Network Working Group
Internet-Draft
Intended status: Informational
Expires: 22 July 2026

X. Zinn
Independent

18 January 2026

Message Envelope Format
draft-xzinn-message-envelope-format-00

Abstract

This document defines the message envelope format used for structured, verifiable message exchange.

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 22 July 2026.

Table of Contents

1. Introduction

Abstract

This document defines a message envelope format for identity based communication systems. The envelope provides a structured and verifiable container that binds declared purpose, payload, and cryptographic signatures into a single unit. The envelope format is independent of transport, encoding, and storage medium, and is designed to support deterministic verification, provenance tracking, and policy enforcement.

Introduction

Most communication systems treat messages as opaque payloads delivered over a transport. Meaning, intent, and authorization are inferred externally through application logic, metadata, or platform specific policy. This approach makes verification difficult, weakens provenance, and couples security decisions to implementation details rather than to explicit declarations.

This document defines a message envelope as a first class construct. A message envelope is a container that explicitly declares what a message is for, what content it carries, and who is responsible for it. These declarations are cryptographically bound so that they cannot be separated, reordered, or reinterpreted without detection.

The envelope format defined here does not assume a specific transport protocol, encoding, or storage mechanism. An envelope may be represented as a folder, a structured file, or a binary object. Regardless of representation, the semantics remain invariant.

2. Design Goals

The message envelope is designed to be processed before message content is interpreted. Verification of the envelope precedes payload handling, policy evaluation, and execution. This ordering allows systems to make deterministic decisions about acceptance, rejection, or quarantine without inspecting payload contents.

This document builds on the identity architecture and discovery mechanisms defined in earlier specifications. It does not define how messages are routed, delivered, or displayed. It defines only how a message is packaged so that downstream systems can reason about it safely.

Terminology

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in RFC 2119 and RFC 8174.

Envelope A structured container that binds declared purpose, payload, and cryptographic metadata into a single verifiable unit.

Message An instance of an envelope together with its associated payload. A message is considered incomplete unless its envelope verifies successfully.

Payload The content carried by an envelope. The payload may be a file, data stream, structured document, or reference to external data.

Purpose A declared semantic intent for a message. Purpose describes why a message exists and how it is expected to be handled, independent of payload content.

Envelope Header The portion of the envelope that contains declared purpose, metadata, and references necessary for verification and policy evaluation.

Envelope Body The portion of the envelope that contains or references the payload.

Binding The cryptographic association between envelope header, envelope body, and payload such that none can be modified independently without detection.

3. Envelope Structure

Signer An identity that applies a cryptographic signature to an envelope, asserting responsibility for its declared purpose and contents.

Verification The process of validating an envelope by checking signatures, bindings, and declared metadata prior to payload interpretation.

Representation The concrete encoding or packaging of an envelope, such as a directory structure, structured text format, or binary object.

Transport The mechanism by which an envelope is conveyed between parties. Transport is explicitly outside the scope of this document.

Recipient An identity that receives and verifies an envelope.

Intermediary A system that stores, forwards, or relays envelopes without asserting authorship or semantic authority.

Envelope Model

The message envelope is a logical construct that defines how a message is declared, verified, and handled. It establishes a strict ordering of interpretation that ensures security and policy decisions are made before any payload is processed.

An envelope consists of three conceptual components.

First, a header. The envelope header contains declarative information about the message, including its purpose, identifiers, metadata required for verification, and references to the payload. The header is designed to be small, explicit, and fully interpretable without accessing the payload.

Second, a body. The envelope body contains the payload itself or a reference to the payload. The body does not carry semantic meaning on its own and MUST NOT be interpreted until the envelope has been successfully verified.

4. Header Fields

Third, cryptographic bindings. The header and body are bound together through cryptographic mechanisms such that any modification to either component is detectable. These bindings ensure that declared purpose, payload, and signer assertions cannot be separated or recombined without invalidation.

The envelope model enforces a fixed processing order.

An implementation MUST first parse the envelope header. An implementation MUST then verify cryptographic bindings and signatures. Only after successful verification MAY the payload be accessed or interpreted.

If verification fails at any stage, the payload MUST NOT be processed. Handling of failed verification is defined by error and disposition semantics outside the scope of this document.

The envelope model intentionally decouples semantics from transport and storage. An envelope may be transmitted over a network, stored on disk, embedded within another structure, or represented as a directory hierarchy. Regardless of representation, the logical model remains the same.

The envelope model does not define application level meaning beyond declared purpose. Interpretation of payload contents is the responsibility of higher layer specifications and applications. This separation allows envelope verification and policy enforcement to be implemented uniformly across different message types.

Multiple envelopes MAY be nested or related, provided that each envelope independently satisfies the verification and binding requirements defined in this document. Nested envelopes MUST NOT weaken the verification guarantees of outer envelopes.

By enforcing declaration before interpretation and verification before execution, the envelope model provides a deterministic foundation for secure messaging, consent enforcement, and provenance tracking.

Envelope Model

The message envelope is a logical construct that defines how a message is declared, verified, and handled. It establishes a strict ordering of interpretation that ensures security and policy decisions are made before any payload is processed.

An envelope consists of three conceptual components.

5. Payload Semantics

First, a header. The envelope header contains declarative information about the message, including its purpose, identifiers, metadata required for verification, and references to the payload. The header is designed to be small, explicit, and fully interpretable without accessing the payload.

Second, a body. The envelope body contains the payload itself or a reference to the payload. The body does not carry semantic meaning on its own and MUST NOT be interpreted until the envelope has been successfully verified.

Third, cryptographic bindings. The header and body are bound together through cryptographic mechanisms such that any modification to either component is detectable. These bindings ensure that declared purpose, payload, and signer assertions cannot be separated or recombined without invalidation.

The envelope model enforces a fixed processing order.

An implementation MUST first parse the envelope header. An implementation MUST then verify cryptographic bindings and signatures. Only after successful verification MAY the payload be accessed or interpreted.

If verification fails at any stage, the payload MUST NOT be processed. Handling of failed verification is defined by error and disposition semantics outside the scope of this document.

The envelope model intentionally decouples semantics from transport and storage. An envelope may be transmitted over a network, stored on disk, embedded within another structure, or represented as a directory hierarchy. Regardless of representation, the logical model remains the same.

The envelope model does not define application level meaning beyond declared purpose. Interpretation of payload contents is the responsibility of higher layer specifications and applications. This separation allows envelope verification and policy enforcement to be implemented uniformly across different message types.

Multiple envelopes MAY be nested or related, provided that each envelope independently satisfies the verification and binding requirements defined in this document. Nested envelopes MUST NOT weaken the verification guarantees of outer envelopes.

By enforcing declaration before interpretation and verification before execution, the envelope model provides a deterministic foundation for secure messaging, consent enforcement, and provenance tracking.

Signing and Binding Rules

6. Signing and Integrity

Signing and binding rules define how the elements of an envelope are cryptographically associated to ensure integrity, authenticity, and non separation of declared semantics and payload.

An envelope MUST be signed by the issuer identity identified in the envelope header. The signature asserts responsibility for the declared purpose, payload reference, and associated metadata.

The signature MUST bind, at a minimum, the following elements.

’ Envelope Identifier ’ Declared Purpose ’ Issuer Identity ’ Creation Time ’ Payload Reference ’ Optional Metadata present in the header

If any bound element is modified, added, or removed, signature verification MUST fail.

The payload itself MUST be included in the binding either directly or indirectly. Direct inclusion binds the payload content itself. Indirect inclusion binds a cryptographic digest of the payload. Implementations MAY choose either method based on payload size, transport constraints, or storage model, provided that modification of the payload is detectable.

Signatures MUST be verifiable without access to external policy state, consent stores, or application specific context. Verification relies solely on the envelope contents and identity verification mechanisms defined elsewhere.

Multiple signatures MAY be applied to an envelope. When multiple signatures are present, each signature MUST independently satisfy the binding requirements. The presence of multiple signatures does not alter the declared purpose unless explicitly defined by a higher layer specification.

Detached signatures are permitted, provided that the binding between signature material and envelope elements is explicit and unambiguous. Detached signatures MUST NOT allow envelope elements to be recombined with different payloads or purposes.

Signing order MUST be deterministic. Implementations MUST define a canonical representation of bound elements prior to signing to ensure consistent signature results across

Representation Independence

7. Encryption Considerations

The message envelope is defined independently of any specific encoding, file format, or transport. This document specifies logical structure and binding rules, not concrete serialization.

An envelope MAY be represented in multiple forms, including but not limited to a directory hierarchy, a structured text document, or a binary object. Regardless of representation, the envelope MUST preserve the logical elements and binding relationships defined in this specification.

All representations MUST support deterministic canonicalization of bound elements. Canonicalization ensures that equivalent envelope contents produce identical cryptographic verification results across different implementations and encodings.

Representations MUST NOT introduce implicit fields, inferred semantics, or hidden metadata that affect envelope verification. Any information that influences verification or declared purpose MUST be explicitly present in the envelope header and included in the cryptographic binding.

When envelopes are transformed between representations, the transformation process MUST preserve all bound elements exactly. Transformations that alter envelope identifiers, declared purpose, payload references, or signature material invalidate the envelope.

Implementations MAY support multiple representations concurrently. Support for one representation MUST NOT imply support for others, provided that the logical envelope model remains consistent.

Representation independence allows envelopes to be stored, transmitted, archived, and processed across heterogeneous systems without weakening verification guarantees. It also enables gradual evolution from human readable representations to compact binary forms without changing semantics.

This document does not define canonical encodings. Representation specific profiles may define concrete formats and canonicalization rules, provided that they conform to the logical model and binding requirements defined here.

Security Considerations

The message envelope format defined in this document establishes security properties through explicit declaration, cryptographic binding, and deterministic verification. These properties are intended to reduce ambiguity, prevent substitution attacks, and enable early rejection of invalid or unauthorized messages.

A primary security objective is semantic integrity. Declared purpose, payload reference, and signer identity are cryptographically bound such that they cannot be altered independently. This prevents an attacker from reusing a valid payload with a different declared purpose or signer.

8. Processing Rules

Verification MUST occur before payload processing. Implementations that access or interpret payload content prior to successful envelope verification risk exposing themselves to malicious or malformed data. Envelope verification is therefore a mandatory first step.

Replay attacks are possible when envelopes are reused outside their intended context. While this document does not define replay prevention mechanisms, envelope identifiers and creation times provide inputs that higher layer specifications MAY use to detect or limit replay.

The envelope format does not assume trust in intermediaries. Envelopes may be stored, forwarded, or relayed by untrusted systems without weakening integrity or authenticity guarantees. Intermediaries MUST NOT be relied upon to enforce envelope semantics.

Compromise of a signing key undermines the integrity of envelopes issued by that identity. Identity systems implementing this specification MUST define mechanisms for key rotation and revocation. Envelope verification SHOULD take revocation status into account where such mechanisms are available.

Multiple signatures increase complexity and MUST be handled carefully. Implementations SHOULD define clear rules for interpreting envelopes with multiple signers to avoid ambiguity or unintended authorization.

The envelope format deliberately excludes policy decisions, consent evaluation, and execution semantics. These concerns are handled by higher layer specifications. Mixing these concerns into the envelope would weaken determinism and increase attack surface.

By enforcing declaration before interpretation and binding before execution, the envelope format reduces the risk of confused deputy attacks and unintended side effects caused by implicit or inferred message semantics.

The ability to observe, inspect, verify, relay, or replicate a message envelope does not confer authority to mutate, decrypt, authorize, sanction, or finalize the associated content.

Privacy Considerations

The message envelope format is designed to support privacy preserving communication by minimizing implicit metadata and enforcing explicit declaration of semantics.

The envelope header contains only information required for verification and policy evaluation. Implementations SHOULD avoid including extraneous metadata that could be used to infer communication patterns, relationships, or behavioral characteristics beyond what is explicitly declared.

9. Security Considerations

Declared purpose enables early decision making without inspecting payload content. This reduces the need for deep inspection and limits exposure of sensitive payload data to systems that are not authorized to process it.

Envelope verification does not require access to centralized services or global state. As a result, intermediaries and storage systems are not required to observe or record identity relationships in order to process envelopes. This reduces opportunities for large scale aggregation of communication metadata.

Payload references MAY point to external content. Implementations SHOULD ensure that resolution of such references does not inadvertently reveal recipient identity, access patterns, or message context to unauthorized parties.

Multiple representations of an envelope SHOULD preserve privacy properties consistently. Transformations between representations MUST NOT introduce additional metadata, ordering information, or implicit identifiers that could enable correlation across messages.

The envelope format itself does not mandate logging, telemetry, or reporting. Any such behavior is an implementation choice and SHOULD be carefully evaluated against privacy requirements and threat models.

By making semantics explicit and verifiable, the envelope format enables systems to reject or accept messages deterministically without relying on inference, heuristics, or inspection of content, thereby reducing unnecessary data exposure.

IANA Considerations

This document does not define any new namespaces, registries, message types, protocol numbers, port assignments, or parameters that require registration with IANA.

If future specifications define concrete envelope representations, serialization formats, or registry based purpose vocabularies that build upon the envelope model defined here, those documents will specify their own IANA considerations.

References

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.

<<RFC8174>> Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017.

Informative References

<<RFC3986>> Berners Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005.

<<RFC7258>> Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014.

<<RFC4108>> Hansen, T., "Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages", RFC 4108, DOI 10.17487/RFC4108, July 2005.