Remote ATtestation ProcedureS
Internet-Draft
Intended status: Standards Track
Expires: 4 January 2026

S. Frost
Arm
T. Fossati
Linaro
H. Tschofenig
H-BRS

3 July 2025

EAT Measured Component
draft-ietf-rats-eat-measured-component-04

Abstract

The term "measured component" refers to an object within the attester’s target environment whose state can be inspected and, typically, digested. A digest is computed through a cryptographic hash function. Examples of measured components include firmware stored in flash memory, software loaded into memory at start time, data stored in a file system, or values in a CPU register.

This document defines a "measured component" format that can be used with the EAT Measurements claim.

Discussion Venues

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

Discussion of this document takes place on the Remote ATtestation ProcedureS Working Group mailing list (rats@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/rats/.

Source for this draft and an issue tracker can be found at https://github.com/thomas-fossati/draft-fft-rats-eat-measured-component.

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/.

This Internet-Draft will expire on 4 January 2026.

Copyright Notice

Table of Contents

1. Introduction

Section 4.2.16 of [I-D.ietf-rats-eat] defines a Measurements claim that:

> "[c]ontains descriptions, lists, evidence or measurements of the software that exists on the entity or any other measurable subsystem of the entity."

This claim allows for different measurement formats, each identified by a different CoAP Content-Format (Section 12.3 of [RFC7252]). Currently, the only specified format is CoSWID of type "evidence", as per Section 2.9.4 of [RFC9393].

This document introduces a "measured component" format that can be used with the EAT Measurements claim in addition to or as an alternative to CoSWID.

The term "measured component" refers to any measurable object on a target environment, that is, an object whose state can be sampled and, possibly, digested. This includes, for example: the invariant part of a firmware component that is loaded in memory at startup time, a run-time integrity check (RTIC), a file system object, or a CPU register.

2. Conventions and Definitions

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

In this document, CDDL [RFC8610] [RFC9165] [I-D.ietf-cbor-cddl-modules] [I-D.ietf-cbor-cddl-more-control] is used to describe the data formats.

3. Information Model

A "measured component" information element includes the component's sampled state (in digested or raw form) along with metadata that helps in identifying the component. Optionally, any entities responsible for signing the installed component can also be specified.

The information model of a "measured component" is described in Table 1.

IE	Description	Requirement Level
Component Name	The name given to the measured component. It is important that this name remains consistent across different releases to allow for better tracking of the same measured item across updates. When combined with a consistent versioning scheme, it enables better signaling from the appraisal procedure to the relying parties.	REQUIRED
Component Version	A value representing the specific release or development version of the measured component. Using Semantic Versioning (https://semver.org/spec/v2.0.0.html) is RECOMMENDED.	OPTIONAL
Digested or Raw Value	Either the raw value or the digested value of the measured component.	REQUIRED
Digest Algorithm	Hash algorithm used to compute the Digest Value.	REQUIRED only if the value is in the digested form
Signers	One or more unique identifiers of entities signing the measured component.	OPTIONAL

Table 1: Measured Component Information Elements

The format SHOULD also allow a limited amount of extensibility to accommodate profile-specific semantics.

4. Data Model

This section presents a JSON and CBOR data model that implements the information model outlined in Section 3.

The data model is inspired by the "PSA software component" claim (Section 4.4.1 of [I-D.tschofenig-rats-psa-token]), which has been refactored to take into account the recommendations about the design of new EAT claims described in Appendix E of [I-D.ietf-rats-eat].

CDDL is used to express rules and constraints of the data model for both JSON and CBOR. These rules must be strictly followed when creating or validating "measured component" data items. When there is variation between CBOR and JSON, the JC<> CDDL generic defined in Appendix D of [I-D.ietf-rats-eat] is used.

4.1. Common Types

The following three basic types are used at various places within the measured component data model:

```
bytes-b64u = text .b64u bytes
bytes8 = bytes .size 8
bytes8-b64u = text .b64u bytes8
```

4.2. The measured-component Data Item

The measured-component data item is as follows:

```
;# import corim.digest from rfcYYYY as corim
;# import eat.JC from rfc9711 as eat

measured-component = {
  id-label => component-id
  measurement
  ? signers-label => [ + signer-type ]
  ? flags-label => flags-type
}

measurement //= ( digested-measurement-label => corim.digest )
measurement //= ( raw-measurement-label => bytes )

signer-type = eat.JC<bytes-b64u, bytes>
flags-type = eat.JC<bytes8-b64u, bytes8>

id-label = eat.JC<"id", 1>
digested-measurement-label = eat.JC<"digested-measurement", 2>
raw-measurement-label = eat.JC<"raw-measurement", 5>
signers-label = eat.JC<"signers", 3>
flags-label = eat.JC<"flags", 4>
```

The members of the measured-component CBOR map / JSON object are:

"id" (index 1):
: The measured component identifier encoded according to the format described in Section 4.2.1.

"measurement":
: Either a digest value and algorithm (index 2), encoded using CoRIM digest format (Section 1.3.8 of [I-D.ietf-rats-corim]), or the "raw" measurement (index 5), encoded as a byte string.

"signers" (index 3):
: One or more signing entities, see Section 4.2.2.

"flags" (index 4):
: a 64-bit field with profile-defined semantics, see Section 4.2.3.

4.2.1. Component Identifier

The component-id data item is as follows:

```
component-id = [
  name:      text
  ? version: version
]

;# import coswid.$version-scheme from rfc9393 as coswid

version = [
  val:      text
  ? scheme: coswid.$version-scheme
]
```

name A string that provides a human readable identifier for the component in question. Format and adopted conventions depend on the component type.

version A compound version data item that reuses encoding and semantics of [I-D.ietf-rats-eat] sw-version-type.

4.2.2. Signer

A signer is an entity that digitally signed the measured component. Typically, the signature is verified during installation or when the measured component is executed by the boot ROM, operating system, or application launcher. For example, as in UEFI Secure Boot [UEFI2] and Arm Trusted Board Boot [TBBR-CLIENT]. Another example may be the controlling entity in an app store. It is important to note that a signer is different from the identity of the manufacturer of the component, such as would be found in a manifest like a payload CoSWID.

A signer is associated with a public key. It could be an X.509 certificate, a raw public key, a public key thumbprint, or some other identifier that can be uniquely associated with the signing entity. In some cases, multiple parties may need to sign a component to indicate their endorsement or approval. This could include roles such as a firmware update system, fleet owner, or third-party auditor. The specific purpose of each signature may depend on the deployment, and the order of signers within the array could indicate meaning.

If an EAT profile (Section 6 of [I-D.ietf-rats-eat]) uses measured components, it MUST specify whether the signers field is used. If it is used, the profile MUST also specify what each of the entries in the signers array represents, and how to interpret the corresponding signer-type.

The signer-type is defined as follows:

```
signer-type = eat.JC<bytes-b64u, bytes>
```

4.2.3. Profile-specific Flags

This field contains at most 64-bit of profile-defined semantics. It can be used to carry information in fixed-size chunks, such as a bit mask or a single value within a predetermined set of codepoints. Regardless of its internal structure, the size of this optional field is exactly 8 bytes.

The flags-type is defined as follows:

```
flags-type = eat.JC<bytes8-b64u, bytes8>
```

If an EAT profile (Section 6 of [I-D.ietf-rats-eat]) uses measured components, it MUST specify whether the profile-flags field is used. If it is used, the profile MUST also specify how to interpret the 64 bits.

4.3. EAT measurements-format Extensions

The CDDL in Figure 1 extends the $measurements-body-cbor and $measurements-body-json EAT sockets to add support for measured-components to the Measurements claim.

```
mc-cbor = bytes .cbor measured-component
mc-json = text .json measured-component

; EAT CBOR (‘.feature "cbor"‘)
$measurements-body-cbor /= mc-cbor ; native
$measurements-body-cbor /= mc-json ; tunnel

; EAT JSON (‘.feature "json"‘)
$measurements-body-json /= mc-json            ; native
$measurements-body-json /= text .b64u mc-cbor ; tunnel
```

Figure 1: EAT measurements-format Extensions

Each socket is extended with two new types: a "native" representation that is used when measured-component and the EAT have the same serialization (e.g., they are both CBOR), and a "tunnel" representation that is used when the serializations differ.

4.4. measurements-format for CBOR EAT

The entries in Table 2 are the allowed content-type / content-format pairs when the measured-component is carried in a CBOR EAT.

Note the use of the "native" and "tunnel" formats from Figure 1, and how the associated CoAP Content-Format is used to describe the original serialization.

content-type (CoAP C-F equivalent)	content-format
application/measured-component+cbor	mc-cbor
application/measured-component+json	mc-json

Table 2: measurement-format for EAT CWT

4.5. measurements-format for JSON EAT

Table 3 is the equivalent of Table 2 for JSON-serialized EAT.

content-type (CoAP C-F equivalent)	content-format
application/measured-component+json	mc-json
application/measured-component+cbor	tstr .b64u mc-cbor

Table 3: measurement-format for EAT JWT

5. EAT Profiles and Measured Components

The semantics of the signers and profile flags fields are defined by the applicable EAT profile, i.e., the profile of the wrapping EAT.

If the profile of the EAT is not known to the consumer and one or more Measured Components within that EAT include signers and/or profile flags, the consumer MUST reject the EAT.

6. Examples

The example in Figure 2 is a digested measured component with all the fields populated.

```
{
  / id / 1: [
    / name / "boot loader X",
    / version / [
      "1.2.3rc2",
      16384 / semver /
    ]
  ],
  / measurement / 2: [
    / alg / "sha-256",
    / val / h'3996003d486fb91ffb056f7d03f2b2992b215b31dbe7af4b37
              3431fc7d319da3'
  ],
  / signers / 3: [
    h'492e9b676c21f6012b1ceeb9032feb4141a880797355f6675015ec59c5
      1ca1ec',
    h'4277bb97ba7b51577a0d38151d3e08b40bdf946753f5b5bdeb814d6ff5
      7a8a5e'
  ],
  / profile-flags / 4: h'0000000000000101'
}
```

Figure 2: Complete Measured Component

The example in Figure 3 is the same measured component as above but used as the format of a measurements claim in a EAT claims-set.

The example uses TBD1 as the content-type value of the measurements-format entry. (This will change to the value assigned by IANA to the mc+cbor Content-Format.)

Note that the array contains only one measured component, but additional entries could be added if the measured TCB is made of multiple, individually measured components.

```
{
  273: [
    [
      TBD1, / mc+cbor /
      <<
        {
          / id / 1: [
            / name / "boot loader X",
            / version / [
              "1.2.3rc2",
              16384 / semver /
            ]
          ],
          / measurement / 2: [
            / alg / "sha-256",
            / val / h'3996003d486fb91ffb056f7d03f2b2992b215b31db
                      e7af4b373431fc7d319da3'
          ],
          / signers / 3: [
            h'492e9b676c21f6012b1ceeb9032feb4141a880797355f66750
              15ec59c51ca1ec',
            h'4277bb97ba7b51577a0d38151d3e08b40bdf946753f5b5bdeb
              814d6ff57a8a5e'
          ]
        }
      >>
    ]
  ]
}
```

Figure 3: EAT Measurements Claim using a Measured Component (CBOR)

The example in Figure 4 illustrates the inclusion of a JSON measured component inside a JSON EAT.

The example uses TBD2 as the content-type value of the measurements-format entry. (This will change to the value assigned by IANA to the mc+json Content-Format.)

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

{
  "measurements": [
    [
      TBD2, / mc+json /
      "{ \"id\": [ \"boot loader X\", [ \"1.2.3rc2\", 16384 ] ], \"\
digested-measurement\": [ \"sha-256\", \"\
OZYAPUhvuR_7BW99A_KymSshWzHb569LNzQx_H0xnaM\" ], \"signers\": [ \"\
SS6bZ2wh9gErHO65Ay_rQUGogHlzVfZnUBXsWcUcoew\", \"\
                  Qne7l7p7UVd6DTgVHT4ItAvflGdT9bW964FNb_V6il4\" ] }"
    ]
  ]
}
```

Figure 4: EAT Measurements Claim using a Measured Component (JSON)

The example in Figure 5 is a measured component representing a boot loader identified by its path name:

```
{
  / id / 1: [
    / name / "/boot/loader.bin"
  ],
  / measurement / 2: [
    / alg / "sha-384",
    / val / h'66ec2fb4e02d8c8b3eee320e750d9389d66c52c51db11cc6
              9cc5e410816283ed60ba573795f5fcc85e513af57b3f6def'
  ],
  / profile-flags / 4: h'0000000000000101'
}
```

Figure 5: Digested Measured Component using File Path as Identifier

The example in Figure 6 is a raw measured component.

```
{
  / id / 1: [
    / name / "hardware-config"
  ],
  / measurement / 5: h'4f6d616861'
}
```

Figure 6: Raw Measured Component

7. Security and Privacy Considerations

The Name and Version of a component can give an attacker detailed information about the software running on a device and its configuration settings. This information could offer an attacker valuable insights. Additionally, the stability requirement of the component's Name could potentially allow for tracking.

8. IANA Considerations

// RFC Editor: replace "RFCthis" with the RFC number assigned to this document.

8.1. Media Types Registrations

IANA is requested to add the following media types to the "Media Types" registry [IANA.media-types].

Name	Template	Reference
mc+cbor	application/measured-component+cbor	RFCthis
mc+json	application/measured-component+json	RFCthis

Table 4: Measured Component Media Types

8.1.1. application/measured-component+cbor

Type name: application
Subtype name: measured-component+cbor
Required parameters: n/a
Optional parameters: n/a
Encoding considerations: binary (CBOR)
Security considerations: Section 7 of RFCthis
Interoperability considerations: n/a
Published specification: RFCthis
Applications that use this media type: Attesters, Verifiers and Relying Parties
Fragment identifier considerations: The syntax and semantics of fragment identifiers are as specified for "application/cbor". (No fragment identification syntax is currently defined for "application/cbor".)
Person & email address to contact for further information: RATS WG mailing list (rats@ietf.org)
Intended usage: COMMON

Restrictions on usage: none
Author/Change controller: IETF
Provisional registration: no

8.1.2. application/measured-component+json

Type name: application
Subtype name: measured-component+json
Required parameters: n/a
Optional parameters: n/a
Encoding considerations: binary (JSON is UTF-8-encoded text)
Security considerations: Section 7 of RFCthis
Interoperability considerations: n/a
Published specification: RFCthis
Applications that use this media type: Attesters, Verifiers and Relying Parties
Fragment identifier considerations: The syntax and semantics of fragment identifiers are as specified for "application/json". (No fragment identification syntax is currently defined for "application/json".)
Person & email address to contact for further information: RATS WG mailing list (rats@ietf.org)
Intended usage: COMMON
Restrictions on usage: none
Author/Change controller: IETF
Provisional registration: no

8.2. Measured Component Content-Format Registrations

IANA is requested to register two Content-Format numbers in the "CoAP Content-Formats" sub-registry, within the "Constrained RESTful Environments (CoRE) Parameters" Registry [IANA.core-parameters], as follows:

Content-Type	Content Coding	ID	Reference
application/measured-component+cbor	-	TBD1	RFCthis
application/measured-component+json	-	TBD2	RFCthis

Table 5

If possible, TBD1 and TBD2 should be assigned in the 256..9999 range.

9. References

9.1. Normative References

[I-D.ietf-cbor-cddl-modules]
Bormann, C. and B. Moran, "CDDL Module Structure", Work in Progress, Internet-Draft, draft-ietf-cbor-cddl-modules-04, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-cbor-cddl-modules-04>.

[I-D.ietf-cbor-cddl-more-control]
Bormann, C., "Concise Data Definition Language (CDDL): Additional Control Operators for the Conversion and Processing of Text", Work in Progress, Internet-Draft, draft-ietf-cbor-cddl-more-control-08, 9 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-cbor-cddl-more-control-08>.

[I-D.ietf-rats-corim]
Birkholz, H., Fossati, T., Deshpande, Y., Smith, N., and W. Pan, "Concise Reference Integrity Manifest", Work in Progress, Internet-Draft, draft-ietf-rats-corim-07, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-corim-07>.

[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>.

[IANA.core-parameters]
IANA, "Constrained RESTful Environments (CoRE) Parameters", <https://www.iana.org/assignments/core-parameters>.

[IANA.media-types]
IANA, "Media Types", <https://www.iana.org/assignments/media-types>.

[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>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/rfc/rfc7252>.

[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>.

[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>.

[RFC9165] Bormann, C., "Additional Control Operators for the Concise Data Definition Language (CDDL)", RFC 9165, DOI 10.17487/RFC9165, December 2021, <https://www.rfc-editor.org/rfc/rfc9165>.

9.2. Informative References

[I-D.tschofenig-rats-psa-token] Tschofenig, H., Frost, S., Brossard, M., Shaw, A. L., and T. Fossati, "Arm’s Platform Security Architecture (PSA) Attestation Token", Work in Progress, Internet-Draft, draft-tschofenig-rats-psa-token-24, 23 September 2024, <https://datatracker.ietf.org/doc/html/draft-tschofenig-rats-psa-token-24>.

[RFC9393] Birkholz, H., Fitzgerald-McKay, J., Schmidt, C., and D. Waltermire, "Concise Software Identification Tags", RFC 9393, DOI 10.17487/RFC9393, June 2023, <https://www.rfc-editor.org/rfc/rfc9393>.

[TBBR-CLIENT] Arm Ltd, "Trusted Board Boot Requirements Client (TBBR-CLIENT) Armv8-A", ARM DEN0006D, September 2018, <https://developer.arm.com/documentation/den0006>.

[UEFI2] UEFI Forum, Inc., "Unified Extensible Firmware Interface (UEFI) Specification", August 2022, <https://uefi.org/sites/default/files/resources/UEFI_Spec_2_10_Aug29.pdf>.

Appendix A. Open Issues

The list of currently open issues for this documents can be found at https://github.com/thomas-fossati/draft-fft-rats-eat-measured-component/issues.

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

Acknowledgments

The authors would like to thank Carl Wallace, Carsten Bormann, Dionna Glaze, Giridhar Mandyam, Laurence Lundblade and Michael Richardson for providing comments, reviews and suggestions that greatly improved this document.

Authors' Addresses

Simon Frost
Arm
Email: Simon.Frost@arm.com

Thomas Fossati
Linaro
Email: Thomas.Fossati@linaro.org

Hannes Tschofenig
University of Applied Sciences Bonn-Rhein-Sieg
Email: Hannes.Tschofenig@gmx.net