Independent Submission
Internet-Draft
Intended status: Experimental
Expires: 1 September 2026

A. Cowles
Quox Ltd

28 February 2026

Agent Envelope Exchange (AEE): A Minimal JSON Envelope Format for Inter-Agent Communication
draft-cowles-aee-00

Abstract

Agent Envelope Exchange (AEE) defines a minimal, transport-independent JSON envelope format for communication between autonomous AI agents, traditional services, and human participants. The envelope comprises 14 well-defined fields that provide message identity, typing, correlation, tracing, priority, and extensibility without prescribing payload semantics or transport mechanisms. AEE separates the concerns of message routing and lifecycle management (the envelope) from domain-specific meaning (intent schemas), enabling portable, composable, and auditable workflows across heterogeneous agent frameworks. This document specifies the envelope structure, validity rules, conformance levels, entity identifier conventions, a reserved intent namespace for protocol negotiation, and a referencing strategy that avoids envelope nesting.

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 1 September 2026.

Copyright Notice

Table of Contents

1. Introduction

The rapid proliferation of autonomous AI agent frameworks -- including LangGraph, AutoGen, CrewAI, and numerous proprietary systems -- has created a fragmented landscape for inter-agent communication. Each framework defines its own message structures, correlation mechanisms, and error conventions, making it difficult to compose workflows that span multiple agent runtimes or organizational boundaries.

Existing approaches suffer from several deficiencies:

* *No standard envelope format:* Messages lack a consistent structure for identity, typing, and metadata, forcing each integration to define bespoke serialization.

* *Lost causality:* Without standardized correlation and reply-to semantics, tracing a request through a chain of agents requires framework-specific tooling.

* *Coupled semantics:* Many systems conflate the message container with the message content, making it impossible to validate routing independently of payload.

* *Human exclusion:* Most agent protocols treat human participants as external to the messaging model rather than as first-class senders and recipients.

AEE addresses these problems by defining a minimal 14-field JSON envelope that standardizes message identity, type classification, sender/recipient addressing, intent declaration, correlation, distributed tracing hooks, priority signaling, execution constraints, payload carriage, and optional cryptographic signatures. The envelope is deliberately small and stable: it provides just enough structure for messages to be routed, correlated, traced, and validated, while delegating all domain-specific semantics to intent schemas that evolve independently.

AEE does not prescribe a transport mechanism. Envelopes MAY be carried over HTTP, WebSocket, NATS, Kafka, Redis Streams, or any other reliable or unreliable channel. The protocol is intentionally transport-agnostic to maximize adoption across diverse deployment topologies.

2. Conventions and Definitions

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

The following terms are used throughout this document:

Envelope The top-level JSON object defined by this specification, containing metadata fields and a payload.

Intent A namespaced identifier declaring the semantic purpose of a message. The intent determines which payload schema applies.

Entity Any participant in AEE communication: an autonomous agent, a traditional service, a human operator, or a broadcast channel.

Correlation The mechanism by which related messages (e.g., a task and its result) are linked via shared identifiers.

MVE Minimal Viable Envelope. A subset of envelope fields sufficient for a particular conformance tier.

3. Envelope Format

An AEE message is a JSON object [RFC8259] with exactly 14 top-level fields. The envelope is designed to be small, stable, and forward-compatible. The meaning of a message is determined by the combination of its "type" and "intent" fields; the "payload" carries intent-specific data whose schema is defined externally.

#	Field	Type	Required	Description
1	v	string	Yes	Envelope schema version. For this specification, the value MUST be "1".
2	id	string	Yes	Unique message identifier. ULID or UUID [RFC4122] is RECOMMENDED. Minimum length: 8 characters.
3	ts	string	Yes	Timestamp of message creation. ISO 8601 UTC

				format is RECOMMENDED (e.g., "2025-12-14T03:45:12Z"). Minimum length: 10 characters.
4	type	string	Yes	Message type. MUST be one of: "task", "result", "event", "error", "stream".
5	from	string	Yes	Sender entity identifier (see Section 5). Minimum length: 1 character.
6	to	string	Yes	Recipient entity identifier or channel (see Section 5). Minimum length: 1 character.
7	intent	string	Yes	Namespaced intent identifier declaring the semantic purpose of the message (see Section 9). Minimum length: 3 characters.
8	corr	string	Yes	Correlation identifier linking all messages in a workflow or conversation thread. Minimum length: 8 characters.
9	reply_to	string or null	Conditional	The "id" of the message being replied to. MUST be non-null (minimum 8 characters) when "type" is "result" or "error". SHOULD be null for other types.
10	trace	object or null	No	Distributed tracing context with "trace_id" and "span_id" string properties. RECOMMENDED

				when an observability system is in use.
11	priority	string	Yes	Message priority. MUST be one of: "low", "normal", "high", "urgent".
12	requires	object or null	No	Execution constraints and expectations. Common keys include "timeout_ms" (number), "human_approval" (boolean), "evidence" (boolean), "format" (string), and "min_confidence" (number, 0 to 1). Unknown keys MUST be ignored by consumers.
13	payload	object	Yes	Intent-specific data. The shape is defined by the intent schema, not by this specification. MUST be a JSON object.
14	sig	object, string, or null	No	Optional cryptographic signature or authentication proof (see Section 8).

Table 1: AEE Envelope Fields

The following example shows a complete AEE envelope of type "task":

```
{
  "v": "1",
  "id": "01JFB2R1JZKQ9V3K8W8Y9W1F2A",
  "ts": "2025-12-14T03:45:12Z",
  "type": "task",
  "from": "agent.manager",
  "to": "agent.backup_auditor",
  "intent": "ops.backup.status.check",
  "corr": "01JFB2QX0K8X5K6ZJ9G2C0C1MW",
  "reply_to": null,
  "trace": {"trace_id": "9f3c", "span_id": "a12b"},
  "priority": "high",
  "requires": {
    "timeout_ms": 30000,
    "evidence": true,
    "human_approval": false
  },
  "payload": {"cluster": "node.lan", "window": "24h"},
  "sig": null
}
```

3.1. JSON Schema

A normative JSON Schema (draft 2020-12) for the AEE v1 envelope is published alongside this specification. The schema validates envelope structure only; intent-specific payload validation is performed by separate schemas associated with each intent.

The schema enforces all required fields, type constraints, enumeration values, and the conditional requirement that "reply_to" MUST be a non-null string of at least 8 characters when "type" is "result" or "error". The schema sets "additionalProperties" to true at the top level to support forward compatibility.

4. Message Types

The "type" field classifies the purpose and expected behavior of each envelope. Exactly five message types are defined:

task A request to perform work. A "task" message sets an expectation that the recipient will eventually respond with either a "result" or an "error" envelope sharing the same "corr" value and a "reply_to" referencing this task's "id". The sender MAY specify execution constraints in the "requires" field (e.g., timeout, required evidence, human approval).

result Successful completion of a task. The "reply_to" field MUST

contain the "id" of the originating "task" envelope. The "payload" carries the intent-specific output of the completed work.

event An informational signal that does not require a reply. Events are used for notifications, status broadcasts, and telemetry. The "reply_to" field SHOULD be null.

error Unsuccessful completion of a task. The "reply_to" field MUST contain the "id" of the originating "task" envelope. The "payload" SHOULD include a "code" string, a human-readable "message" string, and a "retryable" boolean. Additional fields such as "backoff_ms" and "evidence_refs" MAY be included.

stream A partial or progress update related to a task or long-running process. Stream messages allow the recipient to observe intermediate state without waiting for the final result. Multiple "stream" envelopes MAY share the same "corr" and precede the terminal "result" or "error".

The following example shows a "result" envelope responding to the task shown in Section 3:

```
{
  "v": "1",
  "id": "01JFB2S7T8N4J8B7QH1GJ8Z1Y2",
  "ts": "2025-12-14T03:45:20Z",
  "type": "result",
  "from": "agent.backup_auditor",
  "to": "agent.manager",
  "intent": "ops.backup.status.check",
  "corr": "01JFB2QX0K8X5K6ZJ9G2C0C1MW",
  "reply_to": "01JFB2R1JZKQ9V3K8W8Y9W1F2A",
  "trace": {"trace_id": "9f3c", "span_id": "b77c"},
  "priority": "high",
  "requires": {"evidence": true},
  "payload": {
    "status": "PARTIAL_FAILURE",
    "failed": [
      {"node": "pve02", "reason": "connection_refused:8007"}
    ],
    "confidence": 0.96,
    "evidence_refs": ["log:pbs01:job/2025-12-14T02:00Z"]
  },
  "sig": null
}
```

The following example shows an "error" envelope:

```
{
  "v": "1",
  "id": "01JFB2T3W9H0J1FQWQ7B2M3D5E",
  "ts": "2025-12-14T03:45:21Z",
  "type": "error",
  "from": "agent.backup_auditor",
  "to": "agent.manager",
  "intent": "ops.backup.status.check",
  "corr": "01JFB2QX0K8X5K6ZJ9G2C0C1MW",
  "reply_to": "01JFB2R1JZKQ9V3K8W8Y9W1F2A",
  "trace": {"trace_id": "9f3c", "span_id": "c01d"},
  "priority": "high",
  "requires": {"evidence": true},
  "payload": {
    "code": "E_TIMEOUT",
    "message": "PBS API did not respond within 30s",
    "retryable": true,
    "backoff_ms": 60000,
    "evidence_refs": ["tcp:pve02->pbs01:8007"]
  },
  "sig": null
}
```

5. Entity Identifiers

The "from" and "to" fields accept any entity identifier string. AEE treats all participants -- autonomous agents, traditional services, human operators, and broadcast channels -- as first-class senders and recipients. There is no distinction at the envelope level between a message sent by a human and one sent by an agent.

5.1. Identifier Format

The RECOMMENDED identifier format is:

<type>.<name>

where "type" indicates the entity category and "name" is a locally unique identifier within that category. Implementations MAY use any unique string, but the typed format improves debuggability and log readability.

5.2. Common Type Prefixes

The following type prefixes are defined as conventions. This list is not exhaustive; implementations MAY define additional prefixes.

agent.* Autonomous agents (e.g., "agent.backup_auditor",

"agent.code_reviewer").

service.* Traditional (non-agent) services (e.g., "service.billing", "service.auth").

human.* Human participants (e.g., "human.adam", "human.ops_lead").

bus.* Broadcast channels or topic-based routing targets (e.g., "bus.ops", "bus.alerts").

5.3. Causality Tracing Across Entity Types

When a human initiates a task that cascades through multiple agents, the full causality chain is traceable through the combination of "from", "to", "corr", and "reply_to" fields. For example, a chain such as "human.adam" to "agent.router" to "agent.reviewer" to "service.linter" maintains complete provenance regardless of entity type at each hop.

6. Validity Rules

The following rules are normative. An AEE envelope that violates any MUST-level rule is invalid and SHOULD be rejected by conforming implementations.

1. The fields "v", "id", "ts", "type", "from", "to", "intent", "corr", "priority", and "payload" MUST be present in every envelope.

2. The "type" field MUST contain one of the following values: "task", "result", "event", "error", "stream".

3. If "type" is "result" or "error", the "reply_to" field MUST be present and MUST be a non-null string with a minimum length of 8 characters, corresponding to the "id" of the originating "task" envelope.

4. The "payload" field MUST be a JSON object. The structure of the payload is defined by the intent schema associated with the "intent" value.

5. The "v" field MUST contain the string "1" for envelopes conforming to this version of the specification.

6. The "priority" field MUST contain one of the following values: "low", "normal", "high", "urgent".

7. Unknown top-level fields MUST be ignored by consumers. This ensures forward compatibility: future versions or extensions may add fields without breaking existing implementations.

8. Consumers SHOULD log at minimum the following fields for observability: "id", "corr", "intent", "from", "to", "type", and "priority".

9. Producers SHOULD include the "trace" field with "trace_id" and "span_id" values when an observability or distributed tracing system is in use.

10. Within the "requires" object, unknown keys MUST be ignored by consumers. Producers SHOULD keep the "requires" object compact.

7. Conformance Levels

AEE supports incremental adoption through two defined conformance tiers. Not every system needs full specification compliance on initial deployment.

7.1. MVE-Required (Schema-Valid)

The Minimal Viable Envelope (MVE-Required) tier includes all fields required by the JSON Schema:

* "v"
* "id"
* "ts"
* "type"
* "from"
* "to"
* "intent"
* "corr"
* "priority"
* "payload"

A system emitting envelopes with all 10 MVE-Required fields is fully schema-valid and interoperable with any AEE-compliant consumer. A system MAY claim MVE-Required compliance if:

1. It emits envelopes with all 10 required fields.

2. Those envelopes validate against the AEE v1 JSON Schema.

3. The "reply_to" field is non-null for "result" and "error" types.

MVE-Required certification attests to structural compliance only. It does not certify payload schema conformance, semantic correctness of intents, or support for the "aee.*" protocol intents defined in Section 9.

7.2. MVE-5 (Log-Friendly)

For logging, telemetry, and observability-only contexts, a 5-field subset is recognized:

* "v"

* "id"

* "type"

* "from"

* "intent"

MVE-5 is *NOT* schema-valid AEE. It is a convenience envelope for systems that only emit logs or telemetry and do not participate in request/response flows. MVE-5 envelopes:

* Can be parsed by AEE-aware log aggregators.

* Cannot be used for task/result correlation (missing "corr" and "reply_to").

* SHOULD NOT be sent to agents expecting full envelopes.

8. Signatures

The "sig" field provides an OPTIONAL mechanism for cryptographic authentication and integrity verification of AEE envelopes. When signatures are used, the signing process SHOULD bind at minimum the following fields:

* "v"
* "id"
* "ts"
* "type"
* "from"
* "to"
* "intent"
* "corr"
* "reply_to"
* "payload"

The "sig" field MAY be a JSON object or a string. When it is an object, the following structure is RECOMMENDED:

```
{
  "alg": "ed25519",
  "kid": "key-2",
  "value": "<base64-encoded-signature>"
}
```

alg The signature algorithm. Ed25519 is RECOMMENDED for new implementations. HMAC-SHA256 ("HS256") MAY also be used in symmetric-key environments.

kid Key identifier, enabling key rotation and multi-key environments.

value The signature or HMAC value, typically Base64-encoded.

This specification does not define a key distribution mechanism. Deployments SHOULD establish key management procedures appropriate to their trust model. Future companion documents may define a security profile with detailed signing procedures, key exchange, and replay protection.

9. Intent Namespace

The "intent" field carries a namespaced identifier that declares the semantic purpose of a message. The intent determines which payload schema applies: each intent publishes separate schemas for "task", "result", and (optionally) "error" payloads. This separation means the envelope can be validated independently of payload content.

9.1. Naming Conventions

The RECOMMENDED naming format uses flexible dotted notation:

<domain>.<subdomain>.<noun>.<verb>

Examples:

* "ops.network.port.probe"

* "ops.backup.status.check"

* "infra.proxmox.vm.create"

* "docs.summarize.with_citations"

Verbs SHOULD be explicit (e.g., "check", "probe", "create", "summarize"). Implementations SHOULD avoid overloading a single intent with multiple optional behaviors. Intent schemas MAY be versioned independently of the AEE envelope specification.

If two agents cannot agree on an intent identifier, they do not share semantics. The absence of a recognized intent is not an error; it simply indicates that the agents lack a shared vocabulary for that operation.

9.2. Reserved Namespace: aee.*

The "aee.*" namespace is reserved for protocol negotiation and context retrieval. These intents enable agents to discover capabilities, fetch context, and validate payloads without custom integration. They are protocol primitives, not orchestration mechanisms.

The following core intents are defined:

Intent	Purpose
aee.status.ping	Liveness check. The recipient MUST respond with a result containing {"pong": true}.
aee.status.health	Health and readiness status. The result payload MUST include a "status" field with one of: "healthy", "degraded", "unhealthy".
aee.spec.query	Return supported AEE version and capabilities. The result MUST include "aee_version" and MAY include an "extensions" array.
aee.capability.list	List supported intents. The task MAY include a "namespace_filter" string. The result MUST include an "intents" array of strings.
aee.context.fetch	Retrieve an envelope or payload by reference. The task MUST include "id" and MAY include "corr", "locator", and "hash" for verification.
aee.context.refute	Reject a referenced context with a reason. The task MUST include "id" and "reason". The result MUST include "acknowledged" (boolean).
aee.validate.payload	Validate a payload against an intent schema. The task MUST include "intent" and "payload_to_validate". The result MUST include "valid" (boolean) and MAY include an "errors" array.

Table 2: Reserved aee.* Protocol Intents

Implementations claiming AEE compliance are not required to implement all "aee.*" intents. However, "aee.status.ping" SHOULD be supported by all agents to enable basic liveness detection.

Application-specific intents MUST NOT use the "aee.*" namespace. If a use case requires scheduling, workflow, or orchestration semantics, those intents belong in an application namespace (or in a companion protocol such as AOCL [AOCL]).

9.3. Intent Schema Publishing

AEE does not standardize the "payload" field's internal structure. Instead, each intent publishes separate JSON Schema documents for its "task", "result", and "error" payload variants. The RECOMMENDED file layout is:

```
envelope/aee-v1.schema.json
intents/<intent>.task.schema.json
intents/<intent>.result.schema.json
intents/<intent>.error.schema.json
```

AEE does not require a centralized intent registry. Communities MAY publish intent schemas in version-controlled repositories, package registries, or any other distribution mechanism appropriate to their ecosystem.

10. Referencing Strategy

When an envelope needs to reference prior context (e.g., the original task that spawned a result, or related evidence from another workflow), AEE prescribes a reference-based approach rather than embedding full envelopes within payloads.

10.1. Rationale: Avoiding Nested Envelopes

Embedding full envelopes inside payloads creates several problems:

* *Matryoshka bloat:* Deeply nested structures grow with each hop in a multi-agent workflow, consuming storage and bandwidth.

* *Token collapse:* Large language models (LLMs) processing nested envelopes waste context window tokens on redundant structural metadata.

* *Signature ambiguity:* When envelopes are nested, it becomes unclear which envelope's signature is authoritative -- the outer or the inner.

10.2. The payload.references Array

When referencing prior envelopes, implementations SHOULD use a "references" array within the payload:

```
{
  "payload": {
    "references": [
      {
        "id": "01JFB2R1JZKQ9V3K8W8Y9W1F2A",
        "corr": "01JFB2QX0K8X5K6ZJ9G2C0C1MW",
        "type": "aee.envelope",
        "locator": "store://envelopes/01JFB2R1JZKQ9V3K8W8Y9W1F2A",
        "hash": "sha256:a1b2c3..."
      }
    ]
  }
}
```

Each reference object contains the following fields:

id (REQUIRED) The referenced envelope's "id" value.

corr (OPTIONAL) The correlation ID for additional context.

type (REQUIRED) The reference type. For envelope references, the value MUST be "aee.envelope".

locator (OPTIONAL) A URI or path that can be used to retrieve the full referenced envelope, for example via the "aee.context.fetch" intent.

hash (OPTIONAL) A content hash (e.g., "sha256:...") for integrity verification of the referenced envelope.

To retrieve the full envelope behind a reference, consumers SHOULD use the "aee.context.fetch" protocol intent (see Section 9).

10.3. Exception: Archival and Forensic Snapshots

When creating immutable audit logs or forensic snapshots, embedding full envelopes as inert blobs within a "result" payload is acceptable. These embedded envelopes are historical records, not active routing participants. Implementations MUST NOT attempt to route or process embedded archival envelopes as live messages.

11. Security Considerations

11.1. Replay Attacks and Correlation ID Uniqueness

Each AEE envelope carries a unique "id" field and a "corr" field linking related messages. If message identifiers are predictable or reused, an attacker could replay previously captured envelopes to trigger duplicate work or corrupt workflow state. Implementations MUST generate "id" values using cryptographically random or lexicographically-sortable unique identifiers (e.g., ULIDs or UUIDs [RFC4122]). Consumers SHOULD maintain a deduplication window and reject envelopes with previously seen "id" values within that window. The "ts" field SHOULD be checked against a reasonable clock-skew tolerance to detect stale replays.

11.2. Signature Verification and Trust Model

The "sig" field is OPTIONAL, meaning that unsigned envelopes are valid per this specification. Deployments operating in environments where message authenticity is critical MUST require signatures and establish a trust model for key distribution. Ed25519 is RECOMMENDED for asymmetric signatures due to its performance characteristics and resistance to implementation pitfalls. When signatures are present, consumers MUST verify them before processing the envelope. A failed signature verification MUST result in rejection of the envelope.

This specification does not define a public key infrastructure (PKI) or key exchange mechanism. Deployments SHOULD document their key management procedures, including key rotation schedules, revocation mechanisms, and the scope of trust for each key identifier ("kid").

11.3. Envelope Integrity Without Signatures

In deployments where the "sig" field is not used, envelope integrity depends entirely on the security properties of the transport layer. If envelopes are carried over an unencrypted or unauthenticated channel, any intermediary can modify envelope fields -- including "from", "to", "intent", and "payload" -- without detection. Deployments that do not use envelope-level signatures MUST ensure that the transport layer provides adequate integrity and authentication guarantees (e.g., TLS with mutual authentication). For tamper-evident recording of envelope exchanges, the companion VOLT protocol [VOLT] defines cryptographic hash chains that can provide post-hoc verification of envelope integrity.

11.4. Privacy of Payload Contents

AEE envelopes carry arbitrary payload data that may include personally identifiable information (PII), credentials, internal system details, or other sensitive content. The envelope format does not provide payload encryption. Implementations handling sensitive data SHOULD encrypt payloads at the application layer or ensure that the transport provides confidentiality (e.g., TLS). The "requires" field MUST NOT be used to carry secrets; credentials should be transmitted through secure side-channels rather than embedded in envelope metadata.

The logging guidance in Section 6 (rule 8) recommends logging several envelope fields for observability. Implementations MUST ensure that logged fields do not inadvertently expose sensitive payload contents. The "intent" field, while not typically sensitive, may reveal operational patterns; deployments SHOULD consider the information leakage implications of intent names in log aggregation systems.

11.5. Trust Boundaries Between Agents

AEE entity identifiers (see Section 5) are self-asserted: the "from" field is populated by the sender, and there is no built-in mechanism to verify that the claimed sender identity is authentic. In multi-tenant or cross-organizational deployments, a malicious agent could impersonate another agent by setting an arbitrary "from" value. Deployments MUST implement identity verification at trust boundaries, either through envelope signatures (see Section 8), transport-layer authentication, or a gateway that validates sender identity before forwarding envelopes.

When envelopes cross trust boundaries (e.g., from an internal agent network to an external partner’s agents), implementations SHOULD validate all envelope fields at the boundary, strip or redact sensitive payload content, and apply allowlist-based intent filtering to prevent unauthorized operations.

11.6. Denial of Service via Priority Field Abuse

The "priority" field allows senders to signal message urgency. A malicious or misconfigured agent could set all messages to "urgent" priority, starving normal-priority messages of processing resources. Implementations SHOULD enforce per-sender priority quotas or rate limits. Infrastructure components SHOULD NOT blindly trust the "priority" field from untrusted sources; instead, they SHOULD apply policy-based priority assignment or capping at trust boundaries.

Similarly, the "requires" field's "timeout_ms" value could be abused to hold resources for extended periods. Consumers SHOULD enforce maximum timeout values regardless of the requested timeout, and SHOULD release resources promptly when timeouts expire.

12. IANA Considerations

This document has no IANA actions.

13. References

13.1. Normative References

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

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

[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>.

13.2. Informative References

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[AOCL] Cowles, A., "Agent Orchestration Control Layers (AOCL)", Work in Progress, Internet-Draft, draft-cowles-aocl-00, 2026, <https://github.com/AdaminX/AOCL-Agent-Orchestration-Control-Layers-Protocol>.

[VOLT] Cowles, A., "Verifiable Operations Ledger and Trace (VOLT)", Work in Progress, Internet-Draft, draft-cowles-volt-00, 2026, <https://github.com/AdaminX/VOLT-Protocol>.

Acknowledgements

The author wishes to thank the early adopters and reviewers of the AEE specification within the Quox ecosystem, whose feedback on real-world agent interoperability challenges shaped the design of this envelope format.

Author’s Address

Adam Cowles
Quox Ltd
London
United Kingdom
Email: adam@quox.ai
URI: https://quox.ai