

Remote ATtestation Procedures  
Internet-Draft  
Intended status: Standards Track  
Expires: 25 January 2026

T. Fossati  
Linaro  
E. Voit  
Cisco  
S. Trofimov  
Arm Limited  
H. Birkholz  
Fraunhofer SIT  
24 July 2025

EAT Attestation Results  
draft-ietf-rats-ear-01

## Abstract

This document defines the EAT Attestation Result (EAR) message format.

EAR is used by a verifier to encode the result of the appraisal over an attester's evidence. It embeds an AR4SI's "trustworthiness vector" to present a normalized view of the evaluation results, thus easing the task of defining and computing authorization policies by relying parties. Alongside the trustworthiness vector, EAR provides contextual information bound to the appraisal process. This allows a relying party (or an auditor) to reconstruct the frame of reference in which the trustworthiness vector was originally computed. EAR supports simple devices with one attester as well as composite devices that are made of multiple attesters, allowing the state of each attester to be separately examined. EAR can also accommodate registered and unregistered extensions. It can be serialized and protected using either CWT or JWT.

## About This Document

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

The latest revision of this draft can be found at <https://thomas-fossati.github.io/draft-ear/draft-fv-rats-ear.html>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-ietf-rats-ear/>.

Discussion of this document takes place on the Remote ATtestation Procedures Working Group mailing list (<mailto:rats@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/rats/>. Subscribe at <https://www.ietf.org/mailman/listinfo/rats/>.

Source for this draft and an issue tracker can be found at <https://github.com/thomas-fossati/draft-ear>.

## 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 25 January 2026.

## Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Conventions and Definitions . . . . .	4
3. EAT Attestation Result . . . . .	4
3.1. EAR Appraisal Claims . . . . .	5
3.2. JSON Serialisation . . . . .	6
3.2.1. Examples . . . . .	7
3.3. CBOR Serialisation . . . . .	9
3.3.1. Examples . . . . .	9
4. EAR Extensions . . . . .	9
4.1. Unregistered claims . . . . .	10
4.2. Registered claims . . . . .	10
4.3. Choosing between registered and unregistered claims . . .	10

4.4.	TEEP Extension	11
4.4.1.	JSON Serialization Example	11
4.4.2.	CBOR Serialization Example	12
4.5.	Project Veraison Extensions	13
4.5.1.	JSON Serialization Examples	14
4.5.2.	CBOR Serialization Example	16
5.	Media Types	17
6.	Implementation Status	18
6.1.	Project Veraison	18
6.1.1.	github.com/veraison/ear	18
6.1.2.	github.com/veraison/c-ear	19
6.1.3.	github.com/veraison/rust-ear	19
7.	Security Considerations	19
8.	Privacy Considerations	19
9.	IANA Considerations	20
9.1.	New EAT Claims	20
9.1.1.	EAR Status	20
9.1.2.	Trustworthiness Vector	20
9.1.3.	EAR Raw Evidence	21
9.1.4.	EAR Appraisal Policy Identifier	21
9.1.5.	Verifier Software Identifier	21
9.1.6.	EAR TEEP Claims	22
10.	References	22
10.1.	Normative References	22
10.2.	Informative References	23
Appendix A.	Common CDDL Types	24
Appendix B.	Open Policy Agent Example	24
Appendix C.	Open Issues	26
Appendix D.	Document History	26
D.1.	draft-fv-rats-ear-00	26
D.2.	draft-fv-rats-ear-01	27
D.3.	draft-fv-rats-ear-02	27
	Acknowledgments	27
	Authors' Addresses	27

## 1. Introduction

This document defines the EAT [I-D.ietf-rats-eat] Attestation Result (EAR) message format.

EAR is used by a verifier to encode the result of the appraisal over an attester's evidence. It embeds an AR4SI's "trustworthiness vector" [I-D.ietf-rats-ar4si] to present a normalized view of the evaluation results, thus easing the task of defining and computing authorization policies by relying parties. Alongside the trustworthiness vector, EAR provides contextual information bound to the appraisal process. This allows a relying party (or an auditor) to reconstruct the frame of reference in which the trustworthiness

vector was originally computed. EAR supports simple devices with one attester as well as composite devices that are made of multiple attesters (see Section 3.3 of [RFC9334]) allowing the state of each attester to be separately examined. EAR can also accommodate registered and unregistered extensions. It can be serialized and protected using either CWT [RFC8392] or JWT [RFC7519].

## 2. Conventions and Definitions

This document uses terms and concepts defined by the RATS architecture. For a complete glossary see Section 4 of [RFC9334].

The terminology from CBOR [STD94], CDDL [RFC8610] and COSE [STD96] applies; in particular, CBOR diagnostic notation is defined in Section 8 of [STD94] and Appendix G of [RFC8610].

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

## 3. EAT Attestation Result

EAR is an EAT token which can be serialized as JWT [RFC7519] or CWT [RFC8392].

The EAR claims-set is as follows:

```

;# import rfc9711 as eat
;# import ar4si as ar4si

EAR = {
  eat.profile-label => "tag:ietf.org,2025-07:ear"
  eat.iat-claim-label => int
  verifier-id-label => ar4si.verifier-id
  ? raw-evidence-label => eat.binary-data
  eat.submods-label => { + text => EAR-appraisal }
  ? eat.nonce-label => eat.nonce-type
  * $$ear-extension
}

; EAR-specific claims
raw-evidence-label = eat.JC<"ear.raw-evidence", 1002>
verifier-id-label = eat.JC<"ear.verifier-id", 1004>
```

Figure 1: EAR (CDDL Definition)

Where:

`eat_profile` (mandatory) The EAT profile (Section 6 of [I-D.ietf-rats-eat]) associated with the EAR claims-set and encodings defined by this document. It MUST be the following tag URI ([RFC4151]) `tag:ietf.org,2025-07:ear`.

`iat` (mandatory) "Issued At" claim -- the time at which the EAR is issued. See Section 4.1.6 of [RFC7519] and Section 4.3.1 of [I-D.ietf-rats-eat] for the EAT-specific encoding restrictions (i.e., disallowing the floating point representation).

`ear.verifier-id` (mandatory) Identifying information about the appraising verifier. See Section 3.3 of [I-D.ietf-rats-ar4si] for further details on its structure and serialization.

`ear.raw-evidence` (optional) The unabridged evidence submitted for appraisal, including any signed container/envelope. This field may be consumed by other Verifiers in multi-stage verification scenarios or by auditors. There are privacy considerations associated with this claim. See Section 8.

`submods` (mandatory) A submodule map (Section 4.2.18 of [I-D.ietf-rats-eat]) holding one EAR-appraisal for each separately appraised attester. The map MUST contain at least one entry. For each appraised attester the verifier chooses a unique label. For example, when evidence is in EAT format, the label could be constructed from the associated EAT profile. A verifier SHOULD publicly and permanently document its labelling scheme for each supported evidence type, unless EAR payloads are produced and consumed entirely within a private deployment. See Section 3.1 for the details about the contents of an EAR-appraisal.

`eat_nonce` (optional) A user supplied nonce that is echoed by the verifier to provide freshness. The nonce is a sequence of bytes between 8 and 64 bytes long. When serialized as JWT, the nonce MUST be base64 encoded, resulting in a string between 12 and 88 bytes long. See Section 4.1 of [I-D.ietf-rats-eat].

`$$ear-extension` (optional) Any registered or unregistered extension. An EAR extension MUST be a map. See Section 4 for further details.

### 3.1. EAR Appraisal Claims

```
import ar4si as ar4si

EAR-appraisal = {
  status-label => ar4si.trustworthiness-tier
  ? trustworthiness-vector-label => ar4si.trustworthiness-vector
  ? appraisal-policy-id-label => text
  * $$ear-appraisal-extension
}

status-label = eat.JC<"ear.status", 1000>
trustworthiness-vector-label = eat.JC<"ear.trustworthiness-vector", 1001>
appraisal-policy-id-label = eat.JC<"ear.appraisal-policy-id", 1003>
```

Figure 2: EAR Appraisal Claims (CDDL Definition)

**ear.status (mandatory)**

The overall appraisal status for this attester represented as one of the four trustworthiness tiers (Section 3.2 of [I-D.ietf-rats-ar4si]). The value of this claim MUST be set to a tier of no higher trust than the tier corresponding to the worst trustworthiness claim across the entire trustworthiness vector.

**ear.trustworthiness-vector (optional)**

The AR4SI trustworthiness vector providing the breakdown of the appraisal for this attester. See Section 3.1 of [I-D.ietf-rats-ar4si] for the details. This claim MUST be present unless the party requesting Evidence appraisal explicitly asks for it to be dropped, e.g., via an API parameter or similar arrangement. Such consumer would therefore rely entirely on the semantics of the ear.status claim. This behaviour is NOT RECOMMENDED because of the resulting loss of quality of the appraisal result.

**ear.appraisal-policy-id (optional)**

An unique identifier of the appraisal policy used to evaluate the attestation result.

**\$\$ear-appraisal-extension (optional)**

Any registered or unregistered extension. An EAR appraisal extension MUST be a map. See Section 4 for further details.

### 3.2. JSON Serialisation

### 3.2.1. Examples

The example in Figure 3 shows an EAR claims-set corresponding to a "contraindicated" appraisal, meaning the verifier has found some problems with the attester's state reported in the submitted evidence. Specifically, the identified issue is related to unauthorized code or configuration loaded in runtime memory (i.e., value 96 in the executables category). The appraisal is for a device with one attester labelled "PSA". Note that in case there is only one attester, the labelling can be freely chosen because there is no ambiguity.

```
{
  "eat_profile": "tag:ietf.org,2025-07:ear",
  "iat": 1666529184,
  "ear.verifier-id": {
    "developer": "https://veraison-project.org",
    "build": "vts 0.0.1"
  },
  "ear.raw-evidence": "NzQ3MjY5NzM2NTYzNzQK",
  "submods": {
    "PSA": {
      "ear.status": "contraindicated",
      "ear.trustworthiness-vector": {
        "instance-identity": 2,
        "executables": 96,
        "hardware": 2
      },
      "ear.appraisal-policy-id":
        "https://veraison.example/policy/1/60a0068d"
    }
  }
}
```

Figure 3: JSON claims-set: contraindicated appraisal

The breakdown of the trustworthiness vector is as follows:

- \* Instance Identity (affirming): recognized and not compromised
- \* Configuration (warning): known vulnerabilities
- \* Executables (contraindicated): contraindicated run-time
- \* File System (none): no claim being made
- \* Hardware (affirming): genuine

- \* Runtime Opaque (none): no claim being made
- \* Storage Opaque (none): no claim being made
- \* Sourced Data (none): no claim being made

The example in Figure 4 contains the appraisal for a composite device with two attesters named "CCA Platform" and "CCA Realm" respectively. Both attesters have either "affirming" or (implicit) "none" values in their associated trustworthiness vectors. Note that the "none" values can refer to either an AR4SI category that is unapplicable for the specific attester (ideally, the applicability should be specified by the evidence format itself), or to the genuine lack of information at the attester site regarding the specific category. For example, the reference values for the "CCA Realm" executables (i.e., the confidential computing workload) may not be known to the CCA platform verifier. In such cases, it is up to the downstream entity (typically, the relying party) to complete the partial appraisal.

```
{
  "eat_profile": "tag:ietf.org,2025-07:ear",
  "iat": 1666529300,
  "ear.verifier-id": {
    "developer": "https://veraison-project.org",
    "build": "vts 0.0.1"
  },
  "ear.raw-evidence": "NzQ3MjY5NzM2NTYzNzQKNzQ3MjY5NzM2NTYzNzQK",
  "submods": {
    "CCA Platform": {
      "ear.status": "affirming",
      "ear.trustworthiness-vector": {
        "instance-identity": 2,
        "executables": 2,
        "hardware": 2
      },
      "ear.appraisal-policy-id":
        "https://veraison.example/policy/1/60a0068d"
    },
    "CCA Realm": {
      "ear.status": "affirming",
      "ear.trustworthiness-vector": {
        "instance-identity": 2
      },
      "ear.appraisal-policy-id":
        "https://veraison.example/policy/1/60a0068d"
    }
  }
}
```



Figure 4: JSON claims-set: simple affirming appraisal

### 3.3. CBOR Serialisation

#### 3.3.1. Examples

The example in Figure 5 is semantically equivalent to that in Figure 3. It shows the same "contraindicated" appraisal using the more compact CBOR serialization of the EAR claims-set.

```
{
  265: "tag:ietf.org,2025-07:ear",
  6: 1666529184,
  1004: {
    0: "https://veraison-project.org",
    1: "vts 0.0.1"
  },
  1002: h'6C696665626F61746D616E',
  266: {
    "PSA": {
      1000: 96,
      1001: {
        0: 2,
        2: 96,
        4: 2
      },
    },
    1003: "https://veraison.example/policy/1/60a0068d"
  }
}
```

Figure 5: CBOR claims-set: contraindicated appraisal

## 4. EAR Extensions

EAR provides core semantics for describing the result of appraising attestation evidence. However, a given application may offer extra functionality to its relying parties, or tailor the attestation result to the needs of the application (e.g., TEEP [I-D.ietf-teep-protocol]). To accommodate such cases, both EAR and EAR-appraisal claims-sets can be extended by plugging new claims into the \$\$ear-extension (or \$\$ear-appraisal-extension, respectively) CDDL socket.

The rules that govern extensibility of EAR are those defined in [RFC8392] and [RFC7519] for CWTs and JWTs respectively.

An extension MUST NOT change the semantics of the EAR and EAR-appraisal claims-sets.

A receiver MUST ignore any unknown claim.

#### 4.1. Unregistered claims

An application-specific extension will normally mint its claim from the "private space" - using integer values less than -65536 for CWT, and Public or Private Claim Names as defined in Sections 4.2 and 4.3 of [RFC7519] when serializing to JWT.

It is RECOMMENDED that JWT EARs use Collision-Resistant Public Claim Names (Section 2 of [RFC7519]) rather than Private Claim Names.

#### 4.2. Registered claims

If an extension will be used across multiple applications, or is intended to be used across multiple environments, the associated extension claims SHOULD be registered in one, or both, the CWT and JWT claim registries.

In general, if the registration policy requires an accompanying specification document (as it is the case for "specification required" and "standards action"), such document SHOULD explicitly say that the extension is expected to be used in EAR claims-sets identified by this profile.

An up-to-date view of the registered claims can be obtained via the [IANA.cwt] and [IANA.jwt] registries.

#### 4.3. Choosing between registered and unregistered claims

If an extension supports functionality of a specific application (e.g. Project Veraison Services), its claims MAY be registered.

If an extension supports a protocol that may be applicable across multiple applications or environments (e.g., TEEP), its claims SHOULD be registered.

Since, in general, there is no guarantee that an application will be confined within an environment, it is RECOMMENDED that extension claims that have meaning outside the application's context are always registered.

It is also possible that claims that start out as application-specific acquire a more stable meaning over time. In such cases, it is RECOMMENDED that new equivalent claims are created in the "public space" and are registered as described in Section 4.2. The original "private space" claims SHOULD then be deprecated by the application.

#### 4.4. TEEP Extension

The TEEP protocol [I-D.ietf-teep-protocol] specifies the required claims that an attestation result must carry for a TAM (Trusted Application Manager) to make decisions on how to remediate a TEE (Trusted Execution Environment) that is out of compliance, or update a TEE that is requesting an authorized change.

The list is provided in Section 4.3.1 of [I-D.ietf-teep-protocol].

EAR defines a TEEP application extension for the purpose of conveying such claims.

```

;# import rfc9711 as eat
;# import rfc9393

$$ear-appraisal-extension //= (
  ear.teep-claims-label => ear-teep-claims
)

ear-teep-claims = non-empty<{
  ? eat.nonce-label => eat.nonce-type
  ? eat.ueid-label => eat.ueid-type
  ? eat.oemid-label => eat.oemid-pen / eat.oemid-ieee / eat.oemid-random
  ? eat.hardware-model-label => eat.hardware-model-type
  ? eat.hardware-version-label => eat.hardware-version-type
  ? eat.manifests-label => eat.manifests-type
}>

ear.teep-claims-label = eat.JC<"ear.teep-claims", 65000>
```

Figure 6: TEEP Extension (CDDL Definition)

##### 4.4.1. JSON Serialization Example

```
{
  "eat_profile": "tag:ietf.org,2025-07:ear",
  "iat": 1666529184,
  "ear.verifier-id": {
    "developer": "https://veraison-project.org",
    "build": "vts 0.0.1"
  },
  "ear.raw-evidence": "NzQ3MjY5NzM2NTYzNzQK",
  "submods": {
    "PSA": {
      "ear.status": "contraindicated",
      "ear.trustworthiness-vector": {
        "instance-identity": 2,
        "executables": 96,
        "hardware": 2
      },
      "ear.appraisal-policy-id":
        "https://veraison.example/policy/1/60a0068d",
      "ear.teep-claims": {
        "eat_nonce": "80FH7byS7VjfARIq0_KLqu6B9j-F79QtV6p",
        "ueid": "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyAh",
        "oemid": "Av8B",
        "hwmodel": "fJYq",
        "hwversion": ["1.2.5", 16384]
      }
    }
  }
}
```

#### 4.4.2. CBOR Serialization Example

```

{
  265: "tag:ietf.org,2025-07:ear",
  6: 1666529184,
  1004: {
    0: "https://veraison-project.org",
    1: "vts 0.0.1"
  },
  1002: h'6C696665626F61746D616E',
  266: {
    "PSA": {
      1000: 0,
      1001: {
        0: 2,
        1: 2,
        2: 2,
        4: 2
      },
      1003: "https://veraison.example/policy/1/60a0068d",
      65000: {
        10: h'948f8860d13a463e',
        256: h'0198f50a4ff6c05861c8860d13a638ea',
        258: 64242,
        259: h'ee80f5a66c1fb9742999a8fdab930893',
        260: ["1.2.5", 16384]
      }
    }
  }
}

```

#### 4.5. Project Veraison Extensions

The Project Veraison verifier defines three private, application-specific extensions:

##### ear.veraison.annotated-evidence

JSON representation of the evidence claims-set, including any annotations provided by the Project Veraison verifier. There are privacy considerations associated with this claim. See Section 8.

##### ear.veraison.policy-claims

any extra claims added by the policy engine in the Project Veraison verifier.

##### ear.veraison.key-attestation

contains the public key part of a successfully verified attested key. The key is a DER encoded ASN.1 SubjectPublicKeyInfo structure (Section 4.1.2.7 of [RFC5280]).

```

;# import rfc9711 as eat

$$ear-appraisal-extension //= (
    ear.veraison.annotated-evidence-label => ear-veraison-annotated-evidence
)

ear-veraison-annotated-evidence = {
    + text => any
}

$$ear-appraisal-extension //= (
    ear.veraison.policy-claims-label => ear-veraison-policy-claims
)

ear-veraison-policy-claims = {
    + text => any
}

$$ear-appraisal-extension //= (
    ear.veraison.key-attestation-label => ear-veraison-key-attestation
)

ear-veraison-key-attestation = {
    "akpub" => eat.binary-data
}

ear.veraison.annotated-evidence-label = eat.JC<"ear.veraison.annotated-evidence", -700
00>
ear.veraison.policy-claims-label = eat.JC<"ear.veraison.policy-claims", -70001>
ear.veraison.key-attestation-label = eat.JC<"ear.veraison.key-attestation", -70002>

```

Figure 7: Project Veraison Extensions (CDDL Definition)

#### 4.5.1. JSON Serialization Examples

```

{
  "eat_profile": "tag:ietf.org,2025-07:ear",
  "iat": 1666529184,
  "ear.verifier-id": {
    "developer": "https://veraison-project.org",
    "build": "vts 0.0.1"
  },
  "ear.raw-evidence": "NzQ3MjY5NzM2NTYzNzQK",
  "submods": {
    "PSA_IOT": {
      "ear.status": "contraindicated",
      "ear.trustworthiness-vector": {
        "instance-identity": 2,
        "executables": 96,

```

```

    "hardware": 2
  },
  "ear.appraisal-policy-id":
    "https://veraison.example/policy/1/60a0068d",
  "ear.veraison.annotated-evidence": {
    "eat-profile": "http://arm.com/psa/2.0.0",
    "psa-client-id": 1,
    "psa-security-lifecycle": 12288,
    "psa-implementation-id":
      "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA=",
    "psa-software-components": [
      {
        "measurement-value":
          "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA=",
        "signer-id":
          "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA="
      },
      {
        "measurement-value":
          "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA=",
        "signer-id":
          "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA="
      }
    ],
    "psa-nonce":
      "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA=",
    "psa-instance-id":
      "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyAh",
    "psa-certification-reference": "1234567890123-12345"
  },
  "ear.veraison.policy-claims": {
    "psa-certified": {
      "certificate-number": "1234567890123-12345",
      "date-of-issue": "23/06/2022",
      "test-lab": "Riscure",
      "certification-holder": "ACME Inc.",
      "certified-product": "RoadRunner",
      "hardware-version": "Gizmo v1.0.2",
      "software-version": "TrustedFirmware-M v1.0.6",
      "certification-type": "PSA Certified Level 1 v2.1",
      "developer-type": "PSA Certified Device"
    }
  }
}

```

```

{
  "eat_profile": "tag:ietf.org,2025-07:ear",
  "iat": 1666529184,
  "ear.verifier-id": {
    "developer": "https://veraison-project.org",
    "build": "vts 0.0.1"
  },
  "ear.raw-evidence": "NzQ3MjY5NzM2NTYzNzQK",
  "submods": {
    "PARSEC_TPM": {
      "ear.status": "affirming",
      "ear.trustworthiness-vector": {
        "instance-identity": 2,
        "executables": 2,
        "hardware": 2
      },
      "ear.appraisal-policy-id":
        "https://veraison.example/policy/1/60a0068d",
      "ear.veraison.key-attestation": {
        "akpub":
          "MFkwEwYHKoZIzj0CAQYIKoZIz____"
      }
    }
  }
}

```

#### 4.5.2. CBOR Serialization Example

```

{
  265: "tag:ietf.org,2025-07:ear",
  6: 1666529184,
  1004: {
    0: "https://veraison-project.org",
    1: "vts 0.0.1"
  },
  1002: h'6C696665626F61746D616E',
  266: {
    "PSA_IOT": {
      1000: 0,
      1001: {
        0: 2,
        1: 2,
        2: 2,
        4: 2
      },
    },
    1003: "https://veraison.example/policy/1/60a0068d",
    -70000: {
      "eat-profile": "http://arm.com/psa/2.0.0",
    }
  }
}

```



```

"psa-client-id": 1,
"psa-security-lifecycle": 12288,
"psa-implementation-id":
  "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA=",
"psa-software-components": [
  {
    "measurement-value":
      "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA=",
    "signer-id":
      "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA="
  },
  {
    "measurement-value":
      "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA=",
    "signer-id":
      "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA="
  }
],
"psa-nonce":
  "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyA=",
"psa-instance-id":
  "AQIDBAUGBwgJCgsMDQ4PEBESExQVFhcYGRobHB0eHyAh",
"psa-certification-reference": "1234567890123-12345"
},
-70001: {
  "psa-certified": {
    "certificate-number": "1234567890123-12345",
    "date-of-issue": "23/06/2022",
    "test-lab": "Riscure",
    "certification-holder": "ACME Inc.",
    "certified-product": "RoadRunner",
    "hardware-version": "Gizmo v1.0.2",
    "software-version": "TrustedFirmware-M v1.0.6",
    "certification-type": "PSA Certified Level 1 v2.1",
    "developer-type": "PSA Certified Device"
  }
}
}
}
}
}
}

```

## 5. Media Types

Media types for EAR are automatically derived from the base EAT media type [I-D.ietf-rats-eat-media-type] using the profile string defined in Section 3.

For example, a JWT serialization would use:

```
application/eat-jwt; eat_profile="tag:ietf.org,2025-07:ear"
```

A CWT serialization would instead use:

```
application/eat-cwt; eat_profile="tag:ietf.org,2025-07:ear"
```

## 6. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

### 6.1. Project Veraison

The organization responsible for this implementation is Project Veraison, a Linux Foundation project hosted at the Confidential Computing Consortium.

The organization currently provides two separate implementations: one in Golang another in C17.

The developers can be contacted on the Zulip channel:  
<https://veraison.zulipchat.com/#narrow/stream/357929-EAR/>.

#### 6.1.1. [github.com/veraison/ear](https://github.com/veraison/ear)

The software, hosted at <https://github.com/veraison/ear>, provides a Golang package that allows encoding, decoding, signing and verification of EAR payloads together with a CLI (arc) to create, verify and visualize EARs on the command line. The maturity level is currently alpha, and only the JWT serialization is implemented. The license is Apache 2.0. The package is used by the Project Veraison

verifier to produce attestation results.

#### 6.1.2. [github.com/veraison/c-ear](https://github.com/veraison/c-ear)

The software, hosted at <https://github.com/veraison/c-ear>, provides a C17 library that allows verification and partial decoding of EAR payloads. The maturity level is currently pre-alpha, and only the JWT serialization is implemented. The license is Apache 2.0. The library targets relying party applications that need to verify attestation results.

#### 6.1.3. [github.com/veraison/rust-ear](https://github.com/veraison/rust-ear)

The software, hosted at <https://github.com/veraison/rust-ear>, provides a Rust (2021 edition) library that allows verification and partial decoding of EAR payloads. The maturity level is currently pre-alpha, with limited algorithm support. Both JWT and COSE serializations are implemented. The license is Apache 2.0. The library targets verifiers that need to produce attestation results as well as relying party applications that need to verify and consume attestation results.

### 7. Security Considerations

TODO Security

### 8. Privacy Considerations

EAR is designed to expose as little identifying information as possible about the attester. However, certain EAR claims have direct privacy implications. Implementations should therefore allow applying privacy-preserving techniques to those claims, for example allowing their redaction, anonymisation or outright removal. Specifically:

- \* It SHOULD be possible to disable inclusion of the optional `ear.raw-evidence` claim
- \* It SHOULD be possible to disable inclusion of the optional `ear.veraison.annotated-evidence` claim
- \* It SHOULD be possible to allow redaction, anonymisation or removal of specific claims from the `ear.veraison.annotated-evidence` object

EAR is an EAT, therefore the privacy considerations in Section 8 of [I-D.ietf-rats-eat] apply.

## 9. IANA Considerations

### 9.1. New EAT Claims

This specification adds the following values to the "JSON Web Token Claims" registry [IANA.jwt] and the "CBOR Web Token Claims" registry [IANA.cwt].

Each entry below is an addition to both registries.

The "Claim Description", "Change Controller" and "Specification Documents" are common and equivalent for the JWT and CWT registries. The "Claim Key" and "Claim Value Types(s)" are for the CWT registry only. The "Claim Name" is as defined for the CWT registry, not the JWT registry. The "JWT Claim Name" is equivalent to the "Claim Name" in the JWT registry.

#### 9.1.1. EAR Status

- \* Claim Name: ear.status
- \* Claim Description: EAR Status
- \* JWT Claim Name: ear.status
- \* Claim Key: 1000 (suggested)
- \* Claim Value Type(s): unsigned integer (0, 2, 32, 96)
- \* Change Controller: IESG
- \* Specification Document(s): Section 3.1 of RFCthis

#### 9.1.2. Trustworthiness Vector

- \* Claim Name: ear.trustworthiness-vector
- \* Claim Description: EAR Trustworthiness Vector
- \* JWT Claim Name: ear.trustworthiness-vector
- \* Claim Key: 1001 (suggested)
- \* Claim Value Type(s): map
- \* Change Controller: IESG
- \* Specification Document(s): Section 3.1 of RFCthis

### 9.1.3. EAR Raw Evidence

- \* Claim Name: ear.raw-evidence
- \* Claim Description: EAR Raw Evidence
- \* JWT Claim Name: ear.raw-evidence
- \* Claim Key: 1002 (suggested)
- \* Claim Value Type(s): bytes
- \* Change Controller: IESG
- \* Specification Document(s): Section 3 of RFCthis

### 9.1.4. EAR Appraisal Policy Identifier

- \* Claim Name: ear.appraisal-policy-id
- \* Claim Description: EAR Appraisal Policy Identifier
- \* JWT Claim Name: ear.appraisal-policy-id
- \* Claim Key: 1003 (suggested)
- \* Claim Value Type(s): text
- \* Change Controller: IESG
- \* Specification Document(s): Section 3.1 of RFCthis

### 9.1.5. Verifier Software Identifier

- \* Claim Name: ear.verifier-id
- \* Claim Description: AR4SI Verifier Software Identifier
- \* JWT Claim Name: ear.verifier-id
- \* Claim Key: 1004 (suggested)
- \* Claim Value Type(s): map
- \* Change Controller: IESG
- \* Specification Document(s): Section 3 of RFCthis

### 9.1.6. EAR TEEP Claims

TODO

## 10. References

### 10.1. Normative References

#### [I-D.ietf-rats-ar4si]

Voit, E., Birkholz, H., Hardjono, T., Fossati, T., and V. Scarlata, "Attestation Results for Secure Interactions", Work in Progress, Internet-Draft, draft-ietf-rats-ar4si-08, 6 February 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-rats-ar4si-08>>.

#### [I-D.ietf-rats-eat]

Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "The Entity Attestation Token (EAT)", Work in Progress, Internet-Draft, draft-ietf-rats-eat-31, 6 September 2024, <<https://datatracker.ietf.org/doc/html/draft-ietf-rats-eat-31>>.

#### [I-D.ietf-rats-eat-media-type]

Lundblade, L., Birkholz, H., and T. Fossati, "EAT Media Types", Work in Progress, Internet-Draft, draft-ietf-rats-eat-media-type-12, 3 November 2024, <<https://datatracker.ietf.org/doc/html/draft-ietf-rats-eat-media-type-12>>.

#### [I-D.ietf-teep-protocol]

Tschofenig, H., Pei, M., Wheeler, D. M., Thaler, D., and A. Tsukamoto, "Trusted Execution Environment Provisioning (TEEP) Protocol", Work in Progress, Internet-Draft, draft-ietf-teep-protocol-21, 3 March 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-teep-protocol-21>>.

[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/rfc/rfc2119>>.

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

- [RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <<https://www.rfc-editor.org/rfc/rfc7519>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.
- [RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <<https://www.rfc-editor.org/rfc/rfc8392>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/rfc/rfc8610>>.
- [STD94] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <<https://www.rfc-editor.org/rfc/rfc8949>>.
- [STD96] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <<https://www.rfc-editor.org/rfc/rfc9052>>.

## 10.2. Informative References

- [IANA.cwt] IANA, "CBOR Web Token (CWT) Claims", <<https://www.iana.org/assignments/cwt>>.
- [IANA.jwt] IANA, "JSON Web Token (JWT)", <<https://www.iana.org/assignments/jwt>>.
- [RFC4151] Kindberg, T. and S. Hawke, "The 'tag' URI Scheme", RFC 4151, DOI 10.17487/RFC4151, October 2005, <<https://www.rfc-editor.org/rfc/rfc4151>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/rfc/rfc7942>>.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <<https://www.rfc-editor.org/rfc/rfc9334>>.

#### Appendix A. Common CDDL Types

##### non-empty

A CDDL generic that can be used to ensure the presence of at least one item in an object with only optional fields.

non-empty<M> = (M) .within ({ + any => any })

#### Appendix B. Open Policy Agent Example

Open Policy Agent OPA (<https://www.openpolicyagent.org>) is a popular and flexible policy engine that is used in a variety of contexts, from cloud to IoT. OPA policies are written using a purpose-built, declarative programming language called Rego (<https://www.openpolicyagent.org/docs/latest/policy-language/>). Rego has been designed to handle JSON claim-sets and their JWT envelopes as first class objects, which makes it an excellent fit for dealing with JWT EARs.

The following example illustrates an OPA policy that a Relying Party would use to make decisions based on a JWT EAR received from a trusted verifier.



```
package ear

ear_appraisal = {
  "verified": signature_verified,
  "appraisal-status": status,
  "trustworthiness-vector": trust_vector,
} {
  # verify EAR signature is correct and from one of the known and
  # trusted verifiers
  signature_verified := io.jwt.verify_es256(
    input.ear_token,
    json.marshal(input.trusted_verifiers)
  )

  # extract the EAR claims-set
  [_, payload, _] := io.jwt.decode(input.ear_token)

  # access the attester-specific appraisal record
  app_rec := payload.submods.PARSEC_TPM
  status := app_rec["ear.status"] == "affirming"

  # extract the trustworthiness vector for further inspection
  trust_vector := app_rec["ear.trustworthiness-vector"]
}

# add further conditions on the trust_vector here
# ...
```

The result of the policy appraisal is the following JSON object:

```
{
  "ear_appraisal": {
    "appraisal-status": true,
    "trustworthiness-vector": {
      "executables": 2,
      "hardware": 2,
      "instance-identity": 2
    },
    "verified": true
  }
}
```

For completeness, the trusted verifier public key and the EAR JWT used in the example are provided below.

```
===== NOTE: '\' line wrapping per RFC 8792 =====
{
    "ear_token": "eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.eyJlYXIucmF3LWV2aWRlbmNlIjoiTnpRM01qWTVOek0yTlRZek56UUsiLCJlYXludmVyaWZpZXItaWQiOlsiYnVpbGQoiOiJ2dHMgMC4wLjEiLCJkZXZlbnBG9wZXIIiOiJodHRwcovL3ZlcmFpc29uLXBib2plyY3Qub3JnIn0sImVhdF9wcm9maWx1IjoiaHR0cHMTg0ZSswOSwianRpIjoINTViOGIzMfkoGRkMWQ4ZWZfjNGU0OGYxMTdmZTUwOGIxMWY4NDRkOWYwMTg5YmZlZDliODclMTVhbjclNDI2NCIsIm5iZiI6MTY3NzI0Nzg3OSwic3VibW9kcYi6eyJQVJTUNfVFBNiCjpp7ImVhc\i5hcHBByYWlzYWwtcG9saWN5LWlkIjoiaHR0cHMTg0ZSswOSwianRpIjoINTViOGIzMfkoGRkMWQ4ZWZfjNGU0OGYxMTdmZTUwOGIxMWY4NDRkOWYwMTg5YmZlZDliODclMTVhbjclNDI2NCIsIm5iZiI6MTY3NzI0Nzg3OSwic3VibW9kcYi6eyJQVJTUNfVFBNiCjpp7ImVhc\\GljeS8xLzYwYTAWnjhkIiwizWFyLnN0YXRlcyl6ImFmZmlybWluZyIsImVhci50cnVzd\\HdvcnRoaw5lc3MtdmVjdG9yIjp7ImV4ZW5ldGFibGVzIjoyLCJoYXJkd2FyZSI6Miwiaw5zdGFuY2UtawRlbnRpdHkiOjJ9LCJlYXludmVyYWlzb24ua2V5LWF0dGVzdGF0aW9uIjp7ImFrcHVhIjoitUZrd0V3WUhLb1pJemowQ0FRWUlLb1pJemowREFRY0RRZ0FFY2pTc\\DhfTVdNM2d5OFR1Z1dPMVRwUVNqX3ZJa3NMCEmtZzhSNVMzbHBHYjdQV1dhB0NBakVQOF9BNTlWWndMWGD3blp6TjBXeHVCUGpwYVdpV3NmQ1EIFx19fQ.3Ym-f1LEgamxePUM7h\\6Y2RJdGh9eeLOxKorOnlwE9jdAnLNwm3rTKFKV2S2LbqVFoDtK9QGalt2t5RnUdfwZNmg\\",
    "trusted_verifiers": {
        "keys": [
            {
                "alg": "ES256",
                "crv": "P-256",
                "kty": "EC",
                "x": "usWxHK2PmfN HKwXPS54m0kTcGJ90UiglWiGahtagnv8",
                "y": "IBOL-C3BttVivg-lSreASjpkttcsz-lrb7btKLv8EX4"
            }
        ]
    }
}
```

## Appendix C. Open Issues

```
// Note to RFC Editor: please remove before publication.
```

The list of currently open issues for this documents can be found at <https://github.com/thomas-fossati/draft-ear/issues>.

## Appendix D. Document History

```
// Note to RFC Editor: please remove before publication.
```

## D.1. draft-fv-rats-ear-00

Initial release.

## D.2. draft-fv-rats-ear-01

- \* privacy considerations
- \* OPA policy example
- \* add rust-ear crate to the implementation status section

## D.3. draft-fv-rats-ear-02

- \* align JWT and CWT representations of eat\_nonce

## Acknowledgments

Many thanks to Dave Thaler, Greg Kostal, Simon Frost, Yogesh Deshpande for helpful comments and discussions that have shaped this document.

## Authors' Addresses

Thomas Fossati  
Linaro  
Email: [thomas.fossati@linaro.org](mailto:thomas.fossati@linaro.org)

Eric Voit  
Cisco  
Email: [evoit@cisco.com](mailto:evoit@cisco.com)

Sergei Trofimov  
Arm Limited  
Email: [sergei.trofimov@arm.com](mailto:sergei.trofimov@arm.com)

Henk Birkholz  
Fraunhofer SIT  
Rheinstrasse 75  
64295 Darmstadt  
Germany  
Email: [henk.birkholz@ietf.contact](mailto:henk.birkholz@ietf.contact)