Remote ATtestation ProcedureS
Internet-Draft
Intended status: Standards Track
Expires: 22 May 2026

H. Birkholz
Fraunhofer SIT
N. Smith
Independent
T. Fossati
Linaro
H. Tschofenig
H-BRS
D. Glaze
Google LLC

18 November 2025

RATS Conceptual Messages Wrapper (CMW)
draft-ietf-rats-msg-wrap-21

Abstract

The Conceptual Messages introduced by the RATS architecture (RFC 9334) are protocol-agnostic data units that are conveyed between RATS roles during remote attestation procedures. Conceptual Messages describe the meaning and function of such data units within RATS data flows without specifying a wire format, encoding, transport mechanism, or processing details. The initial set of Conceptual Messages is defined in Section 8 of RFC 9334 and includes Evidence, Attestation Results, Endorsements, Reference Values, and Appraisal Policies.

This document introduces the Conceptual Message Wrapper (CMW) that provides a common structure to encapsulate these messages. It defines a dedicated CBOR tag, corresponding JSON Web Token (JWT) and CBOR Web Token (CWT) claims, and an X.509 extension.

This allows CMWs to be used in CBOR-based protocols, web APIs using JWTs and CWTs, and PKIX artifacts like X.509 certificates. Additionally, the draft defines a media type and a CoAP content format to transport CMWs over protocols like HTTP, MIME, and CoAP.

The goal is to improve the interoperability and flexibility of remote attestation protocols. By introducing a shared message format like the CMW, we can consistently support different attestation message types, evolve message serialization formats without breaking compatibility, and avoid having to redefine how messages are handled in each protocol.

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-ftbs-rats-msg-wrap.

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 May 2026.

Copyright Notice

Table of Contents

1. Introduction

The Conceptual Messages introduced by the Remote ATtestation procedureS (RATS) architecture [RFC9334] are protocol-agnostic data units that are conveyed between RATS roles during remote attestation procedures. Conceptual Messages describe the meaning and function of such data units within RATS data flows without specifying a wire format, encoding, transport mechanism, or processing details. The initial set of Conceptual Messages is defined in Section 8 of [RFC9334] and includes Evidence, Attestation Results, Endorsements, Reference Values, and Appraisal Policies.

Each conceptual message can have multiple claims encoding and serialization formats (Section 9 of [RFC9334]). Throughout their lifetime, RATS conceptual messages are typically transported over different protocols. For example,

* In a "background check" topology, Evidence (e.g., EAT [RFC9711]) first flows from the Attester to the Relying Party and then from the Relying Party to the Verifier, each leg following a separate protocol path.

* In a "passport" topology, an attestation result payload (e.g., Attestation Results for Secure Interactions (AR4SI) [I-D.ietf-rats-ar4si]) is initially sent from the Verifier to the Attester, and later, via a different channel, from the Attester to the Relying Party.

By using the CMW format outlined in this document, protocol designers can avoid the need to update protocol specifications to accommodate different conceptual messages and serialization formats used by various attestation technologies. This approach streamlines the implementation process for developers, enabling easier support for diverse attestation technologies. For instance, a Relying Party application implementer does not need to parse attestation-related messages, such as Evidence from Attesters on IoT devices with Trusted Platform Modules (TPM) or servers using confidential computing hardware like Intel Trust Domain Extensions (TDX). Instead, they can leverage the CMW format, remaining agnostic to the specific attestation technology.

A further design goal is extensibility. This means that adding support for new conceptual messages and new attestation technologies should not change the core of the processor, and that a CMW stack can be designed to offer a plug-in interface for both encoding and decoding. To achieve this, the format must provide consistent message encapsulation and explicit typing. These features allow for selecting the appropriate message handler based on its type identifier. An opaque message can then be passed between the core and the handler.

This document defines two encapsulation formats for RATS conceptual messages that aim to achieve the goals stated above.

These encapsulation formats have been specifically designed to possess the following characteristics:

* They are self-describing, which means that they can convey precise typing information without relying on the framing provided by the embedding protocol or the storage system.

* They are based on media types [RFC6838], which allows the cost of their registration to be spread across numerous usage scenarios.

A protocol designer could use these formats, for example, to convey Evidence, Endorsements and Reference Values in certificates and CRLs extensions ([DICE-arch]), to embed Attestation Results or Evidence as first-class authentication credentials in TLS handshake messages [I-D.fossati-tls-attestation] [I-D.fossati-tls-exported-attestation], to transport attestation-related payloads in RESTful APIs, or for stable storage of Attestation Results in the form of file system objects.

This document also defines corresponding CBOR tag, JSON Web Tokens (JWT) and CBOR Web Tokens (CWT) claims, as well as an X.509 extension. These allow embedding the wrapped conceptual messages into CBOR-based protocols, web APIs, and PKIX formats and protocols. In addition, a Media Type and a CoAP Content-Format are defined for transporting CMWs in HTTP, MIME, CoAP and other Internet protocols.

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] is used to describe the data formats.

The reader is assumed to be familiar with the vocabulary and concepts defined in [RFC9334].

This document reuses the terms defined in Section 2 of [RFC9193] (e.g., "Content-Type").

3. Conceptual Message Wrappers

A RATS Conceptual Message Wrapper (CMW) has a tree structure. Leaf nodes are of type "Record" (Section 3.1), or "Tag" (Section 3.2). Intermediate nodes are of type "Collection" (Section 3.3); they hold together multiple CMW items.

The following snippet outlines the productions associated with the top-level types.

```
start = cmw

cmw = json-cmw / cbor-cmw

json-cmw = json-record / json-collection
cbor-cmw = cbor-record / cbor-collection / $cbor-tag
```

The complete CDDL can be found in Section 6.

Section 4.3 and Section 4.4 describe the transport of CMWs using CBOR and JSON Web Tokens and PKIX formats, including Certificate Signing Requests (CSRs), X.509 Certificates, and Certificate Revocation Lists (CRLs).

This document only defines an encapsulation, not a security format. It is the responsibility of the Attester to ensure that the CMW contents have the necessary security protection. Security considerations are discussed in Section 9.

3.1. Record CMW

The format of the Record CMW is shown in Figure 1. The JSON [STD90] and CBOR [STD94] representations are provided separately. Both the json-record and cbor-record have the same fields except for slight differences in the types discussed below.

```
json-record = [
  type: media-type
  value: base64url-string
  ? ind: uint .bits cm-type
]

cbor-record = [
  type: coap-content-format-type / media-type
  value: bytes
  ? ind: uint .bits cm-type
]
```

Figure 1: CDDL definition of the Record CMW

Each contains two or three members:

type:
: Either a text string representing a Content-Type (e.g., an EAT media type [RFC9782]) or an unsigned integer corresponding to a CoAP Content-Format ID (Section 12.3 of [RFC7252]). The latter is not used in the JSON serialization.

value:
: The RATS conceptual message serialized according to the value defined in the type member. When using JSON, the value field MUST be encoded as Base64 using the URL and filename safe alphabet (Section 5 of [RFC4648]) without padding. This always applies, even if the conceptual message format is already textual (e.g., a JWT EAT). When using CBOR, the value field MUST be encoded as a CBOR byte string.

ind:
: An optional bitmap with a maximum size of 4 bytes that indicates which conceptual message types are carried in the value field. Any combination (i.e., any value between 1 and 2^32-1 inclusive) is allowed. Only five bits are registered at the time of writing, so, the acceptable values are currently limited to 1 to 31. This is useful only if the type is potentially ambiguous and there is no further context available to the CMW consumer to decide. For example, this might be the case if the base media type is not profiled (e.g., application/eat+cwt), if the value field contains multiple conceptual messages with different types (e.g., both Reference Values and Endorsements within the same application/rim+cose), or if the same profile identifier is shared by different conceptual messages. The value MUST be non-zero. The absence of conceptual message indicator information is indicated by omitting the ind field entirely. For further details, see Section 3.1.1.

3.1.1. CM Type

The cm-type type is the control type for the ind field. As such, it indicates which bits are allowed to be set in the ind byte string.

```
cm-type = &(
  reference-values: 0
  endorsements: 1
  evidence: 2
  attestation-results: 3
  appraisal-policy: 4
)
```

Figure 2: CDDL definition of the CM Type

The cm-type currently has five allowed values: Reference Values, Endorsements, Evidence, Attestation Results, and Appraisal Policy, as defined in Section 8 of [RFC9334]. Note that that an Appraisal Policy may refer to the appraisal of Evidence or Attestation Results, depending on whether the consumer of the conceptual message is a Verifier or a Relying Party.

Future specifications that extend the RATS Conceptual Messages set can add new values to the cm-type using the process defined in Section 10.4.

3.2. Tag CMW

Tag CMWs derive their tag numbers from a corresponding CoAP Content-Format ID using the TN() transform defined in Appendix B of [RFC9277]. Such CBOR tag numbers are in range [1668546817, 1668612095].

The RATS conceptual message is first serialized according to the Content-Format ID and then encoded as a CBOR byte string, to which the TN-derived tag number is prepended.

The Tag CMW is defined in Figure 3 using two different macros. One for CBOR-encoded types, the other for all other types. Both macros take the CBOR tag number tn as a parameter. The tag-cm-cbor macro takes the CDDL definition of the associated conceptual message fmt as a second parameter.

```
tag-cm-cbor<tn, fmt> = #6.<tn>(bytes .cbor fmt)

tag-cm-data<tn> = #6.<tn>(bytes)
```

Figure 3: CDDL definition of the Tag CMW macros

3.2.1. How To Plug in a New Tag CMW

To plug a new Tag CMW into the CDDL defined in Section 6, the $cbor-tag type socket must be extended with a new instance of the Tag CMW macro (i.e., one of tag-cm-cbor or tag-cm-data).

For instance, if a conceptual message of type my-evidence has a TN-derived CBOR tag 1668576819, $cbor-tag would be extended as follows:

```
$cbor-tag /= tag-cm-cbor<1668576819, my-evidence>

my-evidence = {
  &(eat_nonce: 10) => bytes .size (8..64)
}
```

Instead, if a (non-CBOR) conceptual message has a TN-derived CBOR tag 1668576935, $cbor-tag would be extended as follows:

```
$cbor-tag /= tag-cm-data<1668576935>
```

Note that since this specification defines no Tag CMW, the socket is currently empty.

3.3. Collection CMW

Layered Attesters and composite devices (Sections 3.2 and 3.3 of [RFC9334]) generate Evidence that consists of multiple parts. For example, in data center servers, it is not uncommon for separate attesting environments (AE) to serve a subsection of the entire machine. One AE might measure and attest to what was booted on the main CPU, while another AE might measure and attest to what was booted on a SmartNIC plugged into a PCIe slot, and a third AE might measure and attest to what was booted on the machine’s GPU. To allow aggregation of multiple, potentially non-homogeneous evidence formats collected from different AEs, this document defines a Collection CMW as a container that holds several CMW items, each with a label that is unique within the scope of the Collection.

Although originally designed to support layered Attester and composite device use cases, the Collection CMW can be adapted for other scenarios that require the aggregation of RATS conceptual messages. For instance, Collections may be used to group Endorsements, Reference Values, Attestation Results, and more. A single Collection CMW can contain a mix of different message types, and it can also be used to carry messages related to multiple devices simultaneously.

The Collection CMW (Figure 4) is defined as a CBOR map or JSON object containing CMW values. The position of a cmw entry in the cmw-collection is not significant. Labels can be strings (or integers in the CBOR serialization) that serve as a mnemonic for different conceptual messages in the Collection.

A Collection MUST have at least one CMW entry.

The "__cmwc_t" key is reserved for associating an optional type with the overall Collection and MUST NOT be used for any purpose other than described here.

The value of the "__cmwc_t" key is either a Uniform Resource Identifier (URI) or an object identifier (OID). The OID is always absolute and never relative. The URI MUST be in the absolute form (Section 4.3 of [RFC3986]).

The "__cmwc_t" key functions similar to an EAT profile claim (see Section 4.3.2 of [RFC9711]), but at a higher level. It can be used to indicate basics like CBOR serialization and COSE algorithms just as a profile in EAT does. It provides a namespace in which the collection labels are interpreted. At the higher level, it can be used to describe the allowed CMW collection assembly (this is somewhat parallel to the way EAT profiles indicate which claims are required and/or allowed). For an example of a "__cmwc_t" that is defined for a bundle of endorsements and reference values, see Section 4.3.1 of [I-D.ietf-rats-corim].

Since the Collection CMW is recursive (a Collection CMW is itself a CMW), implementations MAY limit the allowed depth of nesting.

```
json-collection = {
  ? "__cmwc_t": ~uri / oid
  + &(label: text) => json-cmw
}

cbor-collection = {
  ? "__cmwc_t": ~uri / oid
  + &(label: (int / text)) => cbor-cmw
}
```

Figure 4: CDDL definition of the Collection CMW

3.4. Demuxing

The split in the JSON/CBOR decoding path is expected to occur via the media type or content format (see Section 10.5 and Section 10.6, respectively), or via the container context of the embedded CMW (see Section 10.1 and Section 10.2 for CWT/JWT, and Section 10.7 for X.509).

The following pseudocode illustrates how a one-byte look-ahead is sufficient to determine how to decode the remaining byte buffer.

```
func CMWTypeDemux(b []byte) CMWType {
  if len(b) == 0 {
    return Unknown
  }

  switch b[0] {
  case 0x82: // 2-elements cbor-record (w/o ind field)
  case 0x83: // 3-elements cbor-record (w/ ind field)
  case 0x9f: // start of cbor-record using indefinite-length encoding
    return CBORRecord
  case 0xda: // tag-cm-cbor (CBOR Tag in the TN range)
    return CBORTag
  case 0x5b: // ASCII '[', start of json-record
    return JSONRecord
  case 0x7b: // ASCII '{', start of json-collection
    return JSONCollection
  case 0xa0..0xbb: // CBOR map start values, start of cbor-collection
  case 0xbf:       // ditto
    return CBORCollection
  }

  return Unknown
}
```

This code is provided for informational purposes only. It is not expected that implementations will follow this demuxing strategy.

4. Cryptographic Protection of CMWs

This section highlights a number of mechanisms through which protocol designers can add data origin authentication, integrity, and, if used with a challenge-response protocol, anti-replay protection when employing CMWs. These properties must be evaluated carefully in the context of the overall security model of the protocol.

4.1. Signing CBOR CMW using COSE Sign1

A CBOR CMW can be signed using COSE [STD96]. A signed-cbor-cmw is a COSE_Sign1 with the following layout:

```
signed-cbor-cmw = [
  protected: bytes .cbor signed-cbor-cmw-protected-hdr
  unprotected: signed-cbor-cmw-unprotected-hdr
  payload: bytes .cbor cbor-cmw
  signature: bytes
]
```

The payload MUST be the CBOR-encoded Tag, Record, or Collection CMW.

```
signed-cbor-cmw-protected-hdr = {
  1 => int                            ; alg
  3 => "application/cmw+cbor" / 10000 ; cty
  * cose.label => cose.values
}

signed-cbor-cmw-unprotected-hdr = {
  * cose.label => cose.values
}

cose.label = int / text
cose.values = any

; Note: 10000 is a placeholder for the CoAP C-F codepoint corresponding
; to 'application/cmw+cbor'.  This CDDL fragment needs to be updated
; (and this note removed) once the relevant C-F codepoint has been
; assigned by IANA.
```

The protected header MUST include the signature algorithm identifier. The protected header MUST include either the content type application/cmw+cbor or the CoAP Content-Format TBD1. Other header parameters MAY be added to the header buckets, for example a kid that identifies the signing key.

4.2. Signing JSON CMW using JWS

A JSON CMW can be signed using JSON Web Signature (JWS) [RFC7515]. A signed-json-cmw is a JWS object with the following layout:

```
signed-json-cmw = {
  "protected": text .b64u (text .json signed-json-cmw-protected-hdr)
  ? "header": text .b64u (text .json signed-json-cmw-unprotected-hdr)
  "payload": text .b64u (text .json json-cmw)
  "signature": text .b64u bytes
}
```

The payload MUST be the JSON-encoded Record, or Collection CMW.

```
signed-json-cmw-protected-hdr = {
  "alg": text
  "cty": "application/cmw+json"
  * text => text
}

signed-json-cmw-unprotected-hdr = {
  * text => text
}
```

The protected header MUST include the signature algorithm identifier. The protected header MUST include the content type application/cmw+json. Other header parameters MAY be added to the header buckets, for example a kid that identifies the signing key.

For clarity, the above uses the Flattened JSON Serialization (Section 7.2.2 of [RFC7515]). However, the Compact Serialization (Section 3.1 of [RFC7515]) can also be used.

4.3. Transporting CMW in COSE and JOSE Web Tokens

To facilitate the embedding of CMWs in CBOR-based protocols and web APIs, this document defines two "cmw" claims for use with JSON Web Tokens (JWT) and CBOR Web Tokens (CWT).

The definitions for these claims can be found in Section 10.2 and Section 10.1, respectively.

4.3.1. Encoding Requirements

A Collection CMW carried in a "cmw" JWT claim MUST be a json-collection. A Collection CMW carried in a "cmw" CWT claim MUST be a cbor-collection.

A Record CMW carried in a "cmw" JWT claim MUST be a json-record. A Record CMW carried in a "cmw" CWT claim MUST be a cbor-record.

4.4. Transporting CMW in PKIX Formats

CMW may need to be transported in PKIX formats, such as Certificate Signing Requests (CSRs) or in X.509 Certificates and Certificate Revocation Lists (CRLs).

The use of CMW in CSRs is documented in [I-D.ietf-lamps-csr-attestation], while one of the possible applications in X.509 Certificates and CRLs is detailed in Section 6.1 of [DICE-arch].

This section outlines the CMW extension designed to carry CMW objects. Section 8 discusses some privacy considerations related to the transport of CMW in X.509 formats.

The CMW extension MAY be included in X.509 Certificates, CRLs [RFC5280], and CSRs.

The CMW extension MUST be identified by the following object identifier:

```
id-pe-cmw  OBJECT IDENTIFIER ::=
        { iso(1) identified-organization(3) dod(6) internet(1)
          security(5) mechanisms(5) pkix(7) id-pe(1) 35 }
```

This extension SHOULD NOT be marked critical. In cases where the wrapped Conceptual Message is essential for granting resource access, and there is a risk that legacy relying parties would bypass crucial controls, it is acceptable to mark the extension as critical.

The CMW extension MUST have the following syntax:

```
CMW ::= CHOICE {
    json UTF8String,
    cbor OCTET STRING
}
```

The CMW MUST include the serialized CMW object in either JSON or CBOR format, utilizing the appropriate CHOICE entry.

The DER-encoded CMW is the value of the OCTET STRING for the extnValue field of the extension.

4.4.1. ASN.1 Module

This section provides an ASN.1 module [X.680] for the CMW extension, following the conventions established in [RFC5912] and [RFC6268].

```
CMWExtn
  { iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanisms(5) pkix(7) id-mod(0)
    id-mod-cmw-extn(TBD) }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

IMPORTS
  EXTENSION
  FROM PKIX-CommonTypes-2009  -- RFC 5912
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkixCommon-02(57) } ;

-- CMW Extension

ext-CMW EXTENSION ::= {
  SYNTAX CMW
  IDENTIFIED BY id-pe-cmw }

-- CMW Extension OID

id-pe-cmw  OBJECT IDENTIFIER  ::=
   { iso(1) identified-organization(3) dod(6) internet(1)
     security(5) mechanisms(5) pkix(7) id-pe(1) 35 }

-- CMW Extension Syntax

CMW ::= CHOICE {
    json UTF8String,
    cbor OCTET STRING
}

END
```

4.4.2. Compatibility with DICE ConceptualMessageWrapper

Section 6.1.8 of [DICE-arch] specifies the ConceptualMessageWrapper (CMW) format and its corresponding object identifier. The CMW format outlined in [DICE-arch] permits only a subset of the CMW grammar defined in this document. In particular, the Collection format cannot be encoded using DICE CMWs.

5. Examples

The (equivalent) examples in Section 5.1, Section 5.2, and Section 5.3 assume that the Media-Type-Name application/vnd.example.rats-conceptual-msg has been registered alongside a corresponding CoAP Content-Format ID 30001. The CBOR tag 1668576935 is derived applying the TN() transform as described in Section 3.2.

All the examples focus on the wrapping aspects. The wrapped messages are not instances of real Conceptual Messages.

5.1. JSON-encoded Record

```
[
  "application/vnd.example.rats-conceptual-msg",
  "I0faVQ"
]
```

5.2. CBOR-encoded Record

```
[
  30001,
  h'2347da55'
]
```

with the following wire representation:

```
82             # array(2)
   19 7531     # unsigned(30001)
   44          # bytes(4)
      2347da55 # "#G\xDAU"
```

Note that a Media-Type-Name can also be used with the CBOR-encoded Record form, for example if it is known that the receiver cannot handle CoAP Content-Formats, or (unlike the case in point) if a CoAP Content-Format ID has not been registered.

```
[
  "application/vnd.example.rats-conceptual-msg",
  h'2347da55'
]
```

5.3. CBOR-encoded Tag CMW

```
1668576935(h'2347da55')
```

with the following wire representation:

```
da 637476a7    # tag(1668576935)
   44          # bytes(4)
      2347da55 # "#G\xDAU"
```

5.4. CBOR-encoded Record with explicit CM indicator

This is an example of a signed CoRIM (Concise Reference Integrity Manifest) [I-D.ietf-rats-corim] with an explicit ind value of 0b0000_0011 (3), indicating that the wrapped message contains both Reference Values and Endorsements.

```
[
  "application/rim+cose",
  h'd28440a044d901f5a040',
  3
]
```

with the following wire representation:

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

83                                      # array(3)
   74                                   # text(20)
      6170706c69636174696f6e2f72696d2b636f7365 # "application/rim+\
                                                           cose"
   4a                                   # bytes(10)
      d28440a044d901f5a040              # "メ A \xA0D\xD9\u0001\xF5\xA0\
                                                             @"
   03                                   # unsigned(3)
```

5.5. CBOR-encoded Collection

The following example is a CBOR-encoded Collection CMW that assembles conceptual messages from three attesters: Evidence for attesters A and B and Attestation Results for attester C. It is given an explicit "__cmwc_t" using the URI form.

```
{
  "__cmwc_t": "tag:example.com,2024:composite-attester",
  / attester A / 0: [
    30001,
    h'2347da55',
    4
  ],
  / attester B / 1: 1668576935(h'2347da55'),
  / attester C / 2: [
    "application/eat+jwt",
    h'2e2e2e',
    8
  ]
}
```

5.6. JSON-encoded Collection

The following example is a JSON-encoded Collection CMW that assembles Evidence from two attesters.

```
{
  "__cmwc_t": "tag:example.com,2024:another-composite-attester",
  "attester A": [
    "application/eat-ucs+json",
    "e30K",
    4
  ],
  "attester B": [
    "application/eat-ucs+cbor",
    "oA",
    4
  ]
}
```

5.7. Use in JWT

The following example shows the use of the "cmw" JWT claim to transport a Collection CMW in a JWT Claims Set [RFC7519]:

```
{
  "cmw": {
    "__cmwc_t": "tag:example.com,2024:another-composite-attester",
    "attester A": [
      "application/eat-ucs+json",
      "e30K",
      4
    ],
    "attester B": [
      "application/eat-ucs+cbor",
      "oA",
      4
    ]
  },
  "iss": "evidence collection daemon",
  "exp": 1300819380
}
```

6. Collected CDDL

This appendix contains all the CDDL definitions included in this specification.

```
start = cmw

cmw = json-cmw / cbor-cmw

json-cmw = json-record / json-collection
cbor-cmw = cbor-record / cbor-collection / $cbor-tag

json-record = [
  type: media-type
  value: base64url-string
  ? ind: uint .bits cm-type
]

cbor-record = [
  type: coap-content-format-type / media-type
  value: bytes
  ? ind: uint .bits cm-type
]

tag-cm-cbor<tn, fmt> = #6.<tn>(bytes .cbor fmt)

tag-cm-data<tn> = #6.<tn>(bytes)

json-collection = {
  ? "__cmwc_t": ~uri / oid
```

```
    + &(label: text) => json-cmw
  }

  cbor-collection = {
    ? "__cmwc_t": ~uri / oid
    + &(label: (int / text)) => cbor-cmw
  }

  media-type = text .abnf ("Content-Type" .cat Content-Type-ABNF)
  base64url-string = text .regexp "[A-Za-z0-9_-]+"

  coap-content-format-type = uint .size 2

  oid = text .regexp "([0-2])((\\.0)|(\\.[1-9][0-9]*))*"

  cm-type = &(
    reference-values: 0
    endorsements: 1
    evidence: 2
    attestation-results: 3
    appraisal-policy: 4
  )

  Content-Type-ABNF = ’

  Content-Type   = Media-Type-Name *( *SP ";" *SP parameter )
  parameter      = token "=" ( token / quoted-string )

  token          = 1*tchar
  tchar          = "!" / "#" / "$" / "%" / "&" / "\’" / "*"
                 / "+" / "-" / "." / "^" / "_" / "‘" / "|" / "~"
                 / DIGIT / ALPHA
  quoted-string  = %x22 *( qdtext / quoted-pair ) %x22
  qdtext         = SP / %x21 / %x23-5B / %x5D-7E
  quoted-pair    = "\\" ( SP / VCHAR )

  Media-Type-Name = type-name "/" subtype-name

  type-name = restricted-name
  subtype-name = restricted-name

  restricted-name = restricted-name-first *126restricted-name-chars
  restricted-name-first  = ALPHA / DIGIT
  restricted-name-chars  = ALPHA / DIGIT / "!" / "#" /
                           "$" / "&" / "-" / "^" / "_"
  restricted-name-chars =/ "." ; Characters before first dot always
                               ; specify a facet name
  restricted-name-chars =/ "+" ; Characters after last plus always
```

```
                                ; specify a structured syntax suffix

   DIGIT     =  %x30-39           ; 0 - 9
   POS-DIGIT =  %x31-39           ; 1 - 9
   ALPHA     =  %x41-5A / %x61-7A ; A - Z / a - z
   SP        =  %x20
   VCHAR     =  %x21-7E           ; printable ASCII (no SP)
   '


   signed-cbor-cmw = [
     protected: bytes .cbor signed-cbor-cmw-protected-hdr
     unprotected: signed-cbor-cmw-unprotected-hdr
     payload: bytes .cbor cbor-cmw
     signature: bytes
   ]

   signed-cbor-cmw-protected-hdr = {
     1 => int                           ; alg
     3 => "application/cmw+cbor" / 10000 ; cty
     * cose.label => cose.values
   }

   signed-cbor-cmw-unprotected-hdr = {
     * cose.label => cose.values
   }

   cose.label = int / text
   cose.values = any


   signed-json-cmw = {
     "protected": text .b64u (text .json signed-json-cmw-protected-hdr)
     ? "header": text .b64u (text .json signed-json-cmw-unprotected-hdr)
     "payload": text .b64u (text .json json-cmw)
     "signature": text .b64u bytes
   }

   signed-json-cmw-protected-hdr = {
     "alg": text
     "cty": "application/cmw+json"
     * text => text
   }

   signed-json-cmw-unprotected-hdr = {
     * text => text
   }
```

7. Implementation Status

// RFC Editor: Please remove the entire section before publication, as well as the reference to RFC 7942.

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".

7.1. Project Veraison

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

The organisation hosts two libraries which allow encoding, decoding, and manipulation of CMW payloads: one for the Golang ecosystem (https://github.com/veraison/cmw), and one for Rust (https://github.com/veraison/rust-cmw). These implementations cover all the features presented in this draft. The maturity level is alpha. The license is Apache 2.0. The developers can be contacted on the Zulip channel: https://veraison.zulipchat.com/#narrow/stream/383526-CMW/.

8. Privacy Considerations

The privacy considerations outlined in Section 11 of [RFC9334] are fully applicable. In particular, when a CMW contains Personally Identifying Information (PII), which is the case for Evidence and sometimes for other conceptual messages as well, care must be taken to prevent unintended recipients from accessing it. Generally, utilizing secure channels between the parties exchanging CMWs can help address or mitigate these concerns. A specific scenario arises when a public key certificate is issued based on Evidence information provided by the certificate requestor to the issuing Certification Authority (CA). For instance, an individual seeking a publicly-trusted code signing certificate may be willing to disclose the details of the hardware where their code signing keys are stored (e.g., HSM model, patch level, etc.). However, they likely do not want this information to be publicly accessible. Applications that intend to publicly "broadcast" Evidence claims received from a third party via X.509 Certificates should define a Certificate Practices Statement [RFC3647] that clearly specifies the circumstances under which the CA can include such data in the issued certificate. Note that the aforementioned consideration does not apply to cases where X.509 Certificates are explicitly designed as a security envelope for Evidence claims, such as in DICE [DICE-arch].

9. Security Considerations

The security considerations discussed in Section 12.2 of [RFC9334] concerning the protection of conceptual messages are fully applicable. The following subsections provide further elaboration on these points, particularly in relation to Collection CMWs.

9.1. CMW Protection

CMW Records, Tags, and Collections alone do not offer authenticity, integrity protection, or confidentiality. It is the responsibility of the designer for each use case to determine the necessary security properties and implement them accordingly.

RATS conceptual messages are typically secured using cryptography. If the messages are already protected, no additional security requirements are imposed by this encapsulation. If an adversary attempts to modify the payload encapsulation, it will result in incorrect processing of the encapsulated message, leading to an error. If the messages are not protected, additional security must be added at a different layer. For example, a cbor-record containing an Unprotected CWT Claims Set (UCCS) [RFC9781] can be signed as described in Section 4.1.

Section 4 describes a number of methods that can be used to add cryptographic protection to CMW.

9.2. Using Collection CMWs for Evidence of Composite or Layered Devices

When a Collection CMW is used to encapsulate Evidence for composite or layered attestation of a single device, all Evidence messages within the CMW MUST be cryptographically bound together to prevent an attacker from replacing Evidence from a compromised device with that from a non-compromised device. If the Collection CMW is not protected from tampering by external security measures (such as object security primitives) or internal mechanisms (such as intra-item binding), an attacker could manipulate the Collection’s contents to deceive the Verifier into accepting bogus Evidence as genuine.

Authenticity and integrity protection is expected to be provided by the underlying attestation technology. For example, key material used to sign/bind an entire Collection CMW should be an attestation key, handled as described in Section 12.1 of [RFC9334]. The binding does not necessarily have to be a signature over the Collection CMW, it might also be achieved through identifiers, linking claims (e.g., nonces) across CMW collection items, signing or hashing between the members of the Collection. It is the responsibility of the Attester who creates the Collection CMW to ensure that the contents of the Collection are integrity-protected.

9.3. Integrating CMW into Protocols

When CMW is integrated into some hosting protocol (for example, attested CSR [I-D.ietf-lamps-csr-attestation] or attested TLS [I-D.fossati-tls-attestation] [I-D.fossati-tls-exported-attestation]), it is up to that hosting protocol to describe how CMW is intended to be used and how it fits into the overall security model.

Such an analysis should consider the types of conceptual messages allowed, including the permitted combinations, the protection requirements, the interface with the hosting protocol, and any other security-relevant aspect arising from the interaction between the CMW assembly and the hosting protocol.

10. IANA Considerations

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

// RFC Editor: This document uses the CPA (code point allocation) convention described in [I-D.bormann-cbor-draft-numbers]. For each usage of the term "CPA", please remove the prefix "CPA" from the indicated value and replace the residue with the value assigned by IANA; perform an analogous substitution for all other occurrences of the prefix "CPA" in the document. Finally, please remove this note.

10.1. CWT cmw Claim Registration

IANA is requested to add a new cmw claim to the "CBOR Web Token (CWT) Claims" registry [IANA.cwt] as follows:

* Claim Name: cmw
* Claim Description: A RATS Conceptual Message Wrapper
* JWT Claim Name: cmw
* Claim Key: CPA299
* Claim Value Type(s): CBOR map, CBOR array, or CBOR tag
* Change Controller: IETF
* Specification Document(s): Section 3.1, Section 3.3 and Section 3.2 of RFCthis

10.2. JWT cmw Claim Registration

IANA is requested to add a new cmw claim to the "JSON Web Token Claims" registry of the "JSON Web Token (JWT)" registry group [IANA.jwt] as follows:

* Claim Name: cmw
* Claim Description: A RATS Conceptual Message Wrapper
* Change Controller: IETF
* Specification Document(s): Section 3.1 and Section 3.3 of RFCthis

10.3. +jws Structured Syntax Suffix

IANA is requested to register the +jws structured syntax suffix in the "Structured Syntax Suffixes" registry [IANA.media-type-structured-suffix] in the manner described in [RFC6838], which can be used to indicate that the media type is encoded as JSON Web Signature (JWS) [RFC7515].

10.3.1. Registry Contents

Name: JSON Web Signature (JWS)
+suffix: +jws
References: [RFC7515]
Encoding Considerations: binary; values are represented as a JSON Object or as a series of base64url-encoded values each separated from the next by a single period ('.') character.
Interoperability Considerations: n/a
Fragment Identifier Considerations: n/a
Security Considerations: See Section 10 of [RFC7515]
Contact: RATS WG mailing list (rats@ietf.org), or IETF Security Area (saag@ietf.org)
Author/Change Controller: Remote ATtestation ProcedureS (RATS) Working Group. The IETF has change control over this registration.

10.4. RATS Conceptual Message Wrapper (CMW) Indicators Registry

This specification defines a new "RATS Conceptual Message Wrapper (CMW) Indicators" registry, with "IETF Review" policy (Section 4.8 of [BCP26]).

The objective is to register CMW Indicator values for all RATS Conceptual Messages (see Section 8 of [RFC9334]).

This registry is to be added to the Remote Attestation Procedures (RATS) registry group at [IANA.rats].

Indicator values should be added in ascending order, with no gaps between them.

Acceptable values correspond to the RATS conceptual messages defined by the RATS architecture [RFC9334] and any updates to it.

10.4.1. Structure of Entries

Each entry in the registry must include:

Indicator value:
A number corresponding to the bit position in the ind bitmap (Section 3.1).

Conceptual Message name:
A text string describing the RATS conceptual message this indicator corresponds to.

Reference:

A reference to a document, if available, or the registrant.

The initial registrations for the registry are detailed in Table 1.

Indicator value	Conceptual Message name	Reference
0	Reference Values	Section 3.1.1 of RFCthis
1	Endorsements	Section 3.1.1 of RFCthis
2	Evidence	Section 3.1.1 of RFCthis
3	Attestation Results	Section 3.1.1 of RFCthis
4	Appraisal Policy	Section 3.1.1 of RFCthis
5-31	Unassigned	

Table 1: CMW Indicators Registry Initial Contents

10.4.2. Provisional Registration

// RFC Editor: Please remove this section before publication, as well as the reference to the provisional CMW Indicators registry.

Before the creation of the registry by IANA, new codepoints can be added to the provisional CMW Indicators registry (https://github.com/ietf-rats-wg/draft-ietf-rats-msg-wrap/blob/main/provisional/cmw-indicators-registry.md) by following the documented procedure.

Table 1 will be regularly updated, prior to publication of this specification as an RFC, to match the contents of the provisional registry.

The provisional registry will be discontinued once IANA establishes the permanent registry, which is expected to coincide with the publication of the current document.

10.5. Media Types

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

Name	Template	Reference
cmw+cbor	application/cmw+cbor	Section 3.1, Section 3.2 and Section 3.3 of RFCthis
cmw+json	application/cmw+json	Section 3.1 and Section 3.3 of RFCthis
cmw+cose	application/cmw+cose	Section 4.1 of RFCthis
cmw+jws	application/cmw+jws	Section 4.2 of RFCthis

Table 2: CMW Media Types

10.5.1. application/cmw+cbor

Type name: application
Subtype name: cmw+cbor
Required parameters: n/a
Optional parameters: cmwc_t (Collection CMW type in string format. OIDs must use the dotted-decimal notation. The parameter value is case-insensitive. It must not be used for CMW that are not Collections.)
Encoding considerations: binary (CBOR)
Security considerations: Section 9 of RFCthis
Interoperability considerations: n/a
Published specification: RFCthis
Applications that use this media type: Attesters, Verifiers, Endorsers and Reference-Value providers, Relying Parties that need to transfer CMW payloads over HTTP(S), CoAP(S), and other transports.
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

10.5.2. application/cmw+json

Type name: application
Subtype name: cmw+json
Required parameters: n/a
Optional parameters: cmwc_t (Collection CMW type in string format. OIDs must use the dotted-decimal notation. The parameter value is case-insensitive. It must not be used for CMW that are not Collections.)
Encoding considerations: binary (JSON is UTF-8-encoded text)
Security considerations: Section 9 of RFCthis
Interoperability considerations: n/a
Published specification: RFCthis
Applications that use this media type: Attesters, Verifiers, Endorsers and Reference-Value providers, Relying Parties that need to transfer CMW payloads over HTTP(S), CoAP(S), and other transports.
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

10.5.3. application/cmw+cose

Type name: application
Subtype name: cmw+cose
Required parameters: n/a
Optional parameters: cmwc_t (Collection CMW type in string format. OIDs must use the dotted-decimal notation. The parameter value is case-insensitive. It must not be used for CMW that are not Collections.) Note that the cose-type parameter is explicitly not supported, as it is understood to be "cose-sign1".
Encoding considerations: binary (CBOR)
Security considerations: Section 9 of RFCthis
Interoperability considerations: n/a
Published specification: RFCthis
Applications that use this media type: Attesters, Verifiers, Endorsers and Reference-Value providers, Relying Parties that need to transfer CMW payloads over HTTP(S), CoAP(S), and other transports.
Fragment identifier considerations: n/a
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

10.5.4. application/cmw+jws

Type name: application
Subtype name: cmw+jws
Required parameters: n/a
Optional parameters: cmwc_t (Collection CMW type in string format. OIDs must use the dotted-decimal notation. The parameter value is case-insensitive. It must not be used for CMW that are not Collections.)
Encoding considerations: 8bit; values are represented as a JSON Object or as a series of base64url-encoded values each separated from the next by a single period ('.') character.
Security considerations: Section 9 of RFCthis
Interoperability considerations: n/a
Published specification: RFCthis
Applications that use this media type: Attesters, Verifiers, Endorsers and Reference-Value providers, Relying Parties that need to transfer CMW payloads over HTTP(S), CoAP(S), and other transports.
Fragment identifier considerations: n/a
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

10.6. CoAP Content-Formats

IANA is requested to register the following Content-Format IDs in the "CoAP Content-Formats" registry, within the "Constrained RESTful Environments (CoRE) Parameters" registry group [IANA.core-parameters]:

Content-Type	Content Coding	ID	Reference
application/ cmw+cbor	-	TBD1	Section 3.1, Section 3.2 and Section 3.3 of RFCthis
application/ cmw+json	-	TBD2	Section 3.1 and Section 3.3 of RFCthis
application/ cmw+cose	-	TBD3	Section 4.1 of RFCthis
application/ cmw+jws	-	TBD4	Section 4.2 of RFCthis

Table 3: New CoAP Content Formats

If possible, TBD1, TBD2, TBD3 and TBD4 should be assigned in the 256..9999 range.

10.6.1. Registering new CoAP Content-Formats for Parameterized CMW Media Types

New CoAP Content-Formats can be created based on parameterized instances of the application/cmw+json, application/cmw+cbor, application/cmw+cose and application/cmw+jws media types.

When assigning a new CoAP Content-Format ID for a CMW media type that utilizes the cmwc_t parameter, the registrar must check (directly or through the Designated Expert) that:

* The corresponding CMW is a Collection (Section 3.3), and

* The cmwc_t value is either a (non-relative) OID or an absolute URI.

10.6.2. RFC9277 CBOR Tags

// RFC Editor: Once IANA has allocated TBD1..TBD4, please replace the placeholders in the first column of Table 4 with the values computed using the TN() formula in Appendix B of [RFC9277]. Similarly, replace the macro parameters in Figure 5.

Registering the CoAP Content-Formats listed in Table 3 automatically allocates CBOR Tags in the range [1668546817, 1668612095], using the TN() transform defined in Appendix B of [RFC9277]. The allocated CBOR Tag numbers and the corresponding data items are listed in Table 4.

Tag Number	Tag Content
TN(TBD1)	bytes .cbor cbor-cmw
TN(TBD2)	bytes-wrapped json-cmw
TN(TBD3)	bytes .cbor signed-cbor-cmw
TN(TBD4)	bytes-wrapped signed-json-cmw or equivalent using JWS Compact Serialization (Section 3.1 of [RFC7515])

Table 4: TN-derived CBOR Tags

Figure 5 extends the $cbor-tag socket defined in Section 3.2 to add the definitions of the associated Tag CMWs. Note that CMWs in Tag and Record form are excluded from the productions. This is because they can already be represented as a CMW, so the extra wrapping would be redundant.

```
$cbor-tag /= tag-cm-cbor<1668547091, cbor-collection>
$cbor-tag /= tag-cm-cbor<1668547092, signed-cbor-cmw>
$cbor-tag /= tag-cm-data<1668547093> ; bytes(cmw+json collection)
$cbor-tag /= tag-cm-data<1668547094> ; bytes(cmw+jws)
```

Figure 5: Tag CMW definitions

10.7. New SMI Numbers Registrations

IANA has assigned an object identifier (OID) for the CMW extension defined in Section 4.4 in the "SMI Security for PKIX Certificate Extension" registry of the "SMI Numbers" [IANA.smi-numbers] registry group as follows:

Decimal	Description	References
35	id-pe-cmw	Section 4.4 of RFCthis

Table 5: New CMW Extension OID

IANA is requested to assign an object identifier (OID) for the ASN.1 Module defined in Section 4.4.1 in the "SMI Security for PKIX Module Identifier" registry of the "SMI Numbers" [IANA.smi-numbers] registry group:

Decimal	Description	References
TBD	id-mod-cmw-extn	Section 4.4.1 of RFCthis

Table 6: New ASN.1 Module OID

11. References

11.1. Normative References

[BCP26] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

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

[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>.

[IANA.media-type-structured-suffix] IANA, "Structured Syntax Suffixes", <https://www.iana.org/assignments/media-type-structured-suffix>.

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

[IANA.rats]
IANA, "Remote Attestation Procedures (RATS)",
<https://www.iana.org/assignments/rats>.

[IANA.smi-numbers]
IANA, "Structure of Management Information (SMI) Numbers
(MIB Module Registrations)",
<https://www.iana.org/assignments/smi-numbers>.

[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>.

[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/rfc/rfc3986>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data
Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006,
<https://www.rfc-editor.org/rfc/rfc4648>.

[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>.

[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/rfc/rfc6838>.

[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>.

[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/rfc/rfc7515>.

[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>.

[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>.

[RFC9277] Richardson, M. and C. Bormann, "On Stable Storage for Items in Concise Binary Object Representation (CBOR)", RFC 9277, DOI 10.17487/RFC9277, August 2022, <https://www.rfc-editor.org/rfc/rfc9277>.

[STD90] 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/rfc/rfc8259>.

[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>.

[X.680] International Telephone and Telegraph Consultative Committee, "Specification of Abstract Syntax Notation One (ASN.1): Specification of Basic Notation", CCITT Recommendation X.680, July 1994.

11.2. Informative References

[DICE-arch] Trusted Computing Group, "DICE Attestation Architecture", January 2024, <https://trustedcomputinggroup.org/wp-content/uploads/DICE-Attestation-Architecture-Version-1.1-Revision-18_pub.pdf>.

[I-D.bormann-cbor-draft-numbers]
Bormann, C., "Managing CBOR codepoints in Internet-Drafts", Work in Progress, Internet-Draft, draft-bormann-cbor-draft-numbers-06, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-bormann-cbor-draft-numbers-06>.

[I-D.fossati-tls-attestation]
Tschofenig, H., Sheffer, Y., Howard, P., Mihalcea, I., Deshpande, Y., Niemi, A., and T. Fossati, "Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", Work in Progress, Internet-Draft, draft-fossati-tls-attestation-09, 30 April 2025, <https://datatracker.ietf.org/doc/html/draft-fossati-tls-attestation-09>.

[I-D.fossati-tls-exported-attestation]
Fossati, T., Sardar, M. U., Reddy.K, T., Sheffer, Y., Tschofenig, H., and I. Mihalcea, "Remote Attestation with Exported Authenticators", Work in Progress, Internet-Draft, draft-fossati-tls-exported-attestation-02, 3 July 2025, <https://datatracker.ietf.org/doc/html/draft-fossati-tls-exported-attestation-02>.

[I-D.ietf-lamps-csr-attestation]
Ounsworth, M., Tschofenig, H., Birkholz, H., Wiseman, M., and N. Smith, "Use of Remote Attestation with Certification Signing Requests", Work in Progress, Internet-Draft, draft-ietf-lamps-csr-attestation-21, 5 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-csr-attestation-21>.

[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-09, 15 August 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-ar4si-09>.

[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-09, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-corim-09>.

[RFC3647] Chokhani, S., Ford, W., Sabett, R., Merrill, C., and S. Wu, "Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework", RFC 3647, DOI 10.17487/RFC3647, November 2003, <https://www.rfc-editor.org/rfc/rfc3647>.

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <https://www.rfc-editor.org/rfc/rfc5912>.

[RFC6268] Schaad, J. and S. Turner, "Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)", RFC 6268, DOI 10.17487/RFC6268, July 2011, <https://www.rfc-editor.org/rfc/rfc6268>.

[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>.

[RFC9193] Kerテ、nen, A. and C. Bormann, "Sensor Measurement Lists (SenML) Fields for Indicating Data Value Content-Format", RFC 9193, DOI 10.17487/RFC9193, June 2022, <https://www.rfc-editor.org/rfc/rfc9193>.

[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>.

[RFC9711] Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "The Entity Attestation Token (EAT)", RFC 9711, DOI 10.17487/RFC9711, April 2025, <https://www.rfc-editor.org/rfc/rfc9711>.

[RFC9781] Birkholz, H., O'Donoghue, J., Cam-Winget, N., and C. Bormann, "A Concise Binary Object Representation (CBOR) Tag for Unprotected CBOR Web Token Claims Sets (UCCS)", RFC 9781, DOI 10.17487/RFC9781, May 2025, <https://www.rfc-editor.org/rfc/rfc9781>.

[RFC9782] Lundblade, L., Birkholz, H., and T. Fossati, "Entity Attestation Token (EAT) Media Types", RFC 9782, DOI 10.17487/RFC9782, May 2025, <https://www.rfc-editor.org/rfc/rfc9782>.

[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>.

Appendix A. Registering and Using CMWs

Figure 6 describes the registration preconditions for using CMWs in either Record CMW or Tag CMW forms. When using Collection CMW, the preconditions apply for each entry in the Collection.

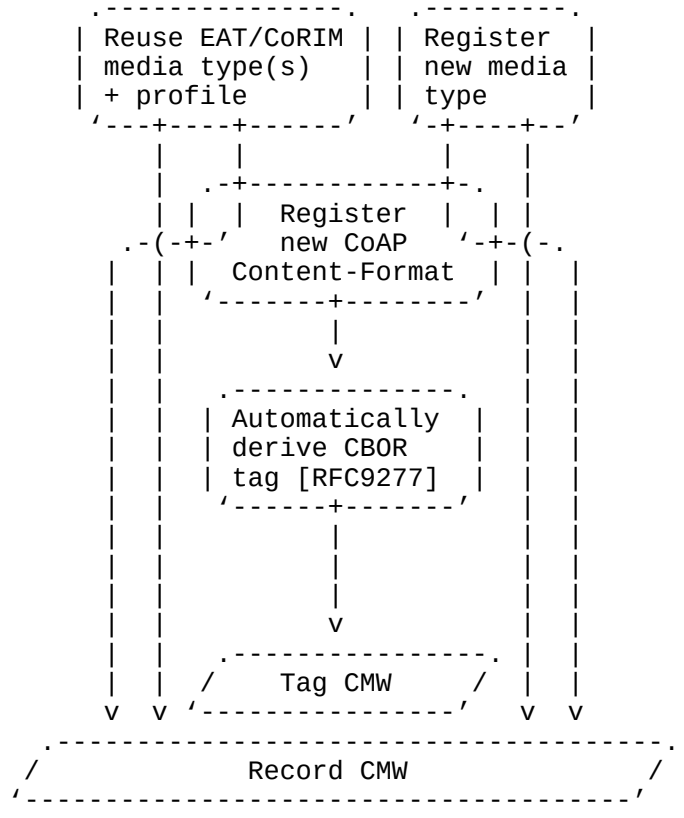

Figure 6: How To Create a CMW

Appendix B. Open Issues

The list of currently open issues for this documents can be found at https://github.com/thomas-fossati/draft-ftbs-rats-msg-wrap/issues.

// RFC Editor: please remove before publication.

Acknowledgments

The authors would like to thank Alexey Melnikov, Benjamin Schwartz, Brian Campbell, Carl Wallace, Carsten Bormann, Christian Amsテシss, Dave Thaler, Deb Cooley, Ionuネ Mihalcea, Michael B. Jones, Mike Ounsworth, Michael StJohns, Mohit Sethi, Paul Howard, Peter Yee, Russ Housley, Steven Bellock, Tim Bray, Tom Jones, and Usama Sardar for their reviews and suggestions.

The definition of a Collection CMW has been modelled on a proposal originally made by Simon Frost for an EAT-based Evidence collection type. The Collection CMW aims at superseding it by generalizing the allowed Evidence formats.

Contributors

Laurence Lundblade
Security Theory LLC
Email: lgl@securitytheory.com

Laurence made significant contributions to enhancing the security requirements and considerations for Collection CMWs.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Email: henk.birkholz@ietf.contact

Ned Smith
Independent
Email: ned.smith.ietf@outlook.com

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

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

Dionna Glaze
Google LLC
Email: dionnaglaze@google.com