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

A. MESSOUS
Huawei R&D

16 February 2026

Entity Attestation Token (EAT) Profile for Autonomous AI Agents
draft-messous-eat-ai-00

Abstract

This document defines a profile for the Entity Attestation Token (EAT) to support remote attestation of autonomous AI agents across domains. It specifies a set of standardized claims for attesting the integrity of AI model parameters, the provenance of training data, and the constraints of inference-time data access policies. Optional extensions for 5G/6G network functions— such as slice-type authorization— are included for interoperability with ETSI ENI and 3GPP architectures. The profile is encoded in CBOR Web Tokens (CWTs) or JSON Web Tokens (JWTs) and is designed to be used within the IETF RATS architecture.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-messous-eat-ai/.

Source for this draft and an issue tracker can be found at https://github.com/https://github.com/mmessous/draft-messous-EAT-AI.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 20 August 2026.

Table of Contents

1. 1. Introduction

Autonomous AI agents— software entities that perceive, reason, and act with minimal human oversight— are deployed across cloud, edge, enterprise, and telecommunications environments. Their autonomy introduces new trust challenges: if an agent' s model is tampered, its training data is non-compliant, or its inference policy is violated, the consequences range from service disruption to regulatory breaches.

The Entity Attestation Token (EAT) [RFC9711] provides a standardized framework for remote attestation. However, EAT does not define claims specific to AI artifacts. This document fills that gap by specifying a *generic EAT profile for AI agents*, with *optional telecom-specific claims* for use in 5G/6G networks (e.g., ETSI ENI AI-Core [ETSI-GR-ENI-051], 3GPP TS 29.510).

This profile enables verifiers—such as OAuth resource servers, network function orchestrators, or policy enforcement points—to make trust decisions based on verifiable evidence about an agent's: - *Model integrity* (weights, architecture), - *Training provenance* (dataset, geography, privacy), - *Runtime authorization* (capabilities, allowed APIs, slice types).

This profile does not define a full AI Bill of Materials (AIBOM). Instead, it provides a minimal set of *verifiable claims* sufficient for remote attestation and policy enforcement. It assumes that richer metadata—such as detailed training data lineage, model cards, or complete dependency graphs—is maintained in external documents (e.g., an AIBOM or SBOM), which may be referenced via claims like ai-sbom-ref or a future ai-bom-ref. Traditional SBOMs remain essential to capture the *software supply chain* (e.g., Python, CUDA, framework versions) on which the AI agent depends. This profile complements, but does not replace, those artifacts.

1.1. 2. Terminology

* *AI Agent*: AI agents are autonomous systems powered by Large Language Models (LLMs) that can reason, plan, use tools, maintain memory, and take actions to accomplish goals.

* *Model Integrity*: The property that AI model weights and architecture have not been altered from a known-good state.

* *Training Provenance*: Metadata describing the origin, scope, and privacy properties of data used to train an AI model.

* *Inference Policy*: Constraints defining the authorized input context (e.g., slice type, geography) under which an agent may operate.

* *EAT-AI*: The EAT profile defined in this document.

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].

1.2. 3. Use Cases

1.2.1. 3.1. Generic AI Agent Attestation

An enterprise AI agent attests its model hash and data retention policy before accessing a protected API. For a more extensive protection, attestation target could also include behavioral manifests, identity, prompts, tools and capabilities, SBOM/AIBOMs etc in the future.

1.2.2. 3.2. 5G/6G Network Functions (Optional Context)

In ETSI ENI AI-Core, an Execution Agent generates instructions for network slice configuration. The agent should prove: - It runs an approved model (ai-model-hash), - It was trained on GDPR-compliant data (training-geo-region, dp-epsilon), - It is authorized for specific slice types (allowed-slice-types).

> *Note*: Telecom-specific claims are *optional* and *only meaningful in 3GPP/ETSI contexts*.

1.3. 4. EAT-AI Claims Definition

Claims are defined for both *CWT (CBOR)* and *JWT (JSON)*. In CWT, claims use signed integer keys; in JWT, they use text names (with hyphens converted to underscores per convention).

1.3.1. 4.1. Core Claims (Generic, Domain-Agnostic)

Claim Name	CBOR Key	JWT Name	Type	Description
ai-model-id	-75000	ai_model_id	text	URN-formatted model identifier (e.g., urn:ietf:ai:model:cnn-v3)
ai-model-hash	-75001	ai_model_hash	digest	Cryptographic hash of the serialized model weights and architecture
model-arch-digest	-75002	model_arch_digest	digest	Cryptographic hash of model computational graph
training-	-75003	training_data_id	text	Unique ID of training

data-id				dataset
dp-epsilon	-75005	dp_epsilon	float	Differential privacy epsilon used during training
input-policy-digest	-75006	input_policy_digest	digest	Cryptographic hash of inference input policy
data-retention-policy	-75008	data_retention_policy	text	e.g., "none", "session", "24h"
owner-id	-75009	owner_id	text	Identity of principal (e.g., GPSI per 3GPP TS 29.222)
capabilities	-75010	capabilities	array of text	High-level functions (e.g., "slice-optimization")
allowed-apis	-75011	allowed_apis	array of URI	Specific endpoints the agent may call
ai-sbom-ref	-75012	ai_sbom_ref	text / map	Reference to a Software Bill of Materials (SBOM) describing the AI agent's runtime dependencies (e.g., Python, CUDA, libraries). MAY be a URI, digest, or embedded SBOM fragment

Table 1

1.3.1.1. 4.1.1. ai-model-id

* ai-model-id: A globally unique model identifier encoded as a URN. The URN *namespace* urn:ietf:ai:model: is reserved for standardized reference models (e.g., defined in RFCs). *Model owners SHOULD use their own URN namespace* (e.g., based on domain name, PEN, or UUID) to avoid central coordination. Examples:

- urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6 (for a private model)

- urn:ietf:ai:model:llama3-8b (for a well-known public model, if later standardized)

- urn:dev:example.com:finance-agent-v2 (enterprise-owned model)

1.3.1.2. 4.1.2. use of cryptography digests

* The claims ai-model-hash, model-arch-digest, and input-policy-digest represent cryptographic digests of serialized artifacts (e.g., model weights, computational graphs, or policy documents). To support algorithm agility and avoid ambiguity, each such claim is defined as a digest structure rather than a bare byte string. A digest structure is encoded as a two-element array:

cbor [alg, hash] where: * *alg* is the Hash Algorithm Identifier, using either the *integer* or *text string* from the IANA COSE Algorithms registry (https://www.iana.org/assignments/cose/cose.xhtml#algorithms), indicating the hash function used (e.g., ’-16’ for SHA-256, -44 for SHA-384, -45 for SHA3-256). * *hash* is the byte string output of applying that hash function to the canonical serialization of the artifact.

In *CBOR*, the digest is represented as a CBOR array: [int / tstr, bstr]. In *JWT* (JSON), it is represented as a JSON object: { "alg": "...", "hash": "base64url-encoded-hash" }. This design aligns with the Detached-Submodule-Digest type defined in [RFC 9711, Section 4.2.18.2] and enables future-proof support for multiple hash algorithms (e.g., SHA-2, SHA-3, post-quantum secure hashes) without requiring new claims or breaking existing parsers.

1.3.1.3. 4.1.3. ai-sbom-ref

* The ai-sbom-ref claim provides a reference to the *Software Bill of Materials (SBOM)* associated with the AI agent. This enables verifiers to assess the integrity, license compliance, and vulnerability status of the agent’s software supply chain. The value MAY be:

* A URI pointing to an SBOM document (e.g., in SPDX or CycloneDX format),

* A digest (using the structured digest format defined in Section 4.1) of an SBOM,

* Or a compact embedded representation (e.g., a minimal map of critical components).

Example (CBOR): cbor / ai-sbom-ref / -75012: "https://example.com/sboms/agent-xyz.spdx.json" Example (embedded digest): cbor / ai-sbom-ref / -75012: [-44, h'abcd1234...'] ; SHA-384 digest of SBOM When used, the SBOM SHOULD include: - Runtime environment (e.g., Python 3.11, CUDA 12.4), - AI framework versions (e.g., PyTorch 2.3, TensorFlow 2.15), - Critical dependencies (e.g., NumPy, cuDNN), - Model serialization format (e.g., ONNX v9, SafeTensors v0.4). This claim complements model integrity (ai-model-hash) by attesting to the execution context in which the model operates—critical for reproducibility and security analysis.

1.3.2. 4.2. Optional Domain-Specific Claims (5G/6G)

Claim Name	CBOR Key	JWT Name	Type	Description
training-geo-region	-75004	training_geo_region	array of text	ISO 3166-1 alpha-2 codes (e.g., ["DE", "FR"])
allowed-slice-types	-75007	allowed_slice_types	array of text	3GPP-defined slice types (e.g., "eMBB", "URLLC")

Table 2

Usage: These claims *SHOULD be used* when attesting agents in *ETSI ENI or 3GPP SBA* environments.

1.3.3. 4.3. Composite and Multi-Component Attestation

This profile utilizes the recursive nesting capability of the submods claim (Key 266) to support three specific composite scenarios:

1.3.3.1. 4.3.1. Multi-Agent Platforms:

To support a user managing multiple agents with varying configurations, we should leverage the recursive nesting capability of the submods claim (CBOR key 266) as defined in [RFC 9711]. In this architectural pattern, the top-level EAT represents the user's platform or trust domain. Each agent is a submodule of that platform, and if an agent uses multiple models, those models are further nested as submodules of that specific agent.

The following CWT example shows a platform hosting two agents. Agent 1 is a complex orchestrator using two models, while Agent 2 is a simple worker using only one.

Code snippet ''' { / ueid / 256: h'0102030405060708', / User/Platform ID / / nonce / 10: h'abcdef1234567890', / Freshness Nonce / / submods / 266: { / Submodules Section /

```
/ --- Agent 1: Multi-Model Orchestrator --- /
"agent-1": {
  / swname / 270: "Orchestrator-Agent-v2",
  / submods / 266: {             / Nested Model Submodules /
    "llm-core": {
      / ai-model-id / -75000: ":uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6",
      / ai-model-hash / -75001: [-44, h'9a8b...']  / SHA-384 /
    },
    "tool-planner": {
      / ai-model-id / -75000: ":uuid:550e8400-e29b-41d4-a716-446655440000",
      / ai-model-hash / -75001: [-16, h'5e4f...']  / SHA-256 /
    }
  }
},

/ --- Agent 2: Single-Model Worker --- /
"agent-2": {
  / swname / 270: "Vision-Worker-v1",
  / ai-model-id / -75000: ":ietf:ai:model:vit-b-16", /
  / ai-model-hash / -75001: [-44, h'd3e2...']            /
}   } } '''
```

1.3.3.2. 4.3.2. Multi-Model Agents:

A single agent utilizing an orchestrator model and task-specific worker models.

Modern AI agents are not necessarly monolithic; sophesticated Agents can consist of an orchestrator model (e.g., a LLM) and several task-specific worker models (e.g., image classifiers or encoders). To

support these configurations, this profile utilizes the submods claim (Key 266) from [RFC 9711]. Each distinct model used by the agent SHOULD be represented as an entry within the submods map. This allows for granular policy appraisal where different models may have different trust levels, privacy parameters (dp_epsilon), or residency requirements.

When a model is represented in a submodule, it carries its own instance of ai-model-id and ai-model-hash. If the model weights are proprietary (e.g., accessed via a cloud API), the submodule SHOULD include an ai-model-id that the Verifier can match against a provider Endorsement.

The following example demonstrates an agent employing an orchestrator LLM and a specialized vision model. Note the use of the digest format [alg, val] to support different hash types for each model.

Code snippet { / ueid / 256: h'0102030405060708', / nonce / 10: h'abcdef1234567890', / submods / 266: { "orchestrator-llm": { / ai-model-id / -75000: ":uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6", / ai-model-hash / -75001: [-44, h'9a8b7c6d...'] / SHA-384 / }, "vision-classifier": { / ai-model-id / -75000: ":ietf:ai:model:vit-b-16", / ai-model-hash / -75001: [-16, h'5e4f3a2b...'], / SHA-256 / / dp-epsilon / -75005: 0.8 } } }

1.3.3.3. 4.3.3. Layered Trust:

Scenarios where different system components (e.g., hardware TEE, OS runtime, and AI model) are owned or managed by different entities.

* *Nesting Mechanism:* Components SHOULD be represented as nested EATs within the submods claim (Key 266). Each nested token MAY be signed by a different attestation key belonging to the respective component owner.

* *Verifier Role:* A Verifier receiving a composite EAT SHOULD follow the Hierarchical Pattern, where it acts as a Lead Verifier and delegates the appraisal of individual submodules to specialized verifiers that hold the appropriate Trust Anchors for each owner.

For multi-owner attestation, a *Lead Verifier* SHOULD follow the *Hierarchical Pattern*, extracting nested sub-tokens and delegating their appraisal to specialized verifiers holding the appropriate Trust Anchors.

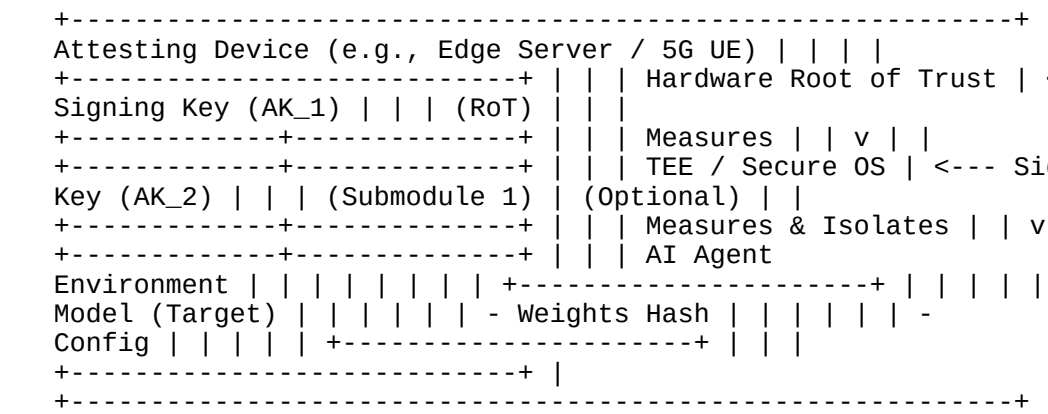

_Figure 1: Example of a Chain of Trust _

Figure 1 illustrates the Chain of Trust. The Hardware Root of Trust (RoT) measures the integrity of the Trusted Execution Environment (TEE) or OS. The TEE, acting as a transitive verifier, subsequently measures the AI Agent’s model binaries and policy configurations. The resulting EAT token reflects this hierarchy using nested submodules, ensuring that the ai-model-hash is reported by a trusted parent rather than the agent itself.

To clarify the hierarchical trust relationships in multi-owner attestation scenarios, Figure 2 illustrates the binding of components across hardware, runtime, and AI model layers. Each layer is attested by a distinct owner and represented as a nested submodule within the top-level EAT per RFC 9711 Section 4.2.18.

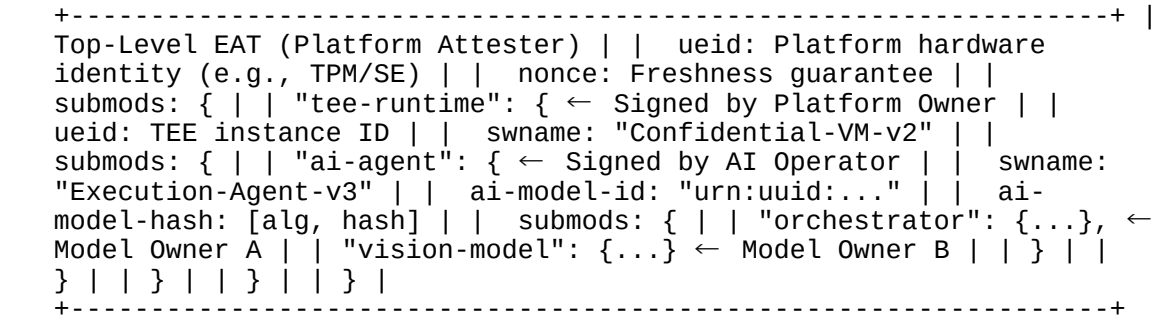

Figure 2: Trust hierarchy for layered attestation using EAT submods

1.3.3.4. *Trust Binding Semantics*

* *Hardware → TEE:* The TEE runtime measurement is included in a platform-signed attestation report (e.g., AMD SEV-SNP, Intel TDX). The top-level EAT's signature binds the tee-runtime submodule to this hardware root.

* *TEE → AI Agent:* The AI agent's code and configuration are measured into the TEE's launch digest. The ai-agent submodule is signed by the AI operator's key, which itself is endorsed by the platform owner (via an Endorsement per RFC 9334).

* *Agent → Models:* Individual models are signed by their respective providers. The agent's runtime verifies model signatures before loading; these signatures are reflected in the nested submods entries.

1.3.3.5. *Appraisal Delegation*

Per RFC 9334 Section 5.3, a Lead Verifier appraising the top-level token: 1- Validates the platform signature against a hardware Trust Anchor 2- Delegates tee-runtime appraisal to a TEE-specific verifier holding platform Endorsements 3- Delegates ai-agent appraisal to an AI policy verifier holding operator Trust Anchors 4- Optionally delegates model submodules to specialized model catalog verifiers

A submodule appraisal failure MUST cause rejection of the entire attestation unless policy explicitly permits partial trust (e.g., non-critical auxiliary models). This failure semantics MUST be defined by the deployment policy—not by this profile.

1.4. 5. Security Considerations

* Claims SHOULD be bound to a hardware-rooted attestation where available.

* *ai-model-hash* SHOULD be computed on the serialized model file (e.g., ONNX, PyTorch), not in-memory tensors.

* *Verifiers* SHOULD validate claims against authoritative registries (e.g., model hash in secure model catalog).

* *_Replay attacks_* SHOULD be mitigated using EAT nonce (CWT key 10) or exp (key 4).

* Verifiers SHOULD validate the referenced SBOM against known vulnerability databases (e.g., NVD) and reject agents using components with unpatched critical flaws.

* Verifiers SHOULD validate that ai-model-id values originate from trusted namespaces (e.g., known domains, approved PENs, or allow-listed UUIDs). Dynamic model deployment does not require central registration, but policy enforcement may restrict acceptable namespaces.

* Verifiers are expected to combine EAT-AI evidence with external SBOM/AIBOM analysis for comprehensive risk assessment.

1.5. 6. Privacy Considerations

* training-geo-region reveals data origin and SHOULD be minimized.

* EAT tokens SHOULD be transmitted over secure channels (e.g., TLS 1.3).

* owner-id SHOULD use pseudonymous identifiers (e.g., GPSI per 3GPP TS 29.222).

* Embedded SBOMs or detailed URIs may reveal deployment topology. When privacy is a concern, use opaque digests or pseudonymized SBOM identifiers.

* High-granularity combinations of training-geo-region + dp-epsilon + allowed-apis may uniquely identify a "private" model even if the ai-model-id is obscured.

1.6. 7. IANA Considerations

7.1. EAT Profile Registration - IANA is requested to register in the "Entity Attestation Token (EAT) Profiles" registry:

Profile Name: Autonomous AI Agent EAT Profile Reference: [THIS DOCUMENT]

1.6.1. 7.2. CWT Claims Registry

IANA is requested to register the following in the "CBOR Web Token (CWT) Claims" registry [IANA-CWT]:

Value	Claim Name	Description
-75000	ai-model-id	AI model URN
-75001	ai-model-hash	Model weights hash
-75002	model-arch-digest	Model graph hash
-75003	training-data-id	Training dataset ID
-75004	training-geo-region	Training data regions
-75005	dp-epsilon	DP epsilon
-75006	input-policy-digest	Inference policy hash
-75007	allowed-slice-types	Authorized slice types
-75008	data-retention-policy	Data retention policy
-75009	owner-id	Resource owner identifier
-75010	capabilities	Agent capabilities
-75011	allowed-apis	Allowed API endpoints
-75012	ai-sbom-ref	Reference to AI agent's Software Bill of Materials (SBOM)

Table 3

The range -75000 to -75012 is reserved for this profile.

1.6.2. 7.3. JWT Claims Registry

IANA is requested to register the corresponding JWT claim names in the "JSON Web Token Claims" registry [IANA-JWT].

1.7. 8. References

8.1. Normative References

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

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

* [RFC8174 (https://www.ietf.org/rfc/rfc8174.html)] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017.

* [RFC8392 (https://datatracker.ietf.org/doc/html/rfc8392)] Jones, M., et al., "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018.

* [RFC9711 (https://datatracker.ietf.org/doc/html/rfc9711)] L. Lundblade, G. Mandyam,J. O'Donoghue,C. Wallace, "The Entity Attestation Token (EAT)", RFC 9711, DOI 10.17487/RFC9711, April 2025.

* [RFC9334 (https://datatracker.ietf.org/doc/html/rfc9334)] Birkett, M., et al., "Remote ATtestation ProcedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023.

* [RFC8126 (https://datatracker.ietf.org/doc/html/rfc8126)] Cotton, M., et al., "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 8126, DOI 10.17487/RFC8126, June 2017.

* [[EAT Measured Component] (https://datatracker.ietf.org/doc/draft-ietf-rats-eat-measured-component/)] Frost S., et al., "EAT Measured Component", Active Internet-Draft (rats WG).

1.7.1. 8.2. Informative References

* [ETSI-GR-ENI-051 (https://www.etsi.org/deliver/etsi_gr/ENI/001_099/051/04.01.01_60/gr_ENI051v040101p.pdf)] ETSI, *"Architectural Framework for ENI in 6G"*, GR ENI 051 V4.1.1, February 2025.

* [3GPP-TR-33.898 (https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=4088)] 3GPP, *"Study on security and privacy of Artificial Intelligence/Machine Learning (AI/ML)-based services and applications in 5G"*, TR 33.898, V18.0.1 July 2023.

* [3GPP-TR-33.784 (https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=4294)] 3GPP, *"Study on security aspects of core network enhanced support for Artificial Intelligence/Machine Learning (AI/ML)"*, TR 33.784 V0.0.0, April 2025.

* [I-D.huang-rats-agentic-eat-cap-attest (https://datatracker.ietf.org/doc/draft-huang-rats-agentic-eat-cap-attest/)] Huang, K., et al., *"Capability Attestation Extensions for the Entity Attestation Token (EAT) in Agentic AI Systems"*, Work in Progress, Internet-Draft, March 2025.

* [draft-ni-wimse-ai-agent-identity (https://datatracker.ietf.org/doc/draft-ni-wimse-ai-agent-identity/)] Yuan, N., Liu, P., *"WIMSE Applicability for AI Agents"*, Work in Progress.

* [draft-liu-oauth-a2a-profile (https://datatracker.ietf.org/doc/draft-liu-oauth-a2a-profile/)] Liu, P., Yuan, N., *"Agent-to-Agent (A2A) Profile for OAuth Transaction Tokens"*, Work in Progress.

1.8. Appendix A. Example EAT-AI Token (CWT)

The following is a CBOR diagnostic notation of an EAT-AI token:

{ / ueid / 256: h’0102030405060708’, / swname / 270: "execution-agent-v3", / ai-model-id / -75000: "urn:etsi:eni:model:slice-opt-cnn:v3", / ai-model-hash / -75001: [-44,h’9a8b7c6d5e4f3a2b1c0d9e8f7a6b5c4d3e2f1a0b9c8d7e6f5a4b3c2d1e0f’], / training-geo-region / -75004: ["DE", "FR"], / dp-epsilon / -75005: 0.5, / input-policy-digest / -75006: [-44,h’a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0’], / ai-sbom-ref / -75012: "https://sbom.example.net/agents/slice-opt-v3.spdx.json", / nonce / 10: h’abcdef1234567890’ }

1.9. Appendix B. Relationship to Existing Standards & Initiatives

This document complements: - IETF RATS (https://datatracker.ietf.org/doc/html/rfc9334): Provides the architectural context for EAT. - ETSI GR ENI 051 (https://www.etsi.org/deliver/etsi_gr/ENI/001_099/051/04.01.01_60/gr_ENI051v040101p.pdf): Defines the AI-Core where these claims are applied.

It differs from I-D.huang-rats-agentic-eat-cap-attest (https://datatracker.ietf.org/doc/draft-huang-rats-agentic-eat-cap-attest/) by specifying measurable, cryptographically verifiable claims rather than abstract capabilities.

Acknowledgments

TODO acknowledge.

Author’s Address

Ayoub MESSOUS
Huawei R&D
Email: ayoub.messous@huaweil.com