

Remote Attestation Procedures  
Internet-Draft  
Intended status: Standards Track  
Expires: 12 January 2023

H. Birkholz  
Fraunhofer SIT  
T. Fossati  
Y. Deshpande  
arm  
N. Smith  
Intel  
W. Pan  
Huawei Technologies  
11 July 2022

Concise Reference Integrity Manifest  
draft-birkholz-rats-corim-03

## Abstract

Remote Attestation Procedures (RATS) enable Relying Parties to assess the trustworthiness of a remote Attester and therefore to decide whether to engage in secure interactions with it. Evidence about trustworthiness can be rather complex and it is deemed unrealistic that every Relying Party is capable of the appraisal of Evidence. Therefore that burden is typically offloaded to a Verifier. In order to conduct Evidence appraisal, a Verifier requires not only fresh Evidence from an Attester, but also trusted Endorsements and Reference Values from Endorsers and Reference Value Providers, such as manufacturers, distributors, or device owners. This document specifies Concise Reference Integrity Manifests (CoRIM) that represent Endorsements and Reference Values in CBOR format. Composite devices or systems are represented by a collection of Concise Module Identifiers (CoMID) and Concise Software Identifiers (CoSWID) bundled in a CoRIM document.

## Discussion Venues

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

Discussion of this document takes place on the RATS Working Group mailing list ([rats@ietf.org](mailto: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/ietf-rats/draft-birkholz-rats-corim>.

## 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 12 January 2023.

#### Copyright Notice

Copyright (c) 2022 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	4
1.1. Terminology and Requirements Language	4
1.2. CDDL Typographical Conventions	4
1.3. Common Types	5
1.3.1. Non-Empty	5
1.3.2. Entity	5
1.3.3. Validity	6
1.3.4. UUID	7
1.3.5. UEID	7
1.3.6. OID	7
1.3.7. Tagged Integer Type	7
1.3.8. Hash Entry	7
2. CoRIM	8
2.1. CoRIM Map	8
2.1.1. Identity	9
2.1.2. Tags	9
2.1.3. Locator Map	9
2.1.4. Profile Types	10
2.1.5. Entities	10
2.2. Signed CoRIM	10

2.2.1.	Protected Header Map . . . . .	11
2.2.2.	Meta Map . . . . .	11
2.2.2.1.	Signer Map . . . . .	12
3.	CoMID . . . . .	12
3.1.	Structure . . . . .	12
3.1.1.	Tag Identity . . . . .	13
3.1.1.1.	Tag ID . . . . .	13
3.1.1.2.	Tag Version . . . . .	14
3.1.2.	Entities . . . . .	14
3.1.3.	Linked Tag . . . . .	15
3.1.4.	Triples . . . . .	15
3.1.4.1.	Common Types . . . . .	16
3.1.4.1.1.	Environment . . . . .	16
3.1.4.1.2.	Class . . . . .	17
3.1.4.1.3.	Instance . . . . .	18
3.1.4.1.4.	Group . . . . .	18
3.1.4.1.5.	Measurements . . . . .	18
3.1.4.1.5.1.	Measurement Keys . . . . .	19
3.1.4.1.5.2.	Measurement Values . . . . .	19
3.1.4.1.5.3.	Version . . . . .	20
3.1.4.1.5.4.	Security Version Number . . . . .	21
3.1.4.1.5.5.	Flags . . . . .	21
3.1.4.1.5.6.	Raw Values Types . . . . .	22
3.1.4.1.5.7.	Address Types . . . . .	22
3.1.4.1.6.	Crypto Keys . . . . .	22
3.1.4.1.7.	Domain Types . . . . .	23
3.1.4.2.	Reference Values Triple . . . . .	23
3.1.4.3.	Endorsed Values Triple . . . . .	24
3.1.4.4.	Device Identity Triple . . . . .	24
3.1.4.5.	Attestation Keys Triple . . . . .	24
3.1.4.6.	Domain Dependency Triple . . . . .	24
3.1.4.7.	Domain Membership Triple . . . . .	25
3.1.4.8.	CoMID-CoSWID Linking Triple . . . . .	25
3.2.	Extensibility . . . . .	25
4.	Implementation Status . . . . .	26
4.1.	Veraison . . . . .	26
5.	Security and Privacy Considerations . . . . .	27
6.	IANA Considerations . . . . .	27
6.1.	New COSE Header Parameters . . . . .	27
6.2.	New CBOR Tags . . . . .	27
6.3.	New CoRIM Registries . . . . .	27
6.4.	New CoMID Registries . . . . .	27
6.5.	New Media Types . . . . .	27
6.5.1.	corim-signed+cbor . . . . .	28
6.5.2.	corim-unsigned+cbor . . . . .	28
6.6.	CoAP Content-Formats Registration . . . . .	29
7.	References . . . . .	29
7.1.	Normative References . . . . .	29

7.2. Informative References . . . . .	31
Appendix A. Full CoRIM CDDL . . . . .	32
Acknowledgments . . . . .	32
Authors' Addresses . . . . .	32

## 1. Introduction

// Content missing. Tracked at: <https://github.com/ietf-rats/draft-birkholz-rats-corim/issues/86>

### 1.1. Terminology and Requirements Language

This document uses terms and concepts defined by the RATS architecture. For a complete glossary see Section 4 of [I-D.ietf-rats-architecture].

The terminology from CBOR [STD94], CDDL [RFC8610] and COSE [RFC8152] 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.

### 1.2. CDDL Typographical Conventions

The CDDL definitions in this document follow the naming conventions illustrated in Table 1.

Type trait	Example	Typographical convention
extensible type choice	int / text / ...	\$NAME-type-choice
closed type choice	int / text	NAME-type-choice
group choice	( 1 => int // 2 => text )	\$\$NAME-group-choice
group	( 1 => int, 2 => text )	NAME-group
type	int	NAME-type
tagged type	#6.123(int)	tagged-NAME-type
map	{ 1 => int, 2 => text }	NAME-map
flags	&( a: 1, b: 2 )	NAME-flags

Table 1: Type Traits &amp; Typographical Conventions

### 1.3. Common Types

The following CDDL types are used in both CoRIM and CoMID.

#### 1.3.1. Non-Empty

The non-empty generic type is used to express that a map with only optional members MUST at least include one of the members.

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

#### 1.3.2. Entity

The entity-map is a generic type describing an organization responsible for the contents of a manifest. It is instantiated by supplying two parameters:

- \* A role-type-choice, i.e., a selection of roles that entities of the instantiated type can claim

- \* An extension-socket, i.e., a CDDL socket that can be used to extend the attributes associated with entities of the instantiated type

```
entity-map<role-type-choice, extension-socket> = {  
  &(entity-name: 0) => $entity-name-type-choice  
  ? &(reg-id: 1) => uri  
  &(role: 2) => [ + role-type-choice ]  
  * extension-socket  
}
```

\$entity-name-type-choice /= text

The following describes each member of the entity-map.

- \* entity-name (index 0): The name of entity which is responsible for the action(s) as defined by the role. \$entity-name-type-choice can only be Other specifications can extend the \$entity-name-type-choice (see Section 6.4).
- \* reg-id (index 1): A URI associated with the organization that owns the entity name
- \* role (index 2): A type choice defining the roles that the entity is claiming. The role is supplied as a parameter at the time the entity-map generic is instantiated.
- \* extension-socket: A CDDL socket used to add new information structures to the entity-map.

Examples of how the entity-map generic is instantiated can be found in Section 2.1.5 and Section 3.1.2.

### 1.3.3. Validity

A validity-map represents the time interval during which the signer warrants that it will maintain information about the status of the signed object (e.g., a manifest).

In a validity-map, both ends of the interval are encoded as epoch-based date/time as per Section 3.4.2 of [STD94].

```
validity-map = {  
  ? &(not-before: 0) => time  
  &(not-after: 1) => time  
}
```

- \* not-before (index 0): the date on which the signed manifest validity period begins
- \* not-after (index 1): the date on which the signed manifest validity period ends

#### 1.3.4. UUID

Used to tag a byte string as a binary UUID defined in Section 4.1.2. of [RFC4122].

```
uuid-type = bytes .size 16
tagged-uuid-type = #6.37(uuid-type)
```

#### 1.3.5. UEID

Used to tag a byte string as Universal Entity ID Claim (UUID) defined in Section 4.2.1 of [I-D.ietf-rats-eat].

```
ueid-type = bytes .size 33
tagged-ueid-type = #6.550(ueid-type)
```

#### 1.3.6. OID

Used to tag a byte string as the BER encoding [X.690] of an absolute object identifier [RFC9090].

```
oid-type = bytes
tagged-oid-type = #6.111(oid-type)
```

#### 1.3.7. Tagged Integer Type

// Content missing. Tracked at: <https://github.com/ietf-rats/draft-birkholz-rats-corim/issues/87>

```
tagged-int-type = #6.551(int)
```

#### 1.3.8. Hash Entry

A hash entry represents the value of a hashing operation together with the hash algorithm used. Defined in Section 2.9.1 of [I-D.ietf-sacm-coswid]. The CDDL is copied below for convenience.

```
hash-entry = [
  hash-alg-id: int
  hash-value: bytes
]
```

## 2. CoRIM

// Content missing. Tracked at: <https://github.com/ietf-rats/draft-birkholz-rats-corim/issues/98>

At the top-level, a CoRIM can either be a CBOR-tagged corim-map (Section 2.1) or a COSE signed corim-map (Section 2.2).

```
corim = #6.500(concise-rim-type-choice)
```

```
$concise-rim-type-choice /= #6.501(corim-map)
```

```
$concise-rim-type-choice /= #6.502(signed-corim)
```

### 2.1. CoRIM Map

The CDDL specification for the corim-map is as follows and this rule and its constraints must be followed when creating or validating a CoRIM map.

```
corim-map = {
  &(id: 0) => $corim-id-type-choice
  &(tags: 1) => [ + $concise-tag-type-choice ]
  ? &(dependent-rims: 2) => [ + corim-locator-map ]
  ? &(profile: 3) => [ + profile-type-choice ]
  ? &(rim-validity: 4) => validity-map
  ? &(entities: 5) => [ + corim-entity-map ]
  * $$corim-map-extension
}
```

The following describes each child item of this map.

- \* id (index 0): A globally unique identifier to identify a CoRIM. Described in Section 2.1.1
- \* tags (index 1): An array of one or more CoMID or CoSWID tags. Described in Section 2.1.2
- \* dependent-rims (index 2): One or more services supplying additional, possibly dependent, manifests or related files. Described in Section 2.1.3
- \* profile (index 3): One or more unique identifiers for the profiles of the tags contained in this CoRIM. All the listed profiles MUST be understood. Failure to recognize a profile identifier MUST result in the rejection of the entire processing. Described in Section 2.1.4



- \* rim-validity (index 4): Specifies the validity period of the CoRIM. Described in Section 1.3.3
- \* entities (index 5): A list of entities involved in a CoRIM life-cycle. Described in Section 2.1.5
- \* \$\$corim-map-extension: This CDDL socket is used to add new information structures to the corim-map. See Section 6.3.

#### 2.1.1. Identity

A CoRIM id can be either a text string or a UUID type that uniquely identifies a CoRIM.

```
$corim-id-type-choice /= tstr
$corim-id-type-choice /= uuid-type
```

#### 2.1.2. Tags

A \$concise-tag-type-choice is a tagged CBOR payload that carries either a CoMID (Section 3) or a CoSWID [I-D.ietf-sacm-coswid].

```
$concise-tag-type-choice /= #6.505(bytes .cbor concise-swid-tag)
$concise-tag-type-choice /= #6.506(bytes .cbor concise-mid-tag)
```

#### 2.1.3. Locator Map

The locator map contains pointers to repositories where dependent manifests, certificates, or other relevant information can be retrieved by the Verifier.

```
corim-locator-map = {
  &(href: 0) => uri
  ? &(thumbprint: 1) => hash-entry
}
```

The following describes each child element of this type.

- \* href (index 0): URI identifying the additional resource that can be fetched
- \* thumbprint (index 1): expected digest of the resource referenced by href. See Section 1.3.8.

#### 2.1.4. Profile Types

A profile specifies which of the optional parts of a CoRIM are required, which are prohibited and which extension points are exercised and how.

```
profile-type-choice = uri / tagged-oid-type
```

#### 2.1.5. Entities

The CoRIM Entity is an instantiation of the Entity generic (Section 1.3.2) using a \$corim-role-type-choice.

The only role defined in this specification for a CoRIM Entity is manifest-creator.

The \$\$corim-entity-map-extension extension socket is empty in this specification.

```
corim-entity-map =  
  entity-map<$corim-role-type-choice, $$corim-entity-map-extension>
```

```
$corim-role-type-choice /= &(manifest-creator: 1)
```

#### 2.2. Signed CoRIM

Signing a CoRIM follows the procedures defined in CBOR Object Signing and Encryption [RFC8152]. A CoRIM tag MUST be wrapped in a COSE\_Sign1 structure. The CoRIM MUST be signed by the CoRIM creator.

The following CDDL specification defines a restrictive subset of COSE header parameters that MUST be used in the protected header alongside additional information about the CoRIM encoded in a corim-meta-map (Section 2.2.2).

```
COSE-Sign1-corim = [  
  protected: bstr .cbor protected-corim-header-map  
  unprotected: unprotected-corim-header-map  
  payload: bstr .cbor tagged-corim-map  
  signature: bstr  
]
```

The following describes each child element of this type.

- \* protected: A CBOR Encoded protected header which is protected by the COSE signature. Contains information as given by Protected Header Map below.

- \* unprotected: A COSE header that is not protected by COSE signature.
- \* payload: A CBOR encoded tagged CoRIM.
- \* signature: A COSE signature block which is the signature over the protected and payload components of the signed CoRIM.

#### 2.2.1. Protected Header Map

```
protected-corim-header-map = {  
  &(alg-id: 1) => int  
  &(content-type: 3) => "application/corim-unsigned+cbor"  
  &(issuer-key-id: 4) => bstr  
  &(corim-meta: 8) => bstr .cbor corim-meta-map  
  * cose-label => cose-value  
}
```

The following describes each child item of this map.

- \* alg-id (index 1): An integer that identifies a signature algorithm.
- \* content-type (index 3): A string that represents the "MIME Content type" carried in the CoRIM payload.
- \* issuer-key-id (index 4): A bit string which is a key identity pertaining to the CoRIM Issuer.
- \* corim-meta (index 8): A map that contains metadata associated with a signed CoRIM. Described in Section 2.2.2.

Additional data can be included in the COSE header map as per Section 3 of [RFC8152].

#### 2.2.2. Meta Map

The CoRIM meta map identifies the entity or entities that create and sign the CoRIM. This ensures the consumer is able to identify credentials used to authenticate its signer.

```
corim-meta-map = {  
  &(signer: 0) => corim-signer-map  
  ? &(signature-validity: 1) => validity-map  
}
```

The following describes each child item of this group.

- \* signer (index 0): Information about the entity that signs the CoRIM. Described in Section 2.2.2.1
- \* signature-validity (index 1): Validity period for the CoRIM. Described in Section 1.3.3

#### 2.2.2.1. Signer Map

```
corim-signer-map = {  
  &(signer-name: 0) => $entity-name-type-choice  
  ? &(signer-uri: 1) => uri  
  * $$corim-signer-map-extension  
}  
  
* signer-name (index 0): Name of the organization that performs the  
  signer role  
  
* signer-uri (index 1): A URI identifying the same organization  
  
* $$corim-signer-map-extension: Extension point for future expansion  
  of the Signer map.
```

### 3. CoMID

// Content missing. Tracked at: <https://github.com/ietf-rats/draft-birkholz-rats-corim/issues/88>

#### 3.1. Structure

The CDDL specification for the concise-mid-tag map is as follows and this rule and its constraints MUST be followed when creating or validating a CoMID tag:

```
concise-mid-tag = {  
  ? &(language: 0) => text  
  &(tag-identity: 1) => tag-identity-map  
  ? &(entities: 2) => [ + entity-map ]  
  ? &(linked-tags: 3) => [ + linked-tag-map ]  
  &(triples: 4) => triples-map  
  * $$concise-mid-tag-extension  
}
```

The following describes each member of the concise-mid-tag map.

- \* lang (index 0): A textual language tag that conforms with IANA "Language Subtag Registry" [IANA.language-subtag-registry]. The context of the specified language applies to all sibling and

descendant textual values, unless a descendant object has defined a different language tag. Thus, a new context is established when a descendant object redefines a new language tag. All textual values within a given context MUST be considered expressed in the specified language.

- \* tag-identity (index 1): A tag-identity-map containing unique identification information for the CoMID. Described in Section 3.1.1.
- \* entities (index 2): Provides information about one or more organizations responsible for producing the CoMID tag. Described in Section 3.1.2.
- \* linked-tags (index 3): A list of one or more linked-tag-map (described in Section 3.1.3), providing typed relationships between this and other CoMIDs.
- \* triples (index 4): One or more triples providing information specific to the described module, e.g.: reference or endorsed values, cryptographic material, or structural relationship between the described module and other modules. Described in (Section 3.1.4).

#### 3.1.1. Tag Identity

```
tag-identity-map = {  
  &(tag-id: 0) => $tag-id-type-choice  
  ? &(tag-version: 1) => tag-version-type  
}
```

The following describes each member of the tag-identity-map.

- \* tag-id (index 0): A universally unique identifier for the CoMID. Described in Section 3.1.1.1.
- \* tag-version (index 1): Optional versioning information for the tag-id . Described in Section 3.1.1.2.

##### 3.1.1.1. Tag ID

```
$tag-id-type-choice /= tstr  
$tag-id-type-choice /= uuid-type
```

A Tag ID is either a 16-byte binary string, or a textual identifier, uniquely referencing the CoMID. The tag identifier MUST be globally unique. Failure to ensure global uniqueness can create ambiguity in tag use since the tag-id serves as the global key for matching,

lookups and linking. If represented as a 16-byte binary string, the identifier MUST be a valid universally unique identifier as defined by [RFC4122]. There are no strict guidelines on how the identifier is structured, but examples include a 16-byte GUID (e.g., class 4 UUID) [RFC4122], or a URI [STD66].

#### 3.1.1.2. Tag Version

```
tag-version-type = uint .default 0
```

Tag Version is an integer value that indicates the specific release revision of the tag. Typically, the initial value of this field is set to 0 and the value is increased for subsequent tags produced for the same module release. This value allows a CoMID tag producer to correct an incorrect tag previously released without indicating a change to the underlying module the tag represents. For example, the tag version could be changed to add new metadata, to correct a broken link, to add a missing reference value, etc. When producing a revised tag, the new tag-version value MUST be greater than the old tag-version value.

#### 3.1.2. Entities

```
comid-entity-map =  
  entity-map<$comid-role-type-choice, $$comid-entity-map-extension>
```

The CoMID Entity is an instantiation of the Entity generic (Section 1.3.2) using a \$comid-role-type-choice.

The \$\$comid-entity-map-extension extension socket is empty in this specification.

```
$comid-role-type-choice /= &(tag-creator: 0)  
$comid-role-type-choice /= &(creator: 1)  
$comid-role-type-choice /= &(maintainer: 2)
```

The roles defined for a CoMID entity are:

- \* tag-creator (value 0): creator of the CoMID tag.
- \* creator (value 1): original maker of the module described by the CoMID tag.
- \* maintainer (value 2): an entity making changes to the module described by the CoMID tag.

### 3.1.3. Linked Tag

The linked tag map represents a typed relationship between the embedding CoMID tag (the source) and another CoMID tag (the target).

```
linked-tag-map = {
  &(linked-tag-id: 0) => $tag-id-type-choice
  &(tag-rel: 1) => $tag-rel-type-choice
}
```

The following describes each member of the tag-identity-map.

- \* linked-tag-id (index 0): Unique identifier for the target tag. For the definition see Section 3.1.1.1.
- \* tag-rel (index 1): the kind of relation linking the source tag to the target identified by linked-tag-id.

```
$tag-rel-type-choice /= &(supplements: 0)
$tag-rel-type-choice /= &(replaces: 1)
```

The relations defined in this specification are:

- \* supplements (value 0): the source tag provides additional information about the module described in the target tag.
- \* replaces (value 1): the source tag corrects erroneous information contained in the target tag. The information in the target MUST be disregarded.

### 3.1.4. Triples

The triples-map contains all the CoMID triples broken down per category. Not all category need to be present but at least one category MUST be present and contain at least one entry.

```
triples-map = non-empty<{
  ? &(reference-triples: 0) => [ + reference-triple-record ]
  ? &(endorsed-triples: 1) => [ + endorsed-triple-record ]
  ? &(identity-triples: 2) => [ + identity-triple-record ]
  ? &(attest-key-triples: 3) => [ + attest-key-triple-record ]
  ? &(dependency-triples: 4) => [ + domain-dependency-triple-record ]
  ? &(membership-triples: 5) => [ + domain-membership-triple-record ]
  ? &(coswid-triples: 6) => [ + coswid-triple-record ]
  * $$triples-map-extension
}>
```

The following describes each member of the triples-map:

- \* `reference-triples (index 0)`: Triples containing reference values. Described in Section 3.1.4.2.
- \* `endorsed-triples (index 1)`: Triples containing endorsed values. Described in Section 3.1.4.3.
- \* `identity-triples (index 2)`: Triples containing identity credentials. Described in Section 3.1.4.4.
- \* `attest-key-triples (index 3)`: Triples containing verification keys associated with attesting environments. Described in Section 3.1.4.5.
- \* `dependency-triples (index 4)`: Triples describing trust relationships between domains. Described in Section 3.1.4.6.
- \* `membership-triples (index 5)`: Triples describing topological relationships between (sub-)modules. Described in Section 3.1.4.7.
- \* `coswid-triples (index 6)`: Triples associating modules with existing CoSWID tags. Described in Section 3.1.4.8.

#### 3.1.4.1. Common Types

##### 3.1.4.1.1. Environment

An `environment-map` may be used to represent a whole attester, an attesting environment, or a target environment. The exact semantic depends on the context (triple) in which the environment is used.

An environment is named after a class, instance or group identifier (or a combination thereof).

```
environment-map = non-empty<{  
  ? &(class: 0) => class-map  
  ? &(instance: 1) => $instance-id-type-choice  
  ? &(group: 2) => $group-id-type-choice  

```

The following describes each member of the `environment-map`:

- \* `class (index 0)`: Contains "class" attributes associated with the module. Described in Section 3.1.4.1.2.
- \* `instance (index 1)`: Contains a unique identifier of a module's instance. See Section 3.1.4.1.3.



- \* group (index 2): identifier for a group of instances, e.g., if an anonymization scheme is used.

#### 3.1.4.1.2. Class

The Class name consists of class attributes that distinguish the class of environment from other classes. The class attributes include class-id, vendor, model, layer, and index. The CoMID author determines which attributes are needed.

```
class-map = non-empty<{  
  ? &(class-id: 0) => $class-id-type-choice  
  ? &(vendor: 1) => tstr  
  ? &(model: 2) => tstr  
  ? &(layer: 3) => uint  
  ? &(index: 4) => uint  
>
```

```
$class-id-type-choice /= tagged-oid-type  
$class-id-type-choice /= tagged-uuid-type  
$class-id-type-choice /= tagged-int-type
```

The following describes each member of the class-map:

- \* class-id (index 0): Identifies the environment via a well-known identifier. Typically, class-id is an object identifier (OID) or universally unique identifier (UUID). Use of this attribute is preferred.
- \* vendor (index 1): Identifies the entity responsible for choosing values for the other class attributes that do not already have naming authority.
- \* model (index 2): Describes a product, generation, and family. If populated, vendor MUST also be populated.
- \* layer (index 3): Is used to capture where in a sequence the environment exists. For example, the order in which bootstrap code is executed may have security relevance.
- \* index (index 4): Is used when there are clones (i.e., multiple instances) of the same class of environment. Each clone is given a different index value to disambiguate it from the other clones. For example, given a chassis with several network interface controllers (NIC), each NIC can be given a different index value.

## 3.1.4.1.3. Instance

An instance carries a unique identifier that is reliably bound to an instance of the attester.

The types defined for an instance identifier are UEID or UUID.

```
$instance-id-type-choice /= tagged-ueid-type
$instance-id-type-choice /= tagged-uuid-type
```

## 3.1.4.1.4. Group

A group carries a unique identifier that is reliably bound to a group of attesters, for example when a number of attester are hidden in the same anonymity set.

The type defined for a group identified is UUID.

```
$group-id-type-choice /= tagged-uuid-type
```

## 3.1.4.1.5. Measurements

Measurements can be of a variety of things including software, firmware, configuration files, read-only memory, fuses, IO ring configuration, partial reconfiguration regions, etc. Measurements comprise raw values, digests, or status information.

An environment has one or more measurable elements. Each element can have a dedicated measurement or multiple elements could be combined into a single measurement. Measurements can have class, instance or group scope. This is typically determined by the triple's environment.

Class measurements apply generally to all the attesters in the given class. Instance measurements apply to a specific attester instances. Environments identified by a class identifier have measurements that are common to the class. Environments identified by an instance identifier have measurements that are specific to that instance.

```
measurement-map = {
  ? &(mkey: 0) => $measured-element-type-choice
  &(mval: 1) => measurement-values-map
}
```

The following describes each member of the measurement-map:

- \* mkey (index 0): An optional unique identifier of the measured (sub-)environment. See Section 3.1.4.1.5.1.

- \* mval (index 1): The measurements associated with the (sub-)environment. Described in Section 3.1.4.1.5.2.

#### 3.1.4.1.5.1. Measurement Keys

The types defined for a measurement identifier are OID, UUID or uint.

```
$measured-element-type-choice /= tagged-oid-type
$measured-element-type-choice /= tagged-uuid-type
$measured-element-type-choice /= uint
```

#### 3.1.4.1.5.2. Measurement Values

A measurement-values-map contains measurements associated with a certain environment. Depending on the context (triple) in which they are found, elements in a measurement-values-map can represent class or instance measurements. Note that some of the elements have instance scope only.

```
measurement-values-map = non-empty<{
  ? &(version: 0) => version-map
  ? &(svn: 1) => svn-type-choice
  ? &(digests: 2) => [ + hash-entry ]
  ? &(flags: 3) => flags-map
  ? (
    &(raw-value: 4) => $raw-value-type-choice,
    ? &(raw-value-mask: 5) => raw-value-mask-type
  )
  ? &(mac-addr: 6) => mac-addr-type-choice
  ? &(ip-addr: 7) => ip-addr-type-choice
  ? &(serial-number: 8) => text
  ? &(ueid: 9) => ueid-type
  ? &(uuid: 10) => uuid-type
  ? &(name: 11) => text
  * $$measurement-values-map-extension
}>
```

The following describes each member of the measurement-values-map.

- \* version (index 0): Typically changes whenever the measured environment is updated. Described in Section 3.1.4.1.5.3.
- \* svn (index 1): The security version number typically changes only when a security relevant change is made to the measured environment. Described in Section 3.1.4.1.5.4.

- \* `digests` (index 2): Contains the digest(s) of the measured environment together with the respective hash algorithm used in the process. See Section 1.3.8.
- \* `flags` (index 3): Describes security relevant operational modes. For example, whether the environment is in a debug mode, recovery mode, not fully configured, not secure, not replay protected or not integrity protected. The `flags` field indicates which operational modes are currently associated with measured environment. Described in Section 3.1.4.1.5.5.
- \* `raw-value` (index 4): Contains the actual (not hashed) value of the element. An optional `raw-value-mask` (index 5) indicates which bits in the `raw-value` field are relevant for verification. A mask of all ones ("1") means all bits in the `raw-value` field are relevant. Multiple values could be combined to create a single `raw-value` attribute. The vendor determines how to pack multiple values into a single `raw-value` structure. The same packing format is used when collecting Evidence so that Reference Values and collected values are bit-wise comparable. The vendor determines the encoding of `raw-value` and the corresponding `raw-value-mask`.
- \* `mac-addr` (index 6): A EUI-48 or EUI-64 MAC address associated with the measured environment. Described in Section 3.1.4.1.5.7.
- \* `ip-addr` (index 7): An IPv4 or IPv6 address associated with the measured environment. Described in Section 3.1.4.1.5.7.
- \* `serial-number` (index 8): A text string representing the product serial number.
- \* `ueid` (index 9): UEID associated with the measured environment. See Section 1.3.5.
- \* `uuid` (index 10): UUID associated with the measured environment. See Section 1.3.4.
- \* `name` (index 11): a name associated with the measured environment.

#### 3.1.4.1.5.3. Version

A `version-map` contains details about the versioning of a measured environment.

```
version-map = {  
  &(version: 0) => text  
  ? &(version-scheme: 1) => $version-scheme  
}
```

The following describes each member of the version-map:

- \* version (index 0): the version string
- \* version-scheme (index 1): an optional indicator of the versioning convention used in the version attribute. Defined in Section 4.1 of [I-D.ietf-sacm-coswid]. The CDDL is copied below for convenience.

```
$version-scheme /= &(multipartnumeric: 1)
$version-scheme /= &(multipartnumeric-suffix: 2)
$version-scheme /= &(alphanumeric: 3)
$version-scheme /= &(decimal: 4)
$version-scheme /= &(semver: 16384)
$version-scheme /= int / text
```

#### 3.1.4.1.5.4. Security Version Number

// Content missing. Tracked at: <https://github.com/ietf-rats/draft-birkholz-rats-corim/issues/89>

```
svn-type = uint
svn = svn-type
min-svn = svn-type
tagged-svn = #6.552(svn)
tagged-min-svn = #6.553(min-svn)
svn-type-choice = tagged-svn / tagged-min-svn
```

#### 3.1.4.1.5.5. Flags

The flags-map measurement describes a number of boolean operational modes. If a flags-map value is not specified, then the operational mode is unknown.

```
flags-map = {
  ? &(configured: 0) => bool
  ? &(secure: 1) => bool
  ? &(recovery: 2) => bool
  ? &(debug: 3) => bool
  ? &(replay-protected: 4) => bool
  ? &(integrity-protected: 5) => bool
  * $$flags-map-extension
}
```

The following describes each member of the flags-map:

- \* configured (index 0): The measured environment is fully configured for normal operation if the flag is true.
- \* secure (index 1): The measured environment's configurable security settings are fully enabled if the flag is true.
- \* recovery (index 2): The measured environment is NOT in a recovery state if the flag is true.
- \* debug (index 3): The measured environment is in a debug enabled state if the flag is true.
- \* replay-protected (index 4): The measured environment is protected from replay by a previous image that differs from the current image if the flag is true.
- \* integrity-protected (index 5): The measured environment is protected from unauthorized update if the flag is true.

#### 3.1.4.1.5.6. Raw Values Types

// Content missing. Tracked at: <https://github.com/ietf-rats/draft-birkholz-rats-corim/issues/90>

\$raw-value-type-choice /= #6.560(bytes)

raw-value-mask-type = bytes

#### 3.1.4.1.5.7. Address Types

The types or associating addressing information to a measured environment are:

ip-addr-type-choice = ip4-addr-type / ip6-addr-type

ip4-addr-type = bytes .size 4

ip6-addr-type = bytes .size 16

mac-addr-type-choice = eui48-addr-type / eui64-addr-type

eui48-addr-type = bytes .size 6

eui64-addr-type = bytes .size 8

#### 3.1.4.1.6. Crypto Keys

A cryptographic key can be one of the following formats:

- \* tagged-pkix-base64-key-type: PEM encoded SubjectPublicKeyInfo. Defined in Section 13 of [RFC7468].

- \* tagged-pkix-base64-cert-type: PEM encoded X.509 public key certificate. Defined in Section 5 of [RFC7468].
- \* tagged-pkix-base64-cert-path-type: X.509 certificate chain created by the concatenation of as many PEM encoded X.509 certificates as needed. The certificates MUST be concatenated in order so that each directly certifies the one preceding.

```
$crypto-key-type-choice /= tagged-pkix-base64-key-type  
$crypto-key-type-choice /= tagged-pkix-base64-cert-type  
$crypto-key-type-choice /= tagged-pkix-base64-cert-path-type
```

```
tagged-pkix-base64-key-type = #6.554(tstr)  
tagged-pkix-base64-cert-type = #6.555(tstr)  
tagged-pkix-base64-cert-path-type = #6.556(tstr)
```

#### 3.1.4.1.7. Domain Types

A domain is a context for bundling a collection of related environments and their measurements.

Three types are defined: uint and text for local scope, UUID for global scope.

```
$domain-type-choice /= uint  
$domain-type-choice /= text  
$domain-type-choice /= tagged-uuid-type
```

#### 3.1.4.2. Reference Values Triple

A Reference Values triple relates reference measurements to a Target Environment. For Reference Value Claims, the subject identifies a Target Environment, the object contains measurements, and the predicate asserts that these are the expected (i.e., reference) measurements for the Target Environment.

```
reference-triple-record = [  
  environment-map  
  [ + measurement-map ]  
]
```

#### 3.1.4.3. Endorsed Values Triple

An Endorsed Values triple declares additional measurements that are valid when a Target Environment has been verified against reference measurements. For Endorsed Value Claims, the subject is either a Target or Attesting Environment, the object contains measurements, and the predicate defines semantics for how the object relates to the subject.

```
endorsed-triple-record = [  
  environment-map  
  [ + measurement-map ]  
]
```

#### 3.1.4.4. Device Identity Triple

A Device Identity triple relates one or more cryptographic keys to a device. The subject of an Identity triple uses an instance or class identifier to refer to a device, and a cryptographic key is the object. The predicate asserts that the identity is authenticated by the key. A common application for this triple is device identity.

```
identity-triple-record = [  
  environment-map  
  [ + $crypto-key-type-choice ]  
]
```

#### 3.1.4.5. Attestation Keys Triple

An Attestation Keys triple relates one or more cryptographic keys to an Attesting Environment. The Attestation Key triple subject is an Attesting Environment whose object is a cryptographic key. The predicate asserts that the Attesting Environment signs Evidence that can be verified using the key.

```
attest-key-triple-record = [  
  environment-map  
  [ + $crypto-key-type-choice ]  
]
```

#### 3.1.4.6. Domain Dependency Triple

A Domain Dependency triple defines trust dependencies between measurement sources. The subject identifies a domain (Section 3.1.4.1.7) that has a predicate relationship to the object containing one or more dependent domains. Dependency means the subject domain's trustworthiness properties rely on the object domain(s) trustworthiness having been established before the



trustworthiness properties of the subject domain exists.

```
domain-dependency-triple-record = [  
  $domain-type-choice  
  [ + $domain-type-choice ]  
]
```

#### 3.1.4.7. Domain Membership Triple

A Domain Membership triple assigns domain membership to environments. The subject identifies a domain (Section 3.1.4.1.7) that has a predicate relationship to the object containing one or more environments. Endorsed environments (Section 3.1.4.3) membership is conditional upon successful matching of Reference Values (Section 3.1.4.2) to Evidence.

```
domain-membership-triple-record = [  
  $domain-type-choice  
  [ + environment-map ]  
]
```

#### 3.1.4.8. CoMID-CoSWID Linking Triple

A CoSWID triple relates reference measurements contained in one or more CoSWIDs to a Target Environment. The subject identifies a Target Environment, the object one or more unique tag identifiers of existing CoSWIDs, and the predicate asserts that these contain the expected (i.e., reference) measurements for the Target Environment.

```
coswid-triple-record = [  
  environment-map  
  [ + concise-swid-tag-id ]  
]
```

```
concise-swid-tag-id = text / bstr .size 16
```

### 3.2. Extensibility

```
// Content missing. Tracked at: https://github.com/ietf-rats/draft-birkholz-rats-corim/issues/91
```

#### 4. 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".

##### 4.1. Veraison

- \* Organization responsible for the implementation: Veraison Project, Linux Foundation
- \* Implementation's web page: <https://github.com/veraison/corim/README.md> (<https://github.com/veraison/corim/README.md>)
- \* Brief general description: The corim/corim and corim/comid packages provide a golang API for low-level manipulation of Concise Reference Integrity Manifest (CoRIM) and Concise Module Identifier (CoMID) tags respectively. The corim/cocli package uses the API above (as well as the API from the veraison/swid package) to provide a user command line interface for working with CoRIM, CoMID and CoSWID. Specifically, it allows creating, signing, verifying, displaying, uploading, and more. See <https://github.com/cocli/README.md> (<https://github.com/cocli/README.md>) for further details.
- \* Implementation's level of maturity: alpha.
- \* Coverage: the whole protocol is implemented, including PSA-specific extensions [I-D.fdb-rats-psa-endorsements].
- \* Version compatibility: Version -02 of the draft

- \* Licensing: Apache 2.0 <https://github.com/veraison/corim/blob/main/LICENSE> (<https://github.com/veraison/corim/blob/main/LICENSE>)
- \* Implementation experience: n/a
- \* Contact information: <https://veraison.zulipchat.com> (<https://veraison.zulipchat.com>)
- \* Last updated: <https://github.com/veraison/corim/commits/main> (<https://github.com/veraison/corim/commits/main>)

## 5. Security and Privacy Considerations

// Content missing. Tracked at: <https://github.com/ietf-rats/draft-birkholz-rats-corim/issues/92>

## 6. IANA Considerations

### 6.1. New COSE Header Parameters

// Content missing. Tracked at: <https://github.com/ietf-rats/draft-birkholz-rats-corim/issues/96>

### 6.2. New CBOR Tags

// Content missing. Tracked at: <https://github.com/ietf-rats/draft-birkholz-rats-corim/issues/93>

### 6.3. New CoRIM Registries

// Content missing. Tracked at: <https://github.com/ietf-rats/draft-birkholz-rats-corim/issues/94>

### 6.4. New CoMID Registries

// Content missing. Tracked at: <https://github.com/ietf-rats/draft-birkholz-rats-corim/issues/95>

### 6.5. New Media Types

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

Name	Template	Reference
corim-signed+cbor	application/corim-signed+cbor	RFCthis, Section 6.5.1
corim-unsigned+cbor	application/corim-unsigned+cbor	RFCthis, Section 6.5.2

Table 2: New Media Types

## 6.5.1. corim-signed+cbor

Type name: application  
 Subtype name: corim-signed+cbor  
 Required parameters: n/a  
 Optional parameters: "profile" (CoRIM profile in string format.  
     OIDs MUST use the dotted-decimal notation.)  
 Encoding considerations: binary  
 Security considerations: Section 5 of RFCthis  
 Interoperability considerations: n/a  
 Published specification: RFCthis  
 Applications that use this media type: Attestation Verifiers,  
     Endorsers and Reference-Value providers that need to transfer COSE  
     Sign1 wrapped CoRIM payloads over HTTP(S), CoAP(S), and other  
     transports.  
 Fragment identifier considerations: n/a  
 Magic number(s): D9 01 F6 D2, D9 01 F4 D9 01 F6 D2  
 File extension(s): n/a  
 Macintosh file type code(s): 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? Maybe

## 6.5.2. corim-unsigned+cbor

Type name: application  
 Subtype name: corim-unsigned+cbor  
 Required parameters: n/a  
 Optional parameters: "profile" (CoRIM profile in string format.  
     OIDs MUST use the dotted-decimal notation.)  
 Encoding considerations: binary  
 Security considerations: Section 5 of RFCthis  
 Interoperability considerations: n/a

Published specification: RFCthis  
 Applications that use this media type: Attestation Verifiers,  
 Endorsers and Reference-Value providers that need to transfer  
 unprotected CoRIM payloads over HTTP(S), CoAP(S), and other  
 transports.  
 Fragment identifier considerations: n/a  
 Magic number(s): D9 01 F5, D9 01 F4 D9 01 F5  
 File extension(s): n/a  
 Macintosh file type code(s): 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? Maybe

## 6.6. CoAP Content-Formats Registration

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

Content-Type	Content Coding	ID	Reference
application/corim-signed+cbor	-	TBD1	RFCthis
application/corim-unsigned+cbor	-	TBD2	RFCthis

Table 3: New Content-Formats

## 7. References

### 7.1. Normative References

[I-D.ietf-rats-architecture]

Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote Attestation Procedures Architecture", Work in Progress, Internet-Draft, draft-ietf-rats-architecture-18, 14 June 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-rats-architecture-18>>.

- [I-D.ietf-rats-eat]  
Lundblade, L., Mandyam, G., and J. O'Donoghue, "The Entity Attestation Token (EAT)", Work in Progress, Internet-Draft, draft-ietf-rats-eat-14, 10 July 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-rats-eat-14>>.
- [I-D.ietf-sacm-coswid]  
Birkholz, H., Fitzgerald-McKay, J., Schmidt, C., and D. Waltermire, "Concise Software Identification Tags", Work in Progress, Internet-Draft, draft-ietf-sacm-coswid-21, 7 March 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-sacm-coswid-21>>.
- [IANA.core-parameters]  
IANA, "Constrained RESTful Environments (CoRE) Parameters", <<https://www.iana.org/assignments/core-parameters>>.
- [IANA.language-subtag-registry]  
IANA, "Language Subtag Registry", <<https://www.iana.org/assignments/language-subtag-registry>>.
- [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>>.
- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <<https://www.rfc-editor.org/rfc/rfc4122>>.
- [RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/RFC7468, April 2015, <<https://www.rfc-editor.org/rfc/rfc7468>>.
- [RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <<https://www.rfc-editor.org/rfc/rfc8152>>.

- [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>>.
- [RFC9090] Bormann, C., "Concise Binary Object Representation (CBOR) Tags for Object Identifiers", RFC 9090, DOI 10.17487/RFC9090, July 2021, <<https://www.rfc-editor.org/rfc/rfc9090>>.
- [STD66] 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>>.
- [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.690] International Telecommunications Union, "Information technology ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, August 2015, <<https://www.itu.int/rec/T-REC-X.690>>.

## 7.2. Informative References

- [I-D.fdb-rats-psa-endorsements] Fossati, T., Deshpande, Y., and H. Birkholz, "Arm's Platform Security Architecture (PSA) Attestation Verifier Endorsements", Work in Progress, Internet-Draft, draft-fdb-rats-psa-endorsements-01, 11 May 2022, <<https://datatracker.ietf.org/doc/html/draft-fdb-rats-psa-endorsements-01>>.
- [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>>.

## Appendix A. Full CoRIM CDDL

```
// Content missing. Tracked at: https://github.com/ietf-rats/draft-birkholz-rats-corim/issues/80
```

```
corim = []
```

## Acknowledgments

Carl Wallace for review and comments on this document.

## Authors' Addresses

Henk Birkholz  
Fraunhofer SIT  
Email: [henk.birkholz@sit.fraunhofer.de](mailto:henk.birkholz@sit.fraunhofer.de)

Thomas Fossati  
arm  
Email: [Thomas.Fossati@arm.com](mailto:Thomas.Fossati@arm.com)

Yogesh Deshpande  
arm  
Email: [yogesh.deshpande@arm.com](mailto:yogesh.deshpande@arm.com)

Ned Smith  
Intel  
Email: [ned.smith@intel.com](mailto:ned.smith@intel.com)

Wei Pan  
Huawei Technologies  
Email: [william.panwei@huawei.com](mailto:william.panwei@huawei.com)