Independent Submission
Internet-Draft
Intended status: Informational
Expires: 22 November 2026

L. K. Kroehl
CryptoKRI GmbH

21 May 2026

Agent Authorization Envelope (AAE): A Machine-Evaluable Authorization Structure for Autonomous AI Agents
draft-kroehl-agentic-trust-aae-00

Abstract

Autonomous AI agents now operate at production scale across financial, commercial, and infrastructure domains — executing transactions, invoking APIs, and taking consequential actions without direct human oversight at each step. Existing authorization mechanisms (OAuth 2.0, API keys, ACLs) were designed for human-initiated requests and do not capture the machine-evaluable semantics required for autonomous agent authorization: what the agent is mandated to do, what constraints bound its actions, and for how long the authorization is valid.

This document specifies the Agent Authorization Envelope (AAE), a structured authorization container for autonomous AI agents. AAE defines three mandatory blocks — MANDATE, CONSTRAINTS, and VALIDITY — that together constitute a machine-evaluable, cryptographically verifiable authorization assertion. AAE is designed to be protocol-agnostic, binding to W3C Decentralized Identifiers (DIDs) for agent identity and W3C Verifiable Credentials (VCs) for issuance and signature, and is independent of any specific AI framework, transport protocol, or blockchain.

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 November 2026.

Table of Contents

1. Introduction

The deployment of autonomous AI agents at scale creates an authorization gap that existing Internet protocols do not address. When a human user initiates a request, the authorization question is well-understood: who is the user, what are they permitted to do, and has their session expired? Protocols such as OAuth 2.0 [RFC6749] answer these questions for human-delegated access.

When an autonomous agent initiates a request — potentially on behalf of another agent, under a delegated mandate, with specific constraints on permissible actions — the question requires a different answer. The agent may be acting under a mandate issued hours earlier; its actions may be bounded by value caps, action allowlists, or domain restrictions; and the authorization may expire on a schedule that the receiving system must be able to verify without contacting the issuing party.

No current IETF standard addresses this combination of requirements for autonomous agents. OAuth 2.0 scopes are issuer-defined strings without machine-evaluable semantics for agent mandates. SPIFFE/SVID addresses workload identity but not authorization semantics. JWT claims are flexible but unstructured for agent-specific use cases.

This document specifies the Agent Authorization Envelope (AAE) to fill this gap. AAE is derived from a production deployment operational since March 2026 [ARXIV-AAE] and is aligned with regulatory requirements from Singapore IMDA [IMDA-MGF], NIST [NIST-CAISI], and the EU AI Act (Regulation 2024/1689).

1.1. Regulatory Convergence

Independent regulatory work has converged on the same authorization structure that AAE implements. Two requirements from the Singapore IMDA Model AI Governance Framework for Agentic AI, Version 1.5 [IMDA-MGF], published 20 May 2026, are directly relevant.

First, on agent identity (§ 2.1.2, "Agent identity — Identification"), the framework states:

> An agent should have its own unique, cryptographically verifiable identity, such that it can identify itself to the organisation, its human user, or other agents.

The framework further recommends that agent identities be catalogued and centrally managed, issued from and tracked by a centralised system — the operational model that a DID-based trust registry implements.

Second, on authorization scope (§2.1.2, "Authorisation"), the framework recommends that authorisations be scoped, time- or session-bound, non-transferable, and follow the principle of least privilege by default with explicit escalation paths; that they be bounded by the authorising human's permissions; and that delegations of authority be clearly recorded.

This maps directly to the three AAE blocks: MANDATE defines the scope and action allowlist, CONSTRAINTS implement least-privilege and value bounds, VALIDITY enforces time-bound non-transferable authorization, and the delegation chain structure records authority provenance (Section 3).

A case study in the same framework (§2.3, "Terminal 3 case study") describes an independent implementation of the same pattern: a "Verifiable Credential of Intent" issued by a human principal to an agent before each cycle, defining accessible records, applicable constraints, and a declared ceiling amount. This is a real-world deployment of pre-transaction scoped authorization that is structurally equivalent to AAE.

1.2. Terminology

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

Agent: An autonomous software entity that takes actions on behalf of a principal, potentially without per-action human approval.

Principal: The human or organization that deploys and is ultimately accountable for an agent's actions.

Issuer: The entity that issues the AAE, typically a trust registry or the principal directly.

Relying Party: The system or service that receives a request from an agent and evaluates the AAE to determine whether to process it.

AAE: Agent Authorization Envelope — the structured authorization container defined in this document.

2. The Agent Authorization Envelope

An AAE is a W3C Verifiable Credential [W3C-VC] whose credentialSubject carries an aae object. The aae object has three mandatory members — mandate, constraints, and validity — defined in the following subsections.

An AAE MUST be secured using JOSE. The Verifiable Credential is the payload of a JSON Web Signature (JWS) [RFC7515] in compact serialization, signed by the issuer using the Edwards-curve Digital Signature Algorithm (EdDSA) with the Ed25519 curve, as specified for JOSE in [RFC8037] and for the signature algorithm itself in [RFC8032]. The issuer's signing key is a JSON Web Key (JWK) [RFC7517] bound to the issuer's W3C DID [W3C-DID]: the JWS protected-header kid parameter MUST be a DID URL that dereferences, via the issuer DID document, to the corresponding verification method.

This document distinguishes two media types. An _unsecured AAE_ — the Verifiable Credential before JWS encapsulation — is a JSON object with media type application/aae+json (Section 8). A _secured AAE_ — the JWS in compact serialization — is not a JSON document; it is transported with media type application/jose [RFC7515], and the media type of its JWS payload is indicated by the cty protected-header parameter, "aae+json".

2.1. Structure

On the wire, a secured AAE is a JWS in compact serialization:

BASE64URL(UTF8(JWS Protected Header)) || '.' ||
BASE64URL(JWS Payload) || '.' ||
BASE64URL(JWS Signature)

The JWS Protected Header MUST contain:

```
{
  "alg": "EdDSA",
  "cty": "aae+json",
  "kid": "did:moltrust:registry#key-1"
}
```

alg: REQUIRED. MUST be "EdDSA"; the curve MUST be Ed25519, as specified for JOSE in [RFC8037].

cty: REQUIRED. The media type of the JWS payload; MUST be "aae+json" (Section 8). The "application/" prefix is omitted as permitted by [RFC7515]. Per [RFC7515], cty (content type) identifies the secured payload; the typ parameter, if present, identifies the JWS object itself and MUST NOT carry the payload media type.

kid: REQUIRED. A DID URL identifying the issuer verification method. It MUST dereference, via the issuer DID document, to an Ed25519 verification method authorized for the assertionMethod proof purpose.

The JWS Payload is the unsecured AAE: a W3C Verifiable Credential whose media type is application/aae+json (Section 8).

```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://moltrust.ch/contexts/aae/v1"
  ],
  "type": ["VerifiableCredential", "AgentAuthorizationEnvelope"],
  "id": "urn:uuid:3f2b8c10-7c2e-4f1a-9b6d-1e2a3c4d5e6f",
  "issuer": "did:moltrust:registry",
  "validFrom": "2026-05-20T10:00:00Z",
  "credentialSubject": {
    "id": "did:example:agent-abc123",
    "aae": {
      "mandate": { ... },
      "constraints": { ... },
      "validity": { ... }
    }
  }
}
```

The Verifiable Credential MUST contain an id member whose value is a URI [RFC3986] that is globally unique across all AAEs issued by the issuer (for example, a UUID URN [RFC9562]). The Verifiable Credential MUST NOT contain an embedded proof member; integrity and authenticity are provided solely by the enclosing JWS.

2.2. The MANDATE Block

The MANDATE block specifies what the agent is authorized to do. It MUST contain an actions array of permitted action identifiers and SHOULD contain a purpose string describing the authorization context.

```
"mandate": {
  "actions": ["read", "book", "pay"],
  "purpose": "Travel booking on behalf of principal",
  "scope": "travel-vertical",
  "principal_did": "did:example:principal-xyz"
}
```

actions: REQUIRED. Array of strings. Each string is a permitted action identifier. Relying parties define their own action vocabularies; interoperability is achieved through shared vertical schemas (see Section 4).

purpose: RECOMMENDED. Human-readable description of the authorization context. Used for audit logs.

scope: OPTIONAL. Restricts the MANDATE to a specific vertical or service domain.

principal_did: RECOMMENDED. The DID of the human or organization ultimately accountable for the agent’s actions.

The MANDATE block MAY also contain a delegation object or a delegation_policy object; both are defined in Section 3.

2.3. The CONSTRAINTS Block

The CONSTRAINTS block specifies limits that bound the agent’s actions within the MANDATE.

Relying parties MUST enforce all constraints they recognize. A relying party MUST reject an AAE if any constraint marked required: true is unrecognized or cannot be evaluated. A relying party MAY ignore an unrecognized constraint only if that constraint is explicitly marked required: false. If the required member is absent, the constraint MUST be treated as required: true.

```
"constraints": {
  "max_transaction_value": {
    "value": 500,
    "currency": "USD",
    "required": true
  },
  "allowed_domains": {
    "value": ["booking.example.com", "flights.example.com"],
    "required": true
  },
  "rate_limit": {
    "value": 10,
    "window": "PT1H",
    "required": false
  }
}
```

Constraint keys are extensible. This document defines three RECOMMENDED constraint types:

* max_transaction_value: Maximum value of any single transaction. MUST include value (number) and currency (ISO 4217).

* allowed_domains: Allowlist of domains the agent may contact.

* rate_limit: Maximum number of actions per time window. MUST include value (integer) and window (an ISO 8601 duration, e.g., "PT1H").

A relying party that enforces a rate_limit constraint marked required: true MUST maintain sufficient state to count accepted actions within the specified window. A relying party that cannot maintain such state MUST reject an AAE containing a required rate_limit constraint.

2.4. The VALIDITY Block

The VALIDITY block specifies the temporal bounds of the authorization.

```
"validity": {
  "not_before": "2026-05-20T10:00:00Z",
  "not_after": "2026-05-20T18:00:00Z",
  "revocation_check": "https://api.moltrust.ch/aae/revocation/{id}",
  "single_use": false
}
```

not_before: REQUIRED. An RFC 3339 [RFC3339] date-time, expressed in UTC with the "Z" offset. The AAE MUST NOT be accepted before this time.

not_after: REQUIRED. An RFC 3339 [RFC3339] date-time, expressed in UTC with the "Z" offset. The AAE MUST NOT be accepted after this time. Relying parties MUST reject expired AAEs.

revocation_check: OPTIONAL. An HTTPS URI Template [RFC6570] for checking AAE revocation status. The template MUST support the {id} variable, expanding (with URI encoding) to the Verifiable Credential id, and SHOULD support the {did} variable, expanding to credentialSubject.id. If revocation_check is present, the relying party MUST query the endpoint over HTTPS; the response MUST be a JSON object containing at least an id member and a boolean revoked member. The relying party MUST reject the AAE if the endpoint indicates revoked: true or if the response cannot be parsed. If revocation status cannot be determined — for example, on network failure or an HTTP 5xx response — the relying party MUST reject the AAE. A relying party MAY apply an explicit, locally configured, auditable fail-open policy only for AAEs whose risk classification permits such behaviour; such a policy SHOULD NOT be used for high-risk actions and MUST be subject to explicit governance and audit logging.

single_use: OPTIONAL. Boolean. Default: false. If true, the relying party MUST maintain state keyed by the Verifiable Credential id (Section 2.1) and MUST reject any subsequent presentation of an AAE bearing the same id after the first successful authorization. A relying party that cannot maintain such state MUST reject any AAE with single_use: true. Where a relying party is deployed across multiple nodes, the single-use state MUST be shared across all nodes that can accept the AAE; otherwise the AAE could be replayed against a different node. In this document, "invalidation" denotes relying-party-local state; it does not invalidate the AAE globally.

If the Verifiable Credential contains a validFrom member, the relying party MUST NOT accept the AAE before the later of validFrom and validity.not_before. Issuers SHOULD set validFrom equal to or earlier than validity.not_before.

3. Delegation Chains

An agent may act under a mandate delegated from another agent. An AAE issued directly by a principal, and not itself delegated, is a _root AAE_: it has no delegation member, and its effective delegation depth is 0. A root AAE that authorizes onward delegation MUST include a delegation_policy object in its MANDATE block with a non-negative integer max_depth member; this value is the parent’s

effective maximum depth for the first delegation link.

```
"mandate": {
  "actions": ["read", "book"],
  "delegation_policy": { "max_depth": 2 }
}
```

A _delegated AAE_ carries a delegation object in its MANDATE block:

```
"mandate": {
  "actions": ["read"],
  "delegation": {
    "delegator_did": "did:example:parent-agent",
    "delegator_aae_id": "urn:uuid:parent-aae-123",
    "delegator_aae_uri": "https://aae.example/p/parent-aae-123",
    "delegator_aae_hash": "sha-256:5b7e2c...",
    "depth": 1,
    "max_depth": 2
  }
}
```

delegator_did: REQUIRED in delegation context. The DID of the delegating agent.

delegator_aae_id: REQUIRED in delegation context. The id of the parent AAE.

delegator_aae_uri: REQUIRED in delegation context, unless the parent AAE is embedded in the request by the transport binding. A URI from which the relying party can retrieve the parent AAE.

delegator_aae_hash: OPTIONAL. A hash of the parent secured AAE. If present, the value MUST have the form sha-256:<base64url-encoded-digest>. The digest input MUST be the exact ASCII octet sequence of the parent AAE JWS compact serialization as retrieved, without additional whitespace, decoding, re-encoding, or JSON canonicalization. SHA-256 is as defined in [RFC6234]. If the computed digest does not match the value in delegator_aae_hash, the relying party MUST reject the delegated AAE.

depth: REQUIRED in delegation context. Integer. The delegation depth of this AAE. It MUST equal the parent AAE's effective delegation depth plus 1. The effective delegation depth of a root AAE is 0; the effective delegation depth of a delegated AAE is its delegation.depth.

max_depth: REQUIRED in delegation context. Integer. The maximum delegation depth permitted for this branch of the chain. It MUST be less than or equal to the parent AAE's effective maximum depth — the parent's delegation.max_depth if the parent is a delegated AAE, or the root AAE's mandate.delegation_policy.max_depth if the parent is a root AAE. A relying party MUST reject a delegated AAE whose depth exceeds its max_depth, and MUST reject any delegation whose parent is a root AAE that has no delegation_policy.

A relying party MUST be able to retrieve the parent AAE in order to verify a delegation chain. The parent AAE's credentialSubject.id MUST equal the delegated AAE's delegation.delegator_did. Which DID is permitted to sign a delegated AAE — the signing-authority rule — is specified in the Verification Algorithm (Section 5).

Delegated AAEs MUST NOT grant actions not present in the parent AAE. Delegated AAEs MUST be strictly subordinate to their parent AAE. For the purposes of this document, "equal to or more restrictive" is defined per element as follows:

* *Actions*: The delegated mandate.actions MUST be a subset of the parent mandate.actions. If the delegated MANDATE contains the delegate action, the parent MANDATE MUST also contain it.

* *Numeric upper-bound constraints* (for example, max_transaction_value): The delegated value MUST be less than or equal to the parent value.

* *Rate-limit constraints* (rate_limit): A delegated rate_limit constraint is equal to or more restrictive than the parent only if the relying party can prove that no execution pattern permitted by the delegated rate limit would violate the parent rate limit. In the absence of a profile defining such comparison semantics, the delegated rate_limit MUST use the same window value as the parent constraint, and the delegated value MUST be less than or equal to the parent value; if the windows differ and no such profile applies, the delegated AAE MUST be rejected.

* *Allowlist constraints* (for example, allowed_domains): The delegated value MUST be a subset of the parent value.

* *Validity*: The delegated validity.not_before MUST be greater than or equal to the parent validity.not_before, and the delegated validity.not_after MUST be less than or equal to the parent validity.not_after.

* *Delegation depth*: The delegated delegation.depth MUST equal the parent AAE's effective delegation depth plus 1; the delegated delegation.max_depth MUST be less than or equal to the parent AAE's effective maximum depth; and delegation.depth MUST NOT exceed delegation.max_depth.

Every constraint present in the parent AAE that is marked required: true, or for which the required member is absent and is therefore treated as required: true (Section 2.3), MUST also be present in the delegated AAE and MUST either be marked required: true or omit the required member. A delegated AAE MUST NOT omit, downgrade, or change to required: false any parent constraint that is required by this rule. A delegated AAE MAY introduce additional constraints, provided they do not contradict the parent constraints. For currency-valued constraints (for example, max_transaction_value), the delegated constraint MUST use the same currency as the parent constraint, unless the relying party has an explicitly configured and auditable currency-conversion policy; if the currencies differ and no such policy exists, the delegated AAE MUST be rejected.

If a relying party cannot determine whether a delegated element is equal to or more restrictive than the corresponding parent element, the delegated AAE MUST be rejected.

4. Action Vocabulary Schemas

Interoperability across relying parties requires shared action vocabularies. This document defines a minimal common vocabulary:

Action	Semantics
read	Retrieve information without side effects
write	Create or modify state
delete	Remove state
pay	Initiate a payment or financial transfer
invoke	Call an external API or service
delegate	Issue a delegated AAE to a sub-agent

Table 1

Vertical-specific vocabularies (travel, finance, healthcare) SHOULD be published as extensions to this base vocabulary, as stable and versioned schemas at publicly accessible URIs. This document does not create an IANA registry for action vocabularies.

5. Verification Algorithm

A relying party receiving an agent request with an attached AAE MUST perform the following checks in order:

1. *Signature verification*: Parse the JWS in compact serialization and read the protected header. The _signing DID_ is the DID portion of the kid parameter. Resolve the signing DID, dereference the referenced verification method, and confirm that (a) the verification method is present in that DID document, (b) it is authorized for the assertionMethod proof purpose, (c) it contains or resolves to a JWK with kty: "OKP" and crv: "Ed25519", and (d) the JWS signature validates under that key. Reject the AAE if the signing DID cannot be resolved, the verification method is absent or not authorized for assertionMethod, the key is not Ed25519, the alg is not "EdDSA", or the signature is invalid.

   The relying party MUST then verify signing authority. For a non-delegated AAE (no mandate.delegation member), the signing DID MUST be identical to the Verifiable Credential issuer. For a delegated AAE, the signing DID MUST satisfy one of the following: (a) it is identical to mandate.delegation.delegator_did, and the Verifiable Credential issuer is also identical to that DID; or (b) it is explicitly authorized by mandate.delegation.delegator_did to issue delegated AAEs on that delegator’s behalf, where the authorization is represented by a verification method or service entry in the delegator’s DID document. If signing authority cannot be established under the applicable rule — or, for case (b), the relying party does not understand the authorization mechanism — the AAE MUST be rejected.

2. *Payload and schema validation*: Parse the JWS payload as UTF-8-encoded JSON. The payload MUST be a W3C Verifiable Credential containing id, issuer, credentialSubject.id, and credentialSubject.aae. The aae object MUST contain mandate, constraints, and validity. The protected-header cty parameter MUST equal "aae+json". Reject the AAE if any required member is absent, has the wrong JSON type, or otherwise violates the structural requirements of this document.

3. *Temporal validity*: Confirm current time is within not_before and not_after. Reject if outside bounds.

4. *Subject binding*: Confirm the agent presenting the AAE controls credentialSubject.id. The relying party MUST generate a fresh, unpredictable nonce with at least 128 bits of entropy and send it to the agent together with a relying-party audience identifier. The agent MUST return a JWS in compact serialization whose payload is the UTF-8 encoding of a JSON object with exactly these members:

```
{
  "nonce":  "<relying-party nonce>",
  "aud":    "<relying-party audience identifier>",
  "iat":    "<RFC 3339 UTC timestamp>",
  "aae_id": "<Verifiable Credential id>"
}
```

   The challenge-response JWS protected header MUST contain "alg": "EdDSA" and a kid DID URL whose DID portion equals credentialSubject.id. The referenced verification method MUST be authorized under the authentication relationship in the resolved DID document. The relying party MUST reject the AAE unless all of the following hold: (a) the challenge-response signature is valid under a key controlled by credentialSubject.id; (b) the verification method is authorized for authentication; (c) the nonce was generated by this relying party and has not been used before; (d) aud identifies this relying party; (e) aae_id equals the Verifiable Credential id; and (f) iat is within the relying party's accepted clock-skew window.

5. *Single-use check*: If validity.single_use is true, the relying party MUST perform an atomic check-and-record operation keyed by the Verifiable Credential id. An id that has already been recorded MUST be rejected. Where the relying party is deployed across multiple nodes, this state MUST be shared across all nodes; concurrent presentations of the same id MUST NOT both succeed.

6. *Action check*: Confirm the requested action is present in mandate.actions. Reject if absent.

7. *Constraint evaluation*: For each constraint in the CONSTRAINTS block, the relying party MUST enforce every constraint it recognizes. If a recognized constraint marked required: true (or with required absent) cannot be evaluated, the relying party MUST reject the AAE. If a recognized constraint marked required: false cannot be evaluated, the relying party MAY ignore that

constraint. If a constraint is unrecognized, the relying party MUST reject the AAE unless that constraint is explicitly marked required: false. The relying party MUST reject the AAE if any enforced constraint is violated.

8. *Revocation check* (if applicable): If validity.revocation_check is present, the relying party MUST query the endpoint over HTTPS and evaluate the result as specified for revocation_check in Section 2.4, including the limited fail-open exception defined there.

9. *Delegation chain* (if applicable): If the AAE contains a delegation field, the relying party MUST verify every ancestor AAE in the chain. The signature, payload-and-schema, temporal-validity, and revocation checks above apply to each ancestor AAE. The subject-binding and single-use checks MUST NOT be performed for ancestor AAEs: ancestor agents are not required to be online to answer a challenge, and single-use state applies only to the presented AAE. For each delegation link, the relying party MUST verify that the parent AAE's credentialSubject.id equals the child's delegation.delegator_did; that the signing authority of each AAE in the chain holds as specified in step 1; and that constraint monotonicity and depth limits hold as defined in Section 3. To detect cycles, the relying party MUST maintain the set of AAE id values already visited in the current verification path and MUST reject the chain immediately if any id appears more than once. The relying party MUST enforce an implementation-defined maximum recursion limit no greater than the smallest max_depth value observed in the chain.

Steps 17 are REQUIRED. Steps 89 are conditional on presence of the relevant fields.

6. Security Considerations

6.1. Replay Attacks

AAEs with broad temporal validity windows are susceptible to replay attacks. Implementations SHOULD use short not_after windows (minutes to hours for high-value actions) and SHOULD implement nonce-based replay protection at the transport layer.

6.2. Constraint Bypass

Relying parties that silently ignore unrecognized constraints create a security gap. Relying parties MUST treat unrecognized constraints with required: true as grounds for rejection.

6.3. Key Compromise

If an issuer’s signing key is compromised, all AAEs signed by that key are potentially invalid. Issuers MUST have an operational key-rotation procedure and SHOULD publish revocation endpoints. Issuers SHOULD retain verification material for retired signing keys for at least as long as AAEs signed by those keys can remain valid, unless a key was retired because of compromise. Issuers SHOULD provide a mechanism for real-time revocation signaling to relying parties; continuous access evaluation profiles defined by other standards bodies are one such mechanism.

6.4. Delegation Amplification

Delegation chains that do not enforce constraint monotonicity allow sub-agents to acquire permissions exceeding those of their parent. Implementations MUST enforce that delegated AAEs are strictly subordinate to their parent AAEs in actions, constraints, and validity.

6.5. Delegation Revocation

If a parent AAE or a delegator agent is compromised, the issuer SHOULD treat all downstream delegated AAEs as revoked. A relying party that determines that a parent AAE in a delegation chain has been revoked SHOULD treat all descendant AAEs in that chain as invalid.

6.6. Clock Skew and Time Synchronization

AAE validity depends on relying-party evaluation of not_before, not_after, and challenge-response timestamps. Relying parties SHOULD use authenticated time synchronization and SHOULD define a maximum accepted clock-skew window. For high-value actions, relying parties SHOULD keep the accepted skew to the minimum operationally feasible value. Excessive clock-skew windows can allow premature use of an AAE or its continued use after expiration.

6.7. On-Chain Anchoring

When AAEs are anchored to a public ledger for tamper-evident audit trails, implementations MUST ensure that no personally identifiable information (PII) is written on-chain. AAE content SHOULD be hashed before anchoring; the hash, timestamp, and issuer DID are sufficient for audit purposes.

7. Privacy Considerations

AAEs contain the agent’s DID and may contain the principal’s DID. Relying parties SHOULD NOT log full AAE payloads unless required for regulatory audit purposes. Where audit logs are required, implementations SHOULD apply data minimization: log the AAE identifier, action taken, timestamp, and outcome — not the full MANDATE or CONSTRAINTS payload.

Even when only hashes of AAEs are anchored on a public ledger, timestamps, issuer DIDs, subject DIDs, and repeated hash-publication patterns can leak metadata and enable linkability across an agent’s activity. Implementations SHOULD assess linkability risk before anchoring AAEs on a public ledger and SHOULD avoid publishing stable identifiers on-chain unless required.

8. IANA Considerations

8.1. Media Type Registration

IANA is requested to register the following media type in the "Media Types" registry, following the procedures of [RFC6838] and the structured syntax suffix rules of [RFC6839]:

Type name: application

Subtype name: aae+json

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: Same as for application/json [RFC8259]; UTF-8.

Security considerations: See Section 6 of this document.

Interoperability considerations: This media type uses the "+json" structured syntax suffix [RFC6839]. Processors that do not understand application/aae+json MAY process it as application/json.

Published specification: This document.

Applications that use this media type: Autonomous AI agent frameworks, authorization servers, policy engines, and relying parties that issue or evaluate Agent Authorization Envelopes.

Fragment identifier considerations: As specified for the "+json" structured syntax suffix in [RFC6839].

Additional information: Deprecated alias names for this type: N/A. Magic number(s): N/A. File extension(s): .aae.json. Macintosh file type code(s): N/A.

Person & email address to contact for further information: Lars Kersten Kroehl lars@moltrust.ch (mailto:lars@moltrust.ch)

Intended usage: COMMON

Restrictions on usage: N/A

Author: Lars Kersten Kroehl

Change controller: Lars Kersten Kroehl, CryptoKRI GmbH

This registration applies to the unsecured JSON AAE payload. A secured AAE in JWS compact serialization is not a JSON document and is transported using the application/jose media type defined in [RFC7515]; this document does not register a separate media type for the secured form.

This document requests no other IANA actions.

9. References

9.1. Normative References

[W3C-DID] W3C, "Decentralized Identifiers (DIDs) v1.0", 19 July 2022, <https://www.w3.org/TR/did-core/>.

[W3C-VC] W3C, "Verifiable Credentials Data Model v2.0", 8 May 2024, <https://www.w3.org/TR/vc-data-model-2.0/>.

[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>.

[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>.

[RFC8037] Liusvaara, I., "CFRG Elliptic Curve Diffie-Hellman (ECDH) and Signatures in JSON Object Signing and Encryption (JOSE)", RFC 8037, DOI 10.17487/RFC8037, January 2017, <https://www.rfc-editor.org/info/rfc8037>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/info/rfc7517>.

[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, <https://www.rfc-editor.org/info/rfc3986>.

[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>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <https://www.rfc-editor.org/info/rfc6570>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

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

[RFC6839] Hansen, T. and A. Melnikov, "Additional Media Type Structured Syntax Suffixes", RFC 6839, DOI 10.17487/RFC6839, January 2013, <https://www.rfc-editor.org/info/rfc6839>.

[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>.

9.2. Informative References

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC9635] Richer, J., Ed. and F. Imbault, "Grant Negotiation and Authorization Protocol (GNAP)", RFC 9635, DOI 10.17487/RFC9635, October 2024, <https://www.rfc-editor.org/info/rfc9635>.

[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>.

[SPIFFE] "SPIFFE: Secure Production Identity Framework for Everyone", n.d., <https://spiffe.io/>.

[ARXIV-AAE] Kroehl, L. K., "From Specification to Deployment: Empirical Evidence from a W3C VC + DID Trust Infrastructure for Autonomous Agents", May 2026, <https://arxiv.org/abs/2605.06738>.

[IMDA-MGF] Singapore IMDA, "Model AI Governance Framework for Agentic AI, Version 1.5", 20 May 2026, <https://www.imda.gov.sg/-/media/imda/files/about/emerging-tech-and-research/artificial-intelligence/mgf-for-agentic-ai.pdf>.

[NIST-CAISI] NIST, "NIST AI 100-1: Artificial Intelligence Risk Management Framework", January 2023.

Appendix A. Example: Travel Booking Agent

A travel booking agent operating under an 8-hour mandate with a $500 transaction cap. The example shows the JWS protected header and the JWS payload (the unsecured Verifiable Credential); on the wire these are BASE64URL-encoded and concatenated with the signature as a JWS in compact serialization (Section 2.1).

JWS protected header:

```
{
  "alg": "EdDSA",
  "cty": "aae+json",
  "kid": "did:moltrust:registry#key-1"
}
```

JWS payload:


```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://moltrust.ch/contexts/aae/v1"
  ],
  "type": ["VerifiableCredential", "AgentAuthorizationEnvelope"],
  "id": "urn:uuid:9b1deb4d-3b7d-4bad-9bdd-2b0d7b3dcb6d",
  "issuer": "did:moltrust:registry",
  "validFrom": "2026-05-20T08:00:00Z",
  "credentialSubject": {
    "id": "did:example:travel-agent-001",
    "aae": {
      "mandate": {
        "actions": ["read", "book", "pay"],
        "purpose": "Business travel booking",
        "scope": "travel-vertical",
        "principal_did": "did:example:enterprise-corp"
      },
      "constraints": {
        "max_transaction_value": {
          "value": 500, "currency": "USD", "required": true
        },
        "allowed_domains": {
          "value": ["flights.example.com", "hotels.example.com"],
          "required": true
        }
      },
      "validity": {
        "not_before": "2026-05-20T08:00:00Z",
        "not_after": "2026-05-20T16:00:00Z",
        "single_use": false
      }
    }
  }
}
```


Appendix B. Relationship to Existing Standards

AAE is designed as a complement to, not a replacement for, existing authorization standards:

* *OAuth 2.0 / GNAP* ([RFC6749], [RFC9635]): Handle human-delegated access tokens. AAE handles machine-to-machine agent authorization with richer semantics.

* *SPIFFE/SVID* ([SPIFFE]): Handles workload identity in service meshes. AAE handles authorization semantics layered above identity.

* *W3C Verifiable Credentials*: AAE is issued as a VC, leveraging the existing VC ecosystem for issuance, verification, and revocation.

* *W3C DIDs*: AAE subjects and issuers are identified by DIDs, enabling decentralized, portable agent identity.

Appendix C. Acknowledgements

The AAE specification is derived from the MolTrust production deployment documented in [ARXIV-AAE]. The author thanks Harald Rssler (DSNCON GmbH) for infrastructure and security review.

Author’s Address

Lars Kersten Kroehl
CryptoKRI GmbH
CH- Zurich
Switzerland
Email: lars@moltrust.ch