Workgroup: RATS Working Group
Internet-Draft: draft-birkholz-rats-corim-01
Published: 26 July 2021
Intended Status: Standards Track
Expires: 27 January 2022
Authors: H. Birkholz (Fraunhofer SIT)
T. Fossati (Arm Limited)
Y. Deshpande (Arm Limited)
N. Smith (Intel)
W. Pan (Huawei Technologies)

Concise Reference Integrity Manifest

Abstract

Remote Attestation Procedures (RATS) enable Relying Parties to put trust in the trustworthiness of a remote Attester and therefore to decide if to engage in secure interactions with it - or not. Evidence about trustworthiness can be rather complex, voluminous or Attester-specific. As it is deemed unrealistic that every Relying Party is capable of the appraisal of Evidence, that burden is taken on by a Verifier. In order to conduct Evidence appraisal procedures, a Verifier requires not only fresh Evidence from an Attester, but also trusted Endorsements and Reference Values from Endorsers, such as manufacturers, distributors, or 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), 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 27 January 2022.

Copyright Notice

Table of Contents

1. Introduction

The Remote Attestation Procedures (RATS) architecture [I-D.ietf-rats-architecture] describes appraisal procedures for attestation Evidence and Attestation Results. Appraisal procedures for Evidence are conducted by Verifiers and are intended to assess the trustworthiness of a remote peer. Appraisal procedures for Attestation Results are conducted by Relying Parties and are intended to operationalize the assessment about a remote peer and to act appropriately based on the assessment. In order to enable their intent, appraisal procedures consume Appraisal Policies, Reference Values, and Endorsements.

This documents specifies a binary encoding for Reference Values using the Concise Binary Object Representation (CBOR). The encoding is based on three parts that are defined using the Concise Data Definition Language (CDDL):

*Concise Reference Integrity Manifests (CoRIM),

*Concise Module Identifiers (CoMID), and

*Concise Software Identifier (CoSWID).

CoRIM and CoMID tags are defined in this document, CoSWID tags are defined in [I-D.ietf-sacm-coswid]. CoRIM provide a wrapper structure, in which CoMID tags, CoSWID tags, as well as corresponding metadata can be bundled and signed as a whole. CoMID tags represent hardware components and provide a counterpart to CoSWID tags, which represent software components.

In accordance to [RFC4949], software components that are stored in hardware modules are referred to as firmware. While firmware can be represented as a software component, it is also very hardware-specific and often resides directly on block devices instead of a file system. In this specification, firmware and their Reference Values are represented via CoMID tags. Reference Values for any other software components stored on a file system are represented via CoSWID tags.

In addition to CoRIM - and respective CoMID tags - this specification defines a Concise Manifest Revocation that represents a list of reference to CoRIM that are actively marked as invalid before their expiration time.

1.1. Requirements Notation

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.

2. Concise Reference Integrity Manifests

This section specifies the Concise RIM (CoRIM) format, the Concise MID format (CoMID), and the extension to the CoSWID specification that augments CoSWID tags to express specific relationships to CoMID tags.

While each specification defines its own start rule, only CoMID and CoSWID are stand-alone specifications. The CoRIM specification - as the bundling format - has a dependency on CoMID and CoSWID and is not a stand-alone specification.

While stand-alone CoSWID tags may be signed [I-D.ietf-sacm-coswid], CoMID tags are not intended to be stand-alone and are always part of a CoRIM that must be signed. [I-D.ietf-sacm-coswid] specifies the use of COSE [RFC7231] for signing. This specification defines how to

generate singed CoRIM tags with COSE to enable proof of authenticity and temper-evidence.

This document uses the Concise Data Definition Language (CDDL [RFC8610]) to define the data structure of CoRIM and CoMID tags, as well as the extensions to CoSWID. The CDDL definitions provided define nested containers. Typically, the CDDL types used for nested containers are maps. Every key used in the maps is a named type that is associated with an corresponding uint via a block of rules appended at the end of the CDDL definition.

Every set of uint keys that is used in the context of the "collision domain" of map is intended to be collision-free (each key is intended to be unique in the scope of a map, not a multimap). To accomplish that, for each map there is an IANA registry for the map members of maps.

2.1. Typographical Conventions

Type names in the following CDDL definitions follow the naming convention illustrated in table Table 1.

type trait	example	typo 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 & Typographical Convention

2.2. Prefixes and Namespaces

The semantics of the information elements (attributes) defined for CoRIM, CoMID tags, and CoSWID tags are sometimes very similar, but often do not share the same scope or are actually quite different. In order to not overload the already existing semantics of the software-centric IANA registries of CoSWID tags with, for example, hardware-centric semantics of CoMID tags, new type names are introduced. For example: both CoSWID tags and CoMID tags define a tag-id. As CoSWID already specifies tag-id, the tag-id in CoMID tags is prefixed with comid. to disambiguate the context, resulting in comid.tag-id. This prefixing provides a well-defined scope for the use of the types defined in this document and guarantees

interoperability (no type name collisions) with the CoSWID CDDL definition. Effectively, the prefixes used in this specification enable simple hierarchical namespaces. The prefixing introduced is also based on the anticipated namespace features for CDDL.

2.3. Extensibility

Both the CoRIM and the CoMID tag specification include extension points using CDDL sockets (see [RFC8610] Section 3.9). The use of CDDL sockets allows for well-formed extensions to be defined in supplementary CDDL definitions that support additional uses of CoRIM and CoMID tags.

There are two types of extensibility supported via the extension points defined in this document. Both types allow for the addition of keys in the scope of a map.

Custom Keys: The CDDL definition allows for the use of negative integers as keys. These keys cannot take on a well-defined global semantic. They can take on custom-defined semantics in a limited or local scope, e.g. vendor-defined scope.

Registered Keys: Additional keys can be registered at IANA via separate specifications.

Both types of extensibility also allow for the definition of new nested maps that again can include additional defined keys.

2.4. Concise RIM Extension Points

The following CDDL sockets (extension points) are defined in the CoRIM specification, which allow the addition of new information structures to their respective CDDL groups.

Map Name	CDDL Socket	Defined in
corim-entity-map	$$corim-entity-map-extension	Section 3.1.2
unsigned-corim-map	$$unsigned-corim-map-extension	Section 3.2
concise-mid-tag	$$comid-extension	Section 3.3
tag-identity-map	$$tag-identity-map-extension	Section 3.4
entity-map	$$entity-map-extension	Section 3.5
triples-map	$$triples-map-extension	Section 3.7
measurement-values-map	$$measurement-values-map-extension	Section 3.10

Table 2: CoMID CDDL Group Extension Points

2.5. CDDL Generic Types

The CDDL definitions for CoRIM and CoMID tags use the two following generic types.

2.5.1. 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 optional members.

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

2.5.2. One-Or-More

The one-or-more generic type allows to omit an encapsulating array, if only one member would be present.

```
one-or-more<T> = T / [ 2* T ] ; 2*
```

3. Concise RIM Data Definition

A CoRIM is a bundle of CoMID tags and/or CoSWID tags that can reference each other and that includes additional metadata about that bundle.

The root of the CDDL specification provided for CoRIM is the rule corim :

```
start = corim
```

3.1. The signed-corim Container

A CoRIM is signed using [RFC7231]. The additional CoRIM-specific COSE header member label corim-meta is defined as well as the corresponding type corim-meta-map as its value. This rule and its constraints MUST be followed when generating or validating a signed CoMID tag.

```
signed-corim = #6.18(COSE-Sign1-corim)

protected-signed-corim-header-map = {
  corim.alg-id => int
  corim.content-type => "application/rim+cbor"
  corim.issuer-key-id => bstr
  corim.meta => corim-meta-map
  * cose-label => cose-values
}

unprotected-signed-corim-header-map = {
  * cose-label => cose-values
}

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

3.1.1. The corim-meta-map Container

This map contains the two additionally defined attributes corim-entity-map and validity-map that are used to annotate a CoRIM with metadata.

```
corim-meta-map = {
  corim.signer => one-or-more<corim-entity-map>
  ? corim.validity => validity-map
}
```

corim.signer: One or more entities that created and/or signed the issued CoRIM.

corim.validity: A time period defining the validity span of a CoRIM.

3.1.2. The corim-entity-map Container

This map is used to identify the signer of a CoRIM via a dedicated entity name, a corresponding role and an optional identifying URI.

```
corim-entity-map = {
  corim.entity-name => $entity-name-type-choice
  ? corim.reg-id => uri
  corim.role => $corim-role-type-choice
  * $$corim-entity-map-extension
}

$corim-role-type-choice /= corim.manifest-creator
$corim-role-type-choice /= corim.manifest-signer
```

- **corim.entity-name:** The name of the organization that takes on the role expressed in corim.role

- **corim.reg-id:** The registration identifier of the organization that has authority over the namespace for corim.entity-name.

- **corim.role:** The role type that is associated with the entity, e.g. the creator of the CoRIM or the signer of the CoRIM.

- **$$corim-entity-map-extension:** This CDDL socket is used to add new information elements to the corim-entity-map container. See FIXME.

3.1.3. The validity-map Container

The members of this map indicate the life-span or period of validity of a CoRIM that is baked into the protected header at the time of signing.

```
validity-map = {
  ? corim.not-before => time
  corim.not-after => time
}
```

- **corim.not-before:** The timestamp indicating the CoRIM's begin of its validity period.

- **corim.not-after:** The timestamp indicating the CoRIM's end of its validity period.

3.2. The unsigned-corim-map Container

This map contains the payload of the COSE envelope that is used to sign the CoRIM. This rule and its constraints MUST be followed when generating or validating an unsigned Concise RIM.

```
unsigned-corim-map = {
  corim.id => $corim-id-type-choice
  corim.tags => one-or-more<$concise-tag-type-choice>
  ? corim.dependent-rims => one-or-more<corim-locator-map>
  * $$unsigned-corim-map-extension
}

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

$concise-tag-type-choice /= #6.TBD-SWID(bytes .cbor concise-swid-tag)
$concise-tag-type-choice /= #6.TBD-CoMID(bytes .cbor concise-mid-tag)
```

corim.id: Typically a UUID or a text string that MUST uniquely identify a CoRIM in a given scope.

corim.tags: A collection of one or more CoMID tags and/or CoSWID tags.

corim.dependent-rims: One or more services available via the Internet that can supply additional, possibly dependent manifests (or other associated resources).

$$unsigned-corim-map-extension: This CDDL socket is used to add new information elements to the unsigned-corim-map container. See FIXME.

3.2.1. The corim-locator-map Container

This map is used to locate and verify the integrity of resources provided by external services, e.g. the CoRIM provider.

```
corim-locator-map = {
  corim.href => uri
  ? corim.thumbprint => hash-entry
}
```

corim.href: A pointer to a services that supplies dependent files or records.

corim.thumbprint: A digest of the reference resource.

3.3. The concise-mid-tag Container

The CDDL specification for the root concise-mid-tag map is as follows. This rule and its constraints MUST be followed when generating or validating a CoMID tag.

```
concise-mid-tag = {
  ? comid.language => language-type
  comid.tag-identity => tag-identity-map
  ? comid.entity => one-or-more<entity-map>
  ? comid.linked-tags => one-or-more<linked-tag-map>
  comid.triples => triples-map
  * $$concise-mid-tag-extension
}
```

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

comid.language: A textual language tag that conforms with the IANA Language Subtag Registry [IANA.language-subtag-registry].

comid.tag-identity: A composite identifier containing identifying attributes that enable global unique identification of a CoMID tag across versions.

comid.entity: A list of entities that contributed to the CoMID tag.

comid.linked-tags: A lost of tags that are linked to this CoMID tag.

comid.triples: A set of relationships in the form of triples, representing a graph-like and semantic reference structure between tags.

$$comid-mid-tag-extension: This CDDL socket is used to add new information elements to the concise-mid-tag root container. See FIXME.

3.4. The tag-identity-map Container

The CDDL specification for the tag-identity-map includes all identifying attributes that enable a consumer of information to anticipate required capabilities to process the corresponding tag that map is included in. This rule and its constraints MUST be followed when generating or validating a CoMID tag.

```
tag-identity-map = {
  comid.tag-id => $tag-id-type-choice
  comid.tag-version => tag-version-type
  * $$tag-identity-map-extension
}

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

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

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

comid.tag-id: An identifier for a CoMID that MUST be globally unique.

comid.tag-version: An unsigned integer used as a version identifier.

$$tag-identity-map-extension: This CDDL socket is used to add new information elements to the tag-identity-map container. See FIXME.

3.5. The entity-map Container

This Container provides qualifying attributes that provide more context information describing the module as well its origin and purpose. This rule and its constraints MUST be followed when generating or validating a CoMID tag.

```
entity-map = {
  comid.entity-name => $entity-name-type-choice
  ? comid.reg-id => uri
  comid.role => one-or-more<$comid-role-type-choice>
  * $$entity-map-extension
}

$comid-role-type-choice /= comid.tag-creator
$comid-role-type-choice /= comid.creator
$comid-role-type-choice /= comid.maintainer
```

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

comid.entity-name:
The name of an organization that performs the roles as indicated by comid.role.

comid.reg-id: The registration identifier of the organization that has authority over the namespace for comid.entity-name.

comid.role: The list of roles a CoMID entity is associated with. The entity that generates the concise-mid-tag SHOULD include a $comid-role-type-choice value of comid.tag-creator.

$$entity-map-extension: This CDDL socket is used to add new information elements to the entity-map container. See FIXME.

3.6. The linked-tag-map Container

A list of tags that are linked to this CoMID tag.

```
linked-tag-map = {
  comid.linked-tag-id => $tag-id-type-choice
  comid.tag-rel => $tag-rel-type-choice
}

$tag-rel-type-choice /= comid.supplements
$tag-rel-type-choice /= comid.replaces
```

The following describes each member of the linked-tag-map container.

comid.linked-tag-id: The tag-id of the linked tag. A linked tag MAY be a CoMID tag or a CoSWID tag.

comid.tag-rel: The relationship type with the linked tag. The relationship type MAY be supplements or replaces, as well as other types well-defined by additional specifications.

3.7. The triples-map Container

A set of directed properties that associate sets of data to provide reference values, endorsed values, verification key material or identifying key material for a specific hardware module that is a component of a composite device. The map provides the core element of CoMID tags that associate remote attestation relevant data with a distinct hardware component that runs an execution environment (a module that is either a Target Environment and/or an Attesting Environment). This rule and its constraints MUST be followed when generating or validating a CoMID tag.

```
triples-map = non-empty<{
  ? comid.reference-triples => one-or-more<reference-triple-record>
  ? comid.endorsed-triples => one-or-more<endorsed-triple-record>
  ? comid.attest-key-triples => one-or-more<attest-key-triple-record>
  ? comid.identity-triples => one-or-more<identity-triple-record>
  * $$triples-map-extension
}>
```

The following describes each member of the triple-map container.

comid.reference-triples: A directed property that associates reference measurements with a module that is a Target Environment.

comid.endorsed-triples: A directed property that associates endorsed measurements with a module that is a Target Environment or Attesting Environment.

comid.attest-key-triples: A directed property that associates key material used to verify evidence generated from a module that is an attesting environment.

comid.identity-triples: A directed property that associates key material used to identify a module instance or a module class that is an identifying part of a device(-set).

$$triples-map-extension: This CDDL socket is used to add new information elements to the triples-map container. See FIXME.

3.8. The environment-map Container

This map represents the module(s) that a triple-map can point directed properties (relationships) from in order to associate them with external information for remote attestation, such as reference values, endorsement and endorsed values, verification key material for evidence, or identifying key material for module (re-)identification. This map can identify a single module instance via comid.instance or groups of modules via comid.group. Referencing classes of modules requires the use of the more complex class-map container. This rule and its constraints MUST be followed when generating or validating a CoMID tag.

```
environment-map = non-empty<{
  ? comid.class => class-map
  ? comid.instance => $instance-id-type-choice
  ? comid.group => $group-id-type-choice
}>

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

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

The following describes each member of the environment-map container.

comid-class: A composite identifier for classes of environments/modules.

comid.instance: An identifier for distinct instances of environments/modules that is either a UEID or a UUID.

comid.group: An identifier for a group of environments/modules that is a UUID.

3.9. The class-map Container

This map enables a composite identifier intended to uniquely identify modules that are of a distinct class of devices. Effectively, all provided members in combination are a composite module class identifier. This rule and its constraints MUST be followed when generating or validating a CoMID tag. This rule and its constraints MUST be followed when generating or validating a CoMID tag.

```
class-map = non-empty<{
  ? comid.class-id => $class-id-type-choice
  ? comid.vendor => tstr
  ? comid.model => tstr
  ? comid.layer => uint
  ? comid.index => uint
}>

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

The following describes each member of the class-map container.

3.10. The measurement-map and measurement-values-map Containers

One of the targets (range) that a triple-map can point to in order to associate it with a module (domain) is the measurement-map. This map is used to provide reference measurements values that can be compared with Evidence Claim values or Endorsements and endorsed values from other sources than the corresponding CoRIM. measurement-map comes with a measurement key that identifies the measured element with via a OID reference or a UUID. measurement-values-map contains the actual measurements associated with the module(s). Byte strings with corresponding bit masks that highlights which bits in the byte string are used as reference measurements or endorsement are located in raw-value-group. The members of measurement-values-map provide well-defined and well-scoped semantics for reference measurement or endorsements with respect to a given module instance, class, or group. This rule and its constraints MUST be followed when generating or validating a CoMID tag.

```
measurement-map = {
  ? comid.mkey => $measured-element-type-choice
  comid.mval => measurement-values-map
}

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

measurement-values-map = non-empty<{
  ? comid.ver => version-map
  ? comid.svn => svn-type-choice
  ? comid.digests => digests-type
  ? comid.flags => flags-type
  ? raw-value-group
  ? comid.mac-addr => mac-addr-type-choice
  ? comid.ip-addr =>  ip-addr-type-choice
  ? comid.serial-number => serial-number-type
  ? comid.ueid => ueid-type
  ? comid.uuid => uuid-type
  * $$measurement-values-map-extension
}>

flags-type = bytes .bits operational-flags

operational-flags = &(
  not-configured: 0
  not-secure: 1
  recovery: 2
  debug: 3
)

serial-number-type = text

digests-type = one-or-more<hash-entry>
```

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

comid.mkey:
An identifier for the set of measurements expressed in measurement-values-map that is either an OID or a UUID.

comid.ver: A version number measurement.

comid.svn: A security related version number measurement.

comid.digests: A digest (typically a hash value) measurement.

comid.flags: Measurements that reflect operational modes that are made permanent at manufacturing time such that they are not expected to change during normal operation of the Attester.

raw-value-group: A measurement in the form of a byte string that can come with a corresponding bit mask defining the relevance of each bit in the byte string as a measurement.

comid.mac-addr: An EUI-48 or EUI-64 MAC address measurement.

comid.ip-addr: An Ipv4 or Ipv6 address measurement.

comid.serial-number: A measurement of a serial number in text.

comid.ueid: A measurement of a Unique Enough Identifier (UEID).

comid.uuid: A measurement of a Universally Unique Identifier (UUID).

$$measurement-values-map-extension: This CDDL socket is used to add new information elements to the measurement-values-map container. See FIXME.

3.10.1. The version-map Container

This map expresses reference values about version information.

```
version-map = {
  comid.version => version-type
  ? comid.version-scheme => $version-scheme
}

version-type = text .default '0.0.0'
```

The following describes each member of the version-map container.

comid.version:
The version in the form of a text string.

comid-version-scheme: The version-scheme of the text string value as defined in [I-D.ietf-sacm-coswid]

3.10.2. The svn-type-choice Enumeration

This choice defines the CBOR tagged Security Version Numbers (SVN) that can be used as reference values for Evidence and Endorsements.

```
svn = int
min-svn = int
tagged-svn = #6.TBD-SVN(svn)
tagged-min-svn = #6.TBD-minSVN(min-svn)
svn-type-choice = tagged-svn / tagged-min-svn
```

The following describes the types in the svn-type-choice enumeration.

tagged-svn: A specific SVN.

tagged-min-svn: A lower bound for allowed SVN.

3.10.3. The raw-value-group Container

FIXME This group can express a single raw byte value and can come with an optional bit mask that defines which bits in the byte string is used as a reference value, by setting corresponding position in the bit mask to 1.

```
raw-value-group = (
  comid.raw-value => raw-value-type
  ? comid.raw-value-mask => raw-value-mask-type
)

raw-value-type = bytes
raw-value-mask-type = bytes
```

The following describes the types in the raw-value-group Container.

comid.raw-value:
FIXME Bit positions in raw-value-type that correspond to bit positions in raw-value-mask-type.

comid.raw-value-mask: A raw-value-mask-type bit corresponding to a bit in raw-value-type MUST be 1 to evaluate the corresponding raw-value-type bit.

3.10.4. The ip-addr-type-choice Enumeration

This type choice expresses IP addresses as reference values.

```
ip-addr-type-choice = ip4-addr-type / ip6-addr-type
ip4-addr-type = bytes .size 4
ip6-addr-type = bytes .size 16
```

3.10.5. The mac-addr-type-choice Enumeration

This type choice expresses MAC addresses as reference values.

```
mac-addr-type-choice = eui48-addr-type / eui64-addr-type
eui48-addr-type = bytes .size 6
eui64-addr-type = bytes .size 8
```

3.11. The verification-key-map Container

One of the targets (range) that a triple-map can point to in order to associate it with a module (domain). This map is used to provide the key material for evidence verification (effectively signature checking or a lightweight proof-of-possession of private signing key material) or for identity assertion/check (where a proof-of-possession implies a certain device identity). In support of informed trust decisions, an optional trust anchor in the form a PKIX certification path that is associated with the provided key material can be included. This rule and its constraints MUST be followed when generating or validating a CoMID tag.

```
verification-key-map = {
  comid.key => pkix-base64-key-type
  ? comid.keychain => [ + pkix-base64-cert-type ]
}

pkix-base64-key-type = tstr
pkix-base64-cert-type = tstr
```

The following describes each member of the verification-key-map container.

comid.key: Verification key material in DER format base64 encoded. Typically, but not necessarily, a public key.

comid.keychain: One or more base64 encoded PKIX certificates. The certificate containing the public key in comid.key MUST be the first certificate. Additional certificates MAY follow. Each subsequent certificate SHOULD certify the previous certificate.

4. Full CDDL Definition

This section aggregates the CDDL definitions specified in this document in a full CDDL definitions including:

*the COSE envelope for CoRIM: signed-corim

*the CoRIM document: unsigned-corim

*the CoMID document: concise-mid-tag

Not included in the full CDDL definition are CDDL dependencies to CoSWID. The following CDDL definitions can be found in [I-D.ietf-sacm-coswid]:

*the COSE envelope for CoRIM: signed-coswid

*the CoSWID document: concise-swid-tag

```
<CODE BEGINS>
corim = #6.500($concise-reference-integrity-manifest-type-choice)

$concise-reference-integrity-manifest-type-choice /= #6.501(unsigned-corim-map)
$concise-reference-integrity-manifest-type-choice /= #6.502(signed-corim)


signed-corim = #6.18(COSE-Sign1-corim)

protected-signed-corim-header-map = {
  corim.alg-id => int
  corim.content-type => "application/rim+cbor"
  corim.issuer-key-id => bstr
  corim.meta => corim-meta-map
  * cose-label => cose-values
}

corim-meta-map = {
  corim.signer => [ + corim-entity-map ]
  ? corim.validity => validity-map
}

corim-entity-map = {
  corim.entity-name => $entity-name-type-choice
  ? corim.reg-id => uri
  corim.role => $corim-role-type-choice
  * $$corim-entity-map-extension
}

$corim-role-type-choice /= corim.manifest-creator
$corim-role-type-choice /= corim.manifest-signer

validity-map = {
  ? corim.not-before => time
  corim.not-after => time
}

unprotected-signed-corim-header-map = {
  * cose-label => cose-values
}

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


unsigned-corim-map = {
```

```
  corim.id => $corim-id-type-choice
  corim.tags => [ + $concise-tag-type-choice ]
  ? corim.dependent-rims => [ + corim-locator-map ]
  ? corim.profile => [ + profile-type-choice ]
  * $$unsigned-corim-map-extension
}

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

corim-locator-map = {
  corim.href => uri
  ? corim.thumbprint => hash-entry
}

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


concise-mid-tag = {
  ? comid.language => language-type
  comid.tag-identity => tag-identity-map
  ? comid.entity => [ + entity-map ]
  ? comid.linked-tags => [ + linked-tag-map ]
  comid.triples => triples-map
  * $$concise-mid-tag-extension
}

language-type = text

tag-identity-map = {
  comid.tag-id => $tag-id-type-choice
  ? comid.tag-version => tag-version-type
}

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

tag-version-type = uint .default 0

entity-map = {
  comid.entity-name => $entity-name-type-choice
  ? comid.reg-id => uri
  comid.role => [ + $comid-role-type-choice ]
  * $$entity-map-extension
}

$comid-role-type-choice /= comid.tag-creator
$comid-role-type-choice /= comid.creator
$comid-role-type-choice /= comid.maintainer
```

```
linked-tag-map = {
  comid.linked-tag-id => $tag-id-type-choice
  comid.tag-rel => $tag-rel-type-choice
}

$tag-rel-type-choice /= comid.supplements
$tag-rel-type-choice /= comid.replaces

triples-map = non-empty<{
  ? comid.reference-triples => [ + reference-triple-record ]
  ? comid.endorsed-triples => [ + endorsed-triple-record ]
  ? comid.attest-key-triples => [ + attest-key-triple-record ]
  ? comid.identity-triples => [ + identity-triple-record ]
  * $$triples-map-extension
}>

reference-triple-record = [
  environment-map ; target environment
  [ + measurement-map ] ; reference measurements
]

endorsed-triple-record = [
  environment-map ; (target or attesting) environment
  [ + measurement-map ] ; endorsed measurements
]

attest-key-triple-record = [
  environment-map ; attesting environment
  [ + verification-key-map ] ; attestation verification key(s)
]

identity-triple-record = [
  environment-map ; device identifier (instance or class)
  [ + verification-key-map ] ; DevID, or semantically equivalent
]

pkix-base64-key-type = tstr
pkix-base64-cert-type = tstr

verification-key-map = {
  ; Verification key in DER format base64-encoded.
  ; Typically, but not necessarily a public key.
  comid.key => pkix-base64-key-type
  ; Optional X.509 certificate chain corresponding to the public key
  ; in comid.key, encoded as an array of one or more base64-encoded
  ; DER PKIX certificates.  The certificate containing the public key
  ; in comid.key MUST be the first certificate.  This MAY be followed
  ; by additional certificates, with each subsequent certificate
  ; being the one used to certify the previous one.
  ? comid.keychain => [ + pkix-base64-cert-type ]
```

```
}

environment-map = non-empty<{
  ? comid.class => class-map
  ? comid.instance => $instance-id-type-choice
  ? comid.group => $group-id-type-choice
}>

class-map = non-empty<{
  ? comid.class-id => $class-id-type-choice
  ? comid.vendor => tstr
  ? comid.model => tstr
  ? comid.layer => uint
  ? comid.index => uint
}>

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

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

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

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

tagged-uuid-type = #6.37(uuid-type)

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

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

measurement-map = {
  ? comid.mkey => $measured-element-type-choice
  comid.mval => measurement-values-map
}

measurement-values-map = non-empty<{
  ? comid.ver => version-map
  ? comid.svn => svn-type-choice
  ? comid.digests => digests-type
  ? comid.flags => flags-type
  ? raw-value-group
  ? comid.mac-addr => mac-addr-type-choice
  ? comid.ip-addr =>  ip-addr-type-choice
  ? comid.serial-number => serial-number-type
  ? comid.ueid => ueid-type
```

```
  ? comid.uuid => uuid-type
  * $$measurement-values-map-extension
}>

version-map = {
  comid.version => version-type
  ? comid.version-scheme => $version-scheme
}
version-type = text .default '0.0.0'

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

flags-type = bytes .bits operational-flags

operational-flags = &(
  not-configured: 0
  not-secure: 1
  recovery: 2
  debug: 3
)

raw-value-group = (
  comid.raw-value => raw-value-type
  ? comid.raw-value-mask => raw-value-mask-type
)

raw-value-type = bytes
raw-value-mask-type = bytes

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

serial-number-type = text

digests-type = [ + hash-entry ]


concise-swid-tag = {
  tag-id => text / bstr .size 16,
  tag-version => integer,
  ? corpus => bool,
```

```
  ? patch => bool,
  ? supplemental => bool,
  software-name => text,
  ? software-version => text,
  ? version-scheme => $version-scheme,
  ? media => text,
  ? software-meta => one-or-more<software-meta-entry>,
  entity => one-or-more<entity-entry>,
  ? link => one-or-more<link-entry>,
  ? payload-or-evidence,
  * $$coswid-extension,
  global-attributes,
}

payload-or-evidence //= ( payload => payload-entry )
payload-or-evidence //= ( evidence => evidence-entry )

any-uri = uri
label = text / int

$version-scheme /= multipartnumeric
$version-scheme /= multipartnumeric-suffix
$version-scheme /= alphanumeric
$version-scheme /= decimal
$version-scheme /= semver
$version-scheme /= int / text

any-attribute = (
  label => one-or-more<text> / one-or-more<int>
)

one-or-more<T> = T / [ 2* T ]

global-attributes = (
  ? lang => text,
  * any-attribute,
)

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

entity-entry = {
  entity-name => text,
  ? reg-id => any-uri,
  role => one-or-more<$role>,
  ? thumbprint => hash-entry,
  * $$entity-extension,
  global-attributes,
```

```
}

$role /= tag-creator
$role /= software-creator
$role /= aggregator
$role /= distributor
$role /= licensor
$role /= maintainer
$role /= int / text

link-entry = {
  ? artifact => text,
  href => any-uri,
  ? media => text,
  ? ownership => $ownership,
  rel => $rel,
  ? media-type => text,
  ? use => $use,
  * $$link-extension,
  global-attributes,
}

$ownership /= shared
$ownership /= private
$ownership /= abandon
$ownership /= int / text

$rel /= ancestor
$rel /= component
$rel /= feature
$rel /= installationmedia
$rel /= packageinstaller
$rel /= parent
$rel /= patches
$rel /= requires
$rel /= see-also
$rel /= supersedes
$rel /= supplemental
$rel /= -256..64436 / text

$use /= optional
$use /= required
$use /= recommended
$use /= int / text

software-meta-entry = {
  ? activation-status => text,
  ? channel-type => text,
  ? colloquial-version => text,
```

```
  ? description => text,
  ? edition => text,
  ? entitlement-data-required => bool,
  ? entitlement-key => text,
  ? generator => text,
  ? persistent-id => text,
  ? product => text,
  ? product-family => text,
  ? revision => text,
  ? summary => text,
  ? unspsc-code => text,
  ? unspsc-version => text,
  * $$software-meta-extension,
  global-attributes,
}

path-elements-group = ( ? directory => one-or-more<directory-entry>,
                        ? file => one-or-more<file-entry>,
                      )

resource-collection = (
  path-elements-group,
  ? process => one-or-more<process-entry>,
  ? resource => one-or-more<resource-entry>,
  * $$resource-collection-extension,
)

file-entry = {
  filesystem-item,
  ? size => uint,
  ? file-version => text,
  ? hash => hash-entry,
  * $$file-extension,
  global-attributes,
}

directory-entry = {
  filesystem-item,
  ? path-elements => { path-elements-group },
  * $$directory-extension,
  global-attributes,
}

process-entry = {
  process-name => text,
  ? pid => integer,
  * $$process-extension,
  global-attributes,
}
```

```
resource-entry = {
  type => text,
  * $$resource-extension,
  global-attributes,
}

filesystem-item = (
  ? key => bool,
  ? location => text,
  fs-name => text,
  ? root => text,
)

payload-entry = {
  resource-collection,
  * $$payload-extension,
  global-attributes,
}

evidence-entry = {
  resource-collection,
  ? date => integer-time,
  ? device-id => text,
  * $$evidence-extension,
  global-attributes,
}

integer-time = #6.1(int)

tag-id = 0
software-name = 1
entity = 2
evidence = 3
link = 4
software-meta = 5
payload = 6
hash = 7
corpus = 8
patch = 9
media = 10
supplemental = 11
tag-version = 12
software-version = 13
version-scheme = 14
lang = 15
directory = 16
file = 17
process = 18
```

```
resource = 19
size = 20
file-version = 21
key = 22
location = 23
fs-name = 24
root = 25
path-elements = 26
process-name = 27
pid = 28
type = 29
entity-name = 31
reg-id = 32
role = 33
thumbprint = 34
date = 35
device-id = 36
artifact = 37
href = 38
ownership = 39
rel = 40
media-type = 41
use = 42
activation-status = 43
channel-type = 44
colloquial-version = 45
description = 46
edition = 47
entitlement-data-required = 48
entitlement-key = 49
generator = 50
persistent-id = 51
product = 52
product-family = 53
revision = 54
summary = 55
unspsc-code = 56
unspsc-version = 57

multipartnumeric = 1
multipartnumeric-suffix = 2
alphanumeric = 3
decimal = 4
semver = 16384

tag-creator=1
software-creator=2
aggregator=3
distributor=4
```

```
licensor=5
maintainer=6

shared=1
private=2
abandon=3

ancestor=1
component=2
feature=3
installationmedia=4
packageinstaller=5
parent=6
patches=7
requires=8
see-also=9
supersedes=10

optional=1
required=2
recommended=3

comid.language = 0
comid.tag-identity = 1
comid.entity = 2
comid.linked-tags = 3
comid.triples = 4

comid.tag-id = 0
comid.tag-version = 1

comid.entity-name = 0
comid.reg-id = 1
comid.role = 2

comid.linked-tag-id = 0
comid.tag-rel = 1

comid.reference-triples = 0
comid.endorsed-triples = 1
comid.identity-triples = 2
comid.attest-key-triples = 3

comid.class = 0
comid.instance = 1
comid.group = 2

comid.class-id = 0
comid.vendor = 1
comid.model = 2
```

```
comid.layer = 3
comid.index = 4

comid.mkey = 0
comid.mval = 1

comid.ver = 0
comid.svn = 1
comid.digests = 2
comid.flags = 3
comid.raw-value = 4
comid.raw-value-mask = 5
comid.mac-addr = 6
comid.ip-addr = 7
comid.serial-number = 8
comid.ueid = 9
comid.uuid = 10

comid.key = 0
comid.keychain = 1

comid.version = 0
comid.version-scheme = 1

comid.supplements = 0

comid.replaces = 1

comid.tag-creator = 0
comid.creator = 1
comid.maintainer = 2


corim.id = 0
corim.tags = 1
corim.dependent-rims = 2
corim.profile = 3

corim.href = 0
corim.thumbprint = 1

corim.alg-id = 1
corim.content-type = 3
corim.issuer-key-id = 4
corim.meta = 8

corim.not-before = 0
corim.not-after = 1

corim.signer = 0
```

```
corim.validity = 1

corim.entity-name = 0
corim.reg-id = 1
corim.role = 2

corim.manifest-creator = 1

corim.manifest-signer = 2


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



cose-label = int / tstr
cose-values = any

$entity-name-type-choice /= text

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

uuid-type = bytes .size 16


<CODE ENDS>
```

5. Privacy Considerations

Privacy Considerations

6. Security Considerations

Security Considerations

7. IANA Considerations

This document has a number of IANA considerations, as described in the following subsections. In summary, 6 new registries are established with this request, with initial entries provided for each registry. New values for 5 other registries are also requested.

7.1. COSE Header Parameters Registry

The 'corim metadata' parameter has been added to the "COSE Header Parameters" registry:

- *Name: 'corim metadata'

*Label: 11

*Value: corim-meta-map

*Description: Provides a map of additional metadata for a CoRIM payload composed of (1) one or more entities that created or signed the corresponding CoRIM and (2) its period of validity

*Reference: 'corim-meta-map' in {model-corim-meta-map} of this document

7.2. CoRIM Map Items Registry

This document defines a new registry titled "CoRIM Map". The registry uses integer values as index values for items in 'unsigned-corim-map' CBOR maps.

Future registrations for this registry are to be made based on [RFC8126] as follows:

Range	Registration Procedures
0-127	Standards Action
128-255	Specification Required

Table 3: CoRIM Map Items Registration Procedures

All negative values are reserved for Private Use.

Initial registrations for the "CoRIM Map" registry are provided below. Assignments consist of an integer index value, the item name, and a reference to the defining specification.

Index	Item Name	Specification
0	corim.id	RFC-AAAA
1	corim.tags	RFC-AAAA
2	corim.dependent-rims	RFC-AAAA
3-255	Unassigned	

Table 4: CoRIM Map Items Initial Registrations

7.3. CoRIM Entity-Map Items Registry

This document defines a new registry titled "CoRIM Entity Map". The registry uses integer values as index values for items in 'corim-enentity-map' CBOR maps.

Future registrations for this registry are to be made based on [RFC8126] as follows:

Range	Registration Procedures
0-127	Standards Action
128-255	Specification Required

Table 5: CoRIM Entity Map Items Registration Procedures

All negative values are reserved for Private Use.

Initial registrations for the "CoRIM Entity Map" registry are provided below. Assignments consist of an integer index value, the item name, and a reference to the defining specification.

Index	Item Name	Specification
0	corim.entity-name	RFC-AAAA
1	corim.reg-id	RFC-AAAA
2	corim.role	RFC-AAAA
3-255	Unassigned	

Table 6: CoRIM Enity Map Items Initial Registrations

7.4. CoRIM Entity-Types Registry

This document defines a new registry titled "CoRIM Entity Types". The registry maintains well-defined integer values as choices for '$entity-name-type-choice' CBOR uints.

Future registrations for this registry are to be made based on [RFC8126] as follows:

Range	Registration Procedures
0-127	Standards Action
128-255	Specification Required

Table 7: CoRIM Entity Types Registration Procedures

All negative values are reserved for Private Use.

Initial registrations for the "CoRIM Entity Types" registry are provided below. Assignments consist of an integer value, the item name, and a reference to the defining specification.

Index	Item Name	Specification
0	corim.manifest-creator	RFC-AAAA
1	corim.manifest-signer	RFC-AAAA
2-255	Unassigned	

Table 8: CoRIM Entity Types Initial Registrations

7.5. CoMID Map Items Registry

This document defines a new registry titled "CoMID Map". The registry uses integer values as index values for items in 'concise-mid-tag' CBOR maps.

Future registrations for this registry are to be made based on [RFC8126] as follows:

Range	Registration Procedures
0-127	Standards Action
128-255	Specification Required

Table 9: CoMID Map Items Registration Procedures

All negative values are reserved for Private Use.

Initial registrations for the "CoMID Map" registry are provided below. Assignments consist of an integer index value, the item name, and a reference to the defining specification.

Index	Item Name	Specification
0	comid.language	RFC-AAAA
1	comid.tag-identity	RFC-AAAA
2	comid.entity	RFC-AAAA
3	comid.linked-tags	RFC-AAAA
4	comid.triples	RFC-AAAA
5-255	Unassigned	

Table 10: CoMID Map Items Initial Registrations

7.6. CoMID Entity-Map Items Registry

This document defines a new registry titled "CoMID Entity Map". The registry uses integer values as index values for items in 'comid-entity-map' CBOR maps.

Future registrations for this registry are to be made based on [RFC8126] as follows:

Range	Registration Procedures
0-127	Standards Action
128-255	Specification Required

Table 11: CoMID Entity Map Items Registration Procedures

All negative values are reserved for Private Use.

Initial registrations for the "CoMID Entity Map" registry are provided below. Assignments consist of an integer index value, the item name, and a reference to the defining specification.

Index	Item Name	Specification
0	comid.entity-name	RFC-AAAA
1	comid.reg-id	RFC-AAAA
2	comid.role	RFC-AAAA
3-255	Unassigned	

Table 12: CoMID Entity Map Items Initial Registrations

7.7. CoMID Triples-Map Items Registry

This document defines a new registry titled "CoMID Triples Map". The registry uses integer values as index values for items in 'comid-triples-map' CBOR maps.

Future registrations for this registry are to be made based on [RFC8126] as follows:

Range	Registration Procedures
0-127	Standards Action
128-255	Specification Required

Table 13: CoMID triples Map Items Registration Procedures

All negative values are reserved for Private Use.

Initial registrations for the "CoMID Triples Map" registry are provided below. Assignments consist of an integer index value, the item name, and a reference to the defining specification.

Index	Item Name	Specification
0	comid.reference-triples	RFC-AAAA
1	comid.endorsed-triples	RFC-AAAA
2	comid.identity-triples	RFC-AAAA
3	comid.attest-key-triples	RFC-AAAA
4-255	Unassigned	

Table 14: CoMID Triples Map Items Initial Registrations

7.8. CoMID Measurement-Values-Map Items Registry

This document defines a new registry titled "CoMID Measurement-Values Map". The registry uses integer values as index values for items in 'comid-measurement-values-map' CBOR maps.

Future registrations for this registry are to be made based on [RFC8126] as follows:

Range	Registration Procedures
0-127	Standards Action
128-255	Specification Required

Table 15: CoMID Measurement-Values Map Items Registration Procedures

All negative values are reserved for Private Use.

Initial registrations for the "CoMID Measurement-Values Map" registry are provided below. Assignments consist of an integer index value, the item name, and a reference to the defining specification.

Index	Item Name	Specification
0	comid.ver	RFC-AAAA
1	comid.svn	RFC-AAAA
2	comid.digests	RFC-AAAA
3	comid.flags	RFC-AAAA
4	comid.raw-value	RFC-AAAA
5	comid.raw-value-mask	RFC-AAAA
6	comid.mac-addr	RFC-AAAA
7	comid.ip-addr	RFC-AAAA
8	comid.serial-number	RFC-AAAA
9	comid.ueid	RFC-AAAA
10	comid.uuid	RFC-AAAA
11-255	Unassigned	

Table 16: CoMID Measurement-Values Map Items Initial Registrations

7.9. CoMID Tag-Relationship-Types Registry

This document defines a new registry titled "CoMID Tag-Relationship Types". The registry maintains well-defined integer values as choices for '$tag-rel-type-choice' CBOR uints.

Future registrations for this registry are to be made based on [RFC8126] as follows:

Range	Registration Procedures
0-127	Standards Action
128-255	Specification Required

Table 17: CoMID Tag-Relationship Types Registration Procedures

All negative values are reserved for Private Use.

Initial registrations for the "CoMID Tag-Relationship Types" registry are provided below. Assignments consist of an integer value, the item name, and a reference to the defining specification.

Index	Item Name	Specification
0	comid.supplements	RFC-AAAA
1	comid.replaces	RFC-AAAA
2-255	Unassigned	

Table 18: CoMID Tag-Relationship Types Initial Registrations

7.10. CoMID Role-Types Registry

This document defines a new registry titled "CoMID Role Types". The registry maintains well-defined integer values as choices for '$comid-role-type-choice' CBOR uints.

Future registrations for this registry are to be made based on [RFC8126] as follows:

Range	Registration Procedures
0-127	Standards Action
128-255	Specification Required

Table 19: CoMID Role Types Registration Procedures

All negative values are reserved for Private Use.

Initial registrations for the "CoMID Role Types" registry are provided below. Assignments consist of an integer value, the item name, and a reference to the defining specification.

Index	Item Name	Specification
0	comid.tag-creator	RFC-AAAA
1	comid.creator	RFC-AAAA
2	comid.maintainer	RFC-AAAA
3-255	Unassigned	

Table 20: CoMID Role Types Initial Registrations

7.11. rim+cbor Media Type Registration

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

Type name: application

Subtype name: rim+cbor

Required parameters: none

Optional parameters: none

Encoding considerations: Must be encoded as using [RFC8949]. See RFC-AAAA for details.

Security considerations: See Section 6 of RFC-AAAA.

Interoperability considerations: Applications MAY ignore any key value pairs that they do not understand. This allows backwards compatible extensions to this specification.

Published specification: RFC-AAAA

Applications that use this media type: The type is used by remote attestation procedures, supply chain integrity management systems, vulnerability assessment systems, and in applications that rely on trustworthy endorsements and reference values describing the intended operational state of a system.

Fragment identifier considerations: Fragment identification for application/rim+cbor is supported by using fragment identifiers as specified by Section 9.5 of [RFC8949].

Additional information:

Magic number(s): first five bytes in hex: 43 4f 52 49 4d

File extension(s): corim

Macintosh file type code(s): none

Macintosh Universal Type Identifier code: org.ietf.corim conforms to public.data

Person & email address to contact for further information: Henk Birkholz <henk.birkholz@sit.fraunhofer.de>

Intended usage: COMMON

Restrictions on usage: None

Author: Henk Birkholz <henk.birkholz@sit.fraunhofer.de>

Change controller: IESG

7.12. CoAP Content-Format Registration

IANA is requested to assign a CoAP Content-Format ID for the CoRIM media type in the "CoAP Content-Formats" sub-registry, from the "IETF Review or IESG Approval" space (256..999), within the "CoRE Parameters" registry [RFC7252] [IANA.core-parameters]:

Media type	Encoding	ID	Reference
application/rim+cbor	-	TBD1	RFC-AAAA

Table 21: CoAP Content-Format IDs

7.13. CoRIM CBOR Tag Registration

IANA is requested to allocate tags in the "CBOR Tags" registry [IANA.cbor-tags], preferably with the specific value requested:

Tag	Data Item	Semantics
500	tagged array or tagged map	Concise Reference Integrity Manifest (CoRIM) [RFC-AAAA]
501	map	unsigned CoRIM [RFC-AAAA]
502	array	signed CoRIM [RFC-AAAA]
505	bstr	byte string with CBOR-encoded Concise SWID tag [RFC-AAAA]
506	bstr	byte string with CBOR-encoded Concise MID tag [RFC-AAAA]

Table 22: CoRIM CBOR Tags

7.14. CoMID CBOR Tag Registration

IANA is requested to allocate tags in the "CBOR Tags" registry [IANA.cbor-tags], preferably with the specific value requested:

Tag	Data Item	Semantics
550	bstr	UEID with max size of 33 bytes [RFC-AAAA]
551	int	Security Version Number [RFC-AAAA]
552	int	lower bound of allowed Security Version Number [RFC-AAAA]

Table 23: CoMID CBOR Tags

8. References

8.1. 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-12, 23 April 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-architecture-12>.

[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-18, 12 July 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-sacm-coswid-18>.

[IANA.cbor-tags] IANA, "Concise Binary Object Representation (CBOR) Tags", <http://www.iana.org/assignments/cbor-tags>.

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

[IANA.language-subtag-registry] IANA, "Language Subtag Registry", <http://www.iana.org/assignments/language-subtag-registry>.

[IANA.media-types] IANA, "Media Types", <http://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>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/rfc/rfc7231>.

[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>.

[RFC8126] 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>.

[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>.

[RFC8949] 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>.

8.2. Informative References

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/rfc/rfc4949>.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
64295 Darmstadt
Germany

Email: henk.birkholz@sit.fraunhofer.de

Thomas Fossati
Arm Limited
United Kingdom

Email: Thomas.Fossati@arm.com

Yogesh Deshpande
Arm Limited
United Kingdom

Email: yogesh.deshpande@arm.com

Ned Smith
Intel Corporation
United States of America

Email: ned.smith@intel.com

Wei Pan
Huawei Technologies

Email: william.panwei@huawei.com