

An independent auditor wishing to verify an Evidence Item MUST execute the following steps without institutional cooperation:

Step 1 -- Retrieve TLPA:
Obtain the TLPA for the relevant bundle from the Auditor API.

Step 2 -- Resolve DOI:
Resolve the DOI at https://doi.org/<doi>. Download the archived bundle. Verify bundle_digest matches the TLPA.

Step 3 -- Verify Blockchain Anchor:
Verify the timestamping proof artifact against the target blockchain using an appropriate verifier for the configured chain and protocol. Confirm the bundle_digest (or anchor_payload) is committed to the proof. Confirm the block_timestamp matches the TLPA.

Step 4 -- Verify Merkle Proof:
For each Evidence Item under investigation, request a Merkle proof from the Auditor API (Section 7.2). Recompute the Merkle root from the proof path and leaf digest. Confirm it matches the bundle's merkle_root.

Step 5 -- Verify Digest Chain:
For sequential Evidence Items in a module, verify that each item's prev_digest equals the SHA-256 of the preceding item's canonical serialization. Gaps or mismatches indicate tampering.

Step 6 -- Verify Signatures:
Verify the JWS signature on each Evidence Item using the institution's published RBIP signing key certificate. Verify the bundle_signature using the institution's bundle signing key certificate.

An Evidence Item is VERIFIED if and only if all six steps succeed. Failure of any step MUST be reported to the institution's regulator as a potential integrity incident.

---

7. Interfaces and APIs

All RBIP API endpoints MUST be served over TLS 1.3 [RFC8446]. All API requests MUST include a JWS-signed JWT [RFC7519] Bearer token in the Authorization header. Mutual TLS (mTLS) [RFC8705] is REQUIRED for the Regulator/Auditor API (Section 7.2).

7.1. Internal Bank Systems Interface

The Internal Bank Systems Interface is a REST API exposed by the RBIP Node to the institution's core banking systems for event submission.

Base URL: https://<rbip-node.institution.internal>/v1/

7.1.1. Submit Evidence Event

POST /events

Request Body:

```
{
  "module":      "PRL" | "LOC" | "KAL",
  "event_type":  "<event type string>",
  "occurred_at": "<ISO 8601 UTC datetime>",
  "data_fields": { ... },
  "metadata":    { ... }
}
```

Response 202 Accepted:

```
{
  "evidence_id": "<module-prefixed UUID>",
```

```
  "accepted_at": "<ISO 8601 UTC datetime>",
  "bundle_queue_position": <integer>
}

Response 400 Bad Request:
{
  "error": "INVALID_EVENT_TYPE" | "MISSING_REQUIRED_FIELD" | "SCHEMA_VIOLATION",
  "detail": "<human-readable description>"
}
```

The endpoint MUST be idempotent with respect to a client-supplied idempotency-key header (format: UUID v4). Duplicate submissions with the same idempotency key MUST return 200 OK with the original evidence_id.

7.1.2. Query Evidence Item

```
GET /events/{evidence_id}

Response 200 OK: Full signed Evidence Item JSON
Response 404 Not Found: Item does not exist
```

7.1.3. Query Module Chain

```
GET /modules/{module}/chain?from_id=<evidence_id>&limit=<int>
```

Returns an ordered array of Evidence Items from from_id forward. Useful for continuous integrity monitoring by internal compliance.

7.1.4. Bundle Status

```
GET /bundles/{bundle_id}

Response 200 OK: Full Evidence Bundle JSON (including TLPA)
Response 404 Not Found
```

7.1.5. List Bundles

```
GET /bundles?module=<module>&start=<datetime>&end=<datetime>&page=<int>
```

Returns paginated list of bundle summaries with doi and anchoring status.

7.2. Regulator / Auditor API

The Regulator/Auditor API is exposed to authorized external parties. All connections MUST use mTLS [RFC8705]. Client certificates MUST be pre-registered with the RBIP Node and bound to an authorized auditor or regulatory identity. All access MUST be logged in a separate tamper-evident access log that is itself RBIP-evidenced.

Base URL: https://<rbip-node.institution.external>/audit/v1/

7.2.1. Get Evidence Bundle

```
GET /bundles/{bundle_id}
```

Returns the full Evidence Bundle including TLPA.
Encrypted data_fields are returned in their encrypted form.
The auditor MAY request selective decryption via a separate Selective Disclosure request (Section 7.2.5).

7.2.2. Get Merkle Proof

```
GET /bundles/{bundle_id}/proof/{evidence_id}
```

```
Returns:
{
  "evidence_id":   "<id>",
  "bundle_id":     "<id>",
  "leaf_digest":   "sha256:<...>",
  "proof_path": [ { "direction": "left"|"right", "digest": "sha256:<...>" }, ... ],
  "merkle_root":   "sha256:<...>",
  "doi":           "<DOI URI>",
  "blockchain_anchor": { ... },
  "tlpa":          { ... }
}
```

7.2.3. Verify Chain Integrity

```
POST /verify/chain

Request Body:
{
  "module":    "PRL" | "LOC" | "KAL",
  "from_date": "<ISO 8601 date>",
  "to_date":   "<ISO 8601 date>"
}

Response:
{
  "status":       "VERIFIED" | "CHAIN_GAP_DETECTED" | "DIGEST_MISMATCH" | "ANCHOR_INVALID",
  "bundle_count": <integer>,
  "item_count":   <integer>,
  "first_bundle_id": "<id>",
  "last_bundle_id":  "<id>",
  "anomalies": [ { "type": "<string>", "bundle_id": "<id>", "detail": "<string>" } ]
}
```

7.2.4. Verify Blockchain Anchor

```
POST /verify/anchor

Request Body:
{
  "bundle_id": "<id>"
}

Response:
{
  "bundle_id":          "<id>",
  "anchor_valid":       <boolean>,
  "chain":              "<chain identifier>",
  "tx_id":              "<hash>",
  "block_height":       <integer>,
  "block_timestamp":    "<ISO 8601 UTC datetime>",
  "confirmations":      <integer>,
  "proof_verification": "VERIFIED" | "PENDING" | "FAILED",
  "doi_resolvable":     <boolean>
}
```

7.2.5. Selective Disclosure Request

```
POST /disclose

Request Body:
{
  "bundle_id":   "<id>",
  "evidence_ids": ["<id>", ...],
  "fields":      ["<field_name>", ...],
  "purpose":     "<regulatory authority and regulatory basis for disclosure>"
```

```
}
```

The RBIP Node verifies the requesting auditor's certificate against the pre-registered authorization list and, if authorized, decrypts and returns only the specifically requested fields. This request MUST itself be RBIP-evidenced as a KAL CaseEscalation event.

7.2.6. Audit Access Log

```
GET /access-log?from=<datetime>&to=<datetime>
```

Returns the tamper-evident log of all Auditor API access events for the specified period. The access log MUST itself be bundled and anchored per RBIP.

7.3. Public Transparency API

Financial institutions MAY expose a Public Transparency API to provide limited, privacy-preserving assurance to the public that reserves and compliance operations are actively evidenced.

This API MUST NOT expose any customer PII, sensitive financial data, or material non-public information. Implementations MUST conduct a legal review before enabling this interface.

Base URL: https://<institution.example.com>/rbip/public/v1/

7.3.1. Get Anchor Summary

```
GET /anchors?module=<module>&period=<YYYY-MM>
```

Returns:

```
{
  "institution_id": "<LEI or BIC>",
  "module":         "PRL" | "LOC" | "KAL",
  "period":         "<YYYY-MM>",
  "bundle_count":   <integer>,
  "latest_anchor": {
    "bundle_id":        "<id>",
    "block_height":     <integer>,
    "block_timestamp":  "<ISO 8601 UTC datetime>",
    "doi":              "<DOI URI>",
    "merkle_root":      "sha256:<...>"
  }
}
```

7.3.2. Verify Public Anchor

```
GET /verify/{bundle_id}
```

Returns a minimal verification response confirming whether the specified bundle is anchored and DOI-archived. Does NOT return any evidence item content.

7.4. Webhook and Notification Interface

RBIP Nodes SHOULD implement a webhook interface to notify registered subscribers (e.g., regulatory systems, compliance dashboards) of key events:

Events triggering webhooks:

- bundle_anchored -- bundle successfully blockchain-anchored
- bundle_doi_issued -- bundle DOI successfully issued
- tlpa_complete -- triple-layer permanence attestation complete
- chain_gap_detected -- digest chain gap detected (CRITICAL)
- anchor_failed -- blockchain anchoring failed (CRITICAL)

- doi_archival_failed -- DOI archival failed (CRITICAL)

Webhook payloads MUST be signed with a JWS [RFC7515] using the RBIP Node's webhook signing key. Subscribers MUST verify the JWS before processing.

---

8. Cryptographic Requirements

8.1. Digest Algorithms

RBIP REQUIRES SHA-256 [FIPS-180-4] as the primary digest algorithm for all data_digest, prev_digest, bundle_digest, and merkle_root computations.

SHA3-256 [FIPS-202] MAY be used as an alternative where required by institutional policy. If SHA3-256 is used, the digest strings MUST be prefixed with "sha3-256:" rather than "sha256:".

MD5 and SHA-1 MUST NOT be used.

8.2. Signing and Key Management

RBIP signing MUST use one of the following algorithms as specified in JSON Web Algorithms [RFC7518]:

- RS256 (RSASSA-PKCS1-v1_5 with SHA-256) -- minimum RSA key size 2048 bits; 4096 bits RECOMMENDED
- ES256 (ECDSA with P-256 and SHA-256) -- RECOMMENDED for new implementations due to smaller signature size
- ES384 (ECDSA with P-384 and SHA-384) -- REQUIRED for high-assurance PRL module signatures
- Ed25519 (EdDSA with Curve25519) -- MAY be used where supported

All signing keys MUST have:

- A defined validity period not exceeding 2 years
- A documented key rotation procedure
- A published X.509 certificate [RFC5280] or JWK Set [RFC7517] accessible to auditors for signature verification

8.3. HSM Requirements

All RBIP signing key operations MUST be performed within a FIPS 140-2 Level 3 or higher HSM. The private key material MUST NEVER exist in plaintext outside the HSM boundary.

HSM requirements:

- Key generation MUST occur within the HSM
- Key backup MUST use HSM-to-HSM key wrapping (key export in plaintext is PROHIBITED)
- All signing operations MUST be logged by the HSM
- HSM audit logs MUST be independently archived per RBIP KAL

---

9. Security Considerations

9.1. Digest Chain Integrity

The prev_digest chain is the primary mechanism preventing undetected insertion, deletion, or reordering of Evidence Items. Implementations MUST maintain an unbroken chain within each module. Any detected gap MUST trigger an immediate alert and MUST be reported to the institution's Chief Compliance Officer and to the relevant regulator within 24 hours.

9.2. Blockchain Finality

RBIP REQUIRES at least 6 block confirmations on the configured blockchain before treating an anchor as final. The precise number of confirmations required SHOULD be calibrated to the finality characteristics of the chosen chain (e.g., longer reorganization windows on chains with lower hash rate require more confirmations). Fewer confirmations risk chain reorganization that could invalidate the anchor. During the confirmation waiting period, bundles MUST be treated as pending and MUST NOT be represented to auditors as final.

9.3. Key Compromise Response

If an RBIP signing key is compromised:
(1) The institution MUST immediately revoke the key certificate.
(2) The institution MUST publish a Key Compromise Event in the RBIP audit trail, anchored under the new key.
(3) All Evidence Items signed with the compromised key MUST be cross-certified by a trusted third-party auditor.
(4) The regulator MUST be notified within 72 hours.

9.4. HSM Failure Response

If the HSM fails during evidence generation, the RBIP Node MUST queue events in an encrypted staging area and halt public API availability until the HSM is restored. Evidence Items generated during an HSM outage MUST be retroactively cross-signed by a backup HSM within 4 hours.

9.5. Denial of Service

The Internal and Auditor APIs MUST implement rate limiting. The Auditor API MUST implement adaptive rate limiting that tightens limits when anomalous query patterns are detected.

9.6. Cryptographic Agility

RBIP is designed for cryptographic agility. Implementations MUST support runtime algorithm configuration, enabling migration to post-quantum algorithms (e.g., ML-DSA [FIPS-204]) as they are standardized. All RBIP data structures include schema_version to facilitate future algorithm upgrades without breaking existing archived evidence.

9.7. Supply Chain Attacks

RBIP Node software MUST be deployed with verifiable build pipelines. All RBIP software dependencies MUST be pinned to specific, audited versions. SBOM (Software Bill of Materials) generation MUST be part of the RBIP Node release process.

---

10. Privacy and Confidentiality Considerations

10.1. PII Prohibition

Personally identifiable information (PII) as defined by GDPR [GDPR] and CCPA MUST NOT appear in any RBIP Evidence Item in plaintext. All PII MUST be either:
(a) Replaced by a stable, institution-internal pseudonymous identifier hashed with SHA-256 (the "hash-and-reference" pattern), or
(b) Encrypted using AES-256-GCM [NIST-SP-800-38D] with a per-

customer encryption key managed in the HSM.

10.2. Selective Disclosure

The RBIP data model separates the cryptographic structure (evidence_id, prev_digest, data_digest, merkle_root, blockchain anchor) from the payload (data_fields). Auditors can verify the existence, integrity, and chronological ordering of events using only the cryptographic structure, without accessing data_fields. This enables a "prove without reveal" disclosure model.

10.3. Right to Erasure Compliance

GDPR Article 17 (Right to Erasure) creates a tension with RBIP's immutability requirements. RBIP addresses this as follows:

- All PII in data_fields MUST be encrypted under per-subject keys. "Erasure" of a subject's data is achieved by destroying the subject's encryption key, rendering data_fields permanently indecipherable without breaking the cryptographic structure.
- The cryptographic structure (digests, anchors) remains intact to preserve the audit chain.
- Institutions MUST maintain a documented legal basis mapping between RBIP evidence retention periods and applicable regulatory retention requirements.

10.4. Cross-Border Data Transfer

Evidence Bundles submitted to DOI archival providers (e.g., Zenodo/CERN, Switzerland) constitute cross-border data transfer under GDPR. Institutions MUST ensure data_fields containing PII are encrypted before submission to DOI providers, and that the decryption keys never leave the institution's jurisdiction. Effectively, the DOI provider archives only ciphertext for PII-containing fields.

---

11. Compliance with Regulatory Standards

This section describes how RBIP addresses specific regulatory requirements. This is informational guidance; institutions MUST seek independent legal and compliance counsel for their specific jurisdictions.

11.1. Basel III / Basel IV (BCBS)

PRL Module directly addresses:

- LCR (Liquidity Coverage Ratio) -- ReserveSnapshot captures HQLA values and LCR ratios required by BCBS 238.
- NSFR (Net Stable Funding Ratio) -- ReserveSnapshot includes NSFR ratio.
- Pillar 3 Disclosure -- RBIP Public Transparency API supports voluntary or mandatory Pillar 3 reserve disclosures.
- Pillar 2 ICAAP -- LiquidityStressTest events archive the methodology, assumptions, and outcomes of internal stress tests.

11.2. Sarbanes-Oxley Act (SOX) Section 404

LOC and KAL Modules support SOX 404 compliance by providing:

- Tamper-evident records of all material financial decisions (loan origination, underwriting, disbursement).
- Cryptographic evidence that records existed at the time of decision (blockchain timestamp).
- Non-repudiable officer attestations (ReserveAttestation).

11.3. Bank Secrecy Act (BSA) and AML

KAL Module directly addresses BSA requirements:

- 31 U.S.C. 5318(g) SAR filing obligations -- SARSubmission events archive metadata of SAR filings with cryptographic timestamps predating any regulatory inquiry.
- Customer Due Diligence (CDD) -- IdentitySubmission and IdentityVerificationResult provide tamper-evident CDD records.
- Transaction Monitoring -- TransactionMonitorAlert provides auditable evidence of the monitoring program's operation.

11.4. EU Digital Operational Resilience Act (DORA)

DORA [DORA2022] requires financial institutions to maintain comprehensive ICT risk management and incident registers. RBIP's tamper-evident evidence trail, blockchain timestamping, and DOI archival satisfy DORA requirements for:

- Article 9 -- Protection of ICT systems and data
- Article 10 -- Detection mechanisms
- Article 12 -- Backup policies
- Article 17 -- ICT-related incident reporting

11.5. EU Markets in Crypto-Assets Regulation (MiCA)

For institutions subject to MiCA [MiCA2023]:

- PRL Module provides the reserve attestation mechanism required for Asset-Referenced Token issuers under MiCA Title III.
- ReserveAttestation provides the cryptographic audit trail required for quarterly reserve audits under MiCA Article 37.

11.6. ISO/IEC 42001:2023 (AI Management Systems)

For institutions using AI-based credit scoring, AML detection, or risk models:

- LOC UnderwritingDecision includes decision_model_id, enabling traceability of AI model versions to decisions.
- KAL TransactionMonitorAlert includes rule_id for the AI/rules model that triggered the alert.
- RBIP provides the audit trail infrastructure required by ISO 42001 for AI decision auditability.

---

12. Implementation Guidance

12.1. Reference Implementation Notes

A conformant RBIP Node implementation MUST provide:

- Event adapter modules for common core banking systems (CBS). Reference adapters SHOULD be provided for ISO 20022 event feeds, T24 (Temenos), Flexcube (Oracle), and FIS Modern Banking Platform.

- A canonical JSON serializer conforming to [RFC8785] (JCS). Implementations MUST NOT use standard JSON pretty-printers, as these produce non-canonical output. Open-source JCS libraries are available for Java, Python, Go, and Rust.

- A Merkle tree implementation conforming to [RFC9162] Section 2.1 using SHA-256 leaves.

- A DOI archival client capable of submitting bundles and receiving DOIs from a DataCite-member archival provider. Implementors MAY use any conforming provider's deposit API. Example: the Zenodo REST API (https://zenodo.org/api/) was used for initial reference implementation archival.

- A blockchain anchoring client supporting a public, proof-of-work or proof-of-stake blockchain with publicly verifiable timestamping. The client MUST produce a proof artifact sufficient for offline verification per Section 6.4. Example: the OpenTimestamps Python client (https://github.com/opentimestamps/opentimestamps-client) was used for initial reference implementation anchoring.

- A FIPS 140-2 Level 3 HSM integration layer. Reference implementations SHOULD support PKCS#11 [PKCS11] and CNG (Windows CryptoNext Generation) interfaces for HSM abstraction.

- An audit log subsystem that applies RBIP evidencing recursively to its own access and operational logs.

12.2. Deployment Topologies

12.2.1. Centralized Deployment (Single RBIP Node)

Suitable for smaller institutions. A single RBIP Node processes all three modules. Evidence bundles are submitted to an external DOI provider and blockchain anchoring service.

12.2.2. Federated Deployment (Module-Separated Nodes)

Each RBIP module runs on a separate, isolated RBIP Node with separate HSM-backed signing keys. Module nodes submit bundles to a central Bundle Aggregator, which coordinates anchoring. Recommended for Tier 1 financial institutions.

12.2.3. Consortium Deployment

Multiple institutions share an RBIP Bundle Aggregator (operated by a trusted third party such as a central bank or industry utility). Each institution runs its own RBIP Node and submits bundles to the shared aggregator, which co-anchors bundles from multiple institutions in a single blockchain transaction. This amortizes anchoring costs and provides cross-institution chronological ordering.

---

13. IANA Considerations

This document requests the registration of the following media types with IANA:

13.1. application/rbip-evidence-item+json

```
Type name:            application
Subtype name:         rbip-evidence-item+json
Required parameters:  version (e.g., "rbip-01")
Optional parameters:  module ("PRL", "LOC", "KAL")
Encoding:             UTF-8
Security considerations: See Section 9
Interoperability considerations: None
Published specification: This document (draft-reilly-banking-integrity-01)
Author: Lawrence John Reilly Jr. <lreilly250@gmail.com>
```

13.2. application/rbip-bundle+json

```
Type name:            application
Subtype name:         rbip-bundle+json
Required parameters:  version (e.g., "rbip-01")
Encoding:             UTF-8
```

Security considerations: See Section 9
Author: Lawrence John Reilly Jr. <lreilly250@gmail.com>

13.3. application/rbip-tlpa+json

Type name:	application
Subtype name:	rbip-tlpa+json
Required parameters:	version (e.g., "rbip-01")
Encoding:	UTF-8

Security considerations: See Section 9 and 6.5
Author: Lawrence John Reilly Jr. <lreilly250@gmail.com>

---

14. Acknowledgments

The author thanks the financial cryptography, auditing, and regulatory technology communities for foundational work on tamper-evident ledgers, blockchain timestamping, and digital preservation that this protocol builds upon. The author acknowledges the open blockchain timestamping community for providing trust-minimized, publicly verifiable anchoring infrastructure, and the open DOI archival community for providing persistent identifier infrastructure that underpins RBIP’s DOI permanence layer.

This protocol specification is protected under triple-layer digital permanence per Section 1.3. The foundational whitepaper was DOI-archived on September 13, 2025 (DOI: 10.5281/zenodo.17114424), blockchain-timestamped via a public proof-of-work blockchain timestamping service, and submitted to the IETF on September 27, 2025 as draft-reilly-banking-integrity-00. This -01 revision was published March 21, 2026.

---

15. References

15.1. 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>.

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, DOI 10.17487/RFC3161, August 2001, <https://www.rfc-editor.org/info/rfc3161>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517,

DOI 10.17487/RFC7517, May 2015,
<https://www.rfc-editor.org/info/rfc7517>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518,
DOI 10.17487/RFC7518, May 2015,
<https://www.rfc-editor.org/info/rfc7518>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web
Token (JWT)", RFC 7519, DOI 10.17487/RFC7519,
May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[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>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON)
Data Interchange Format", STD 90, RFC 8259,
DOI 10.17487/RFC8259, December 2017,
<https://www.rfc-editor.org/info/rfc8259>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS)
Protocol Version 1.3", RFC 8446,
DOI 10.17487/RFC8446, August 2018,
<https://www.rfc-editor.org/info/rfc8446>.

[RFC8705] Campbell, B., Bradley, J., Sakimura, N., and T.
Lodderstedt, "OAuth 2.0 Mutual-TLS Client
Authentication and Certificate-Bound Access Tokens",
RFC 8705, DOI 10.17487/RFC8705, February 2020,
<https://www.rfc-editor.org/info/rfc8705>.

[RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON
Canonicalization Scheme (JCS)", RFC 8785,
DOI 10.17487/RFC8785, June 2020,
<https://www.rfc-editor.org/info/rfc8785>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object
Representation (CBOR)", STD 94, RFC 8949,
DOI 10.17487/RFC8949, December 2020,
<https://www.rfc-editor.org/info/rfc8949>.

[RFC9162] Meeker, A., Stark, E., and E. Kasper, "Certificate
Transparency Version 2.0", RFC 9162,
DOI 10.17487/RFC9162, December 2021,
<https://www.rfc-editor.org/info/rfc9162>.

[RFC9562] Davis, K., Peabody, B., and P. Leach, "Universally
Unique IDentifiers (UUIDs)", RFC 9562,
DOI 10.17487/RFC9562, May 2024,
<https://www.rfc-editor.org/info/rfc9562>.

[FIPS-180-4] National Institute of Standards and Technology,
"Secure Hash Standard (SHS)", FIPS PUB 180-4,
August 2015,
<https://csrc.nist.gov/publications/detail/fips/180/4/final>.

[FIPS-202] National Institute of Standards and Technology,
"SHA-3 Standard: Permutation-Based Hash and Extendable-
Output Functions", FIPS PUB 202, August 2015,
<https://csrc.nist.gov/publications/detail/fips/202/final>.

15.2. Informative References

[PKCS11] OASIS PKCS 11 TC, "PKCS #11 Cryptographic Token
Interface Base Specification Version 3.0",

OASIS Standard, June 2020,
<https://docs.oasis-open.org/pkcs11/pkcs11-base/v3.0/pkcs11-base-v3.0.html>
.

[I-D.draft-reilly-rem-protocol]
Reilly, L.J., "Reilly EternaMark (REM) Protocol:
Dual-Layer Digital Permanence via DOI Archival and
Blockchain Timestamping", Internet-Draft,
draft-reilly-rem-protocol (work in progress),
September 2025,
<https://datatracker.ietf.org/doc/draft-reilly-rem-protocol/>.

[I-D.draft-reilly-cts-00]
Reilly, L.J., "Cognitive Trust Stack (CTS): AI
Behavioral Provenance Framework", Internet-Draft,
draft-reilly-cts-00 (work in progress), 2026,
<https://datatracker.ietf.org/doc/draft-reilly-cts/>.

[RBIP-Whitepaper]
Reilly, L.J., "Reilly Banking Integrity Protocol
(RBIP): Ensuring Permanent and Regulator-Verifiable
Audit Trails", September 13, 2025,
DOI: 10.5281/zenodo.17114424,
<https://zenodo.org/records/17114424>.
NOTE: Archived at Zenodo as the initial reference
implementation of triple-layer permanence for this
specification. Implementations are not required to
use Zenodo; any DataCite-member archival provider
is conforming.

[OTS] Todd, P. et al., "OpenTimestamps: Scalable,
Trust-Minimized, Distributed Timestamping with Bitcoin",
<https://opentimestamps.org/>.
NOTE: Used as the initial blockchain timestamping
implementation for this specification. Implementations
are not required to use OpenTimestamps or Bitcoin;
any public blockchain anchoring service satisfying
Section 6.4 is conforming.

[DataCite4.5] DataCite Metadata Working Group, "DataCite
Metadata Schema Documentation for the Publication and
Citation of Research Data and Other Research Outputs",
Version 4.5, DOI 10.14454/g9e5-6293, 2022,
<https://schema.datacite.org/meta/kernel-4.5/>.

[FIPS-204] National Institute of Standards and Technology,
"Module-Lattice-Based Digital Signature Standard
(ML-DSA)", FIPS PUB 204, August 2024,
<https://csrc.nist.gov/pubs/fips/204/final>.

[NIST-SP-800-38D] Dworkin, M., "Recommendation for Block Cipher
Modes of Operation: Galois/Counter Mode (GCM) and GMAC",
NIST Special Publication 800-38D, November 2007,
<https://csrc.nist.gov/publications/detail/sp/800/38d/final>.

[GDPR] European Parliament and Council, "Regulation (EU)
2016/679 (General Data Protection Regulation)", OJ L
119/1, April 2016.

[DORA2022] European Parliament and Council, "Regulation (EU)
2022/2554 on Digital Operational Resilience for the
Financial Sector (DORA)", OJ L 333/1, December 2022.

[MiCA2023] European Parliament and Council, "Regulation (EU)
2023/1114 on Markets in Crypto-Assets (MiCA)",
OJ L 150/40, June 2023.

Author’s Address

Lawrence John Reilly Jr.
Independent Researcher
Email: lreilly250@gmail.com
IETF Datatracker: https://datatracker.ietf.org/person/lreilly250@gmail.com

This document is protected under triple-layer digital permanence
per Section 1.3:
* DOI: 10.5281/zenodo.17114424 (archived 2025-09-13)
* Blockchain timestamp via public proof-of-work network (2025-09-13)
* IETF submission: draft-reilly-banking-integrity-00 (2025-09-27)
* This version (-01): 2026-03-21