Concise Software Identification Tags
draft-ietf-sacm-coswid-10

Abstract

This document defines a concise representation of ISO/IEC 19770-2:2015 Software Identification (SWID) tags that are interoperable with the XML schema definition of ISO/IEC 19770-2:2015. Next to the inherent capability of SWID tags to express arbitrary context information, Concise SWID (CoSWID) tags support the definition of additional semantics via well-defined data definitions incorporated by extension points.

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 December 26, 2019.

Copyright Notice

Copyright (c) 2019 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
Table of Contents

1. Introduction .................................................. 3
  1.1. The SWID and CoSWID Tag Lifecycle ..................... 4
  1.2. Concise SWID Format ...................................... 6
  1.3. Requirements Notation ..................................... 7
2. Concise SWID Data Definition ................................ 7
  2.1. Concise SWID Extensions ................................. 8
  2.2. The concise-swid-tag Group ............................... 9
  2.3. concise-swid-tag Co-constraints .......................... 13
  2.4. The global-attributes Group ............................. 14
  2.5. The entity-entry Group ................................... 15
  2.6. The link-entry Map ........................................ 17
  2.7. The software-meta-entry Map ............................. 21
  2.8. The Resource Collection Definition .................... 24
    2.8.1. The hash-entry Array .................................. 24
    2.8.2. The resource-collection Group ......................... 24
    2.8.3. The payload-entry Group ............................... 27
    2.8.4. The evidence-entry Group .............................. 28
  2.9. Full CDDL Definition ...................................... 28
3. Determining the Type of CoSWID ......................... 34
4. CoSWID Indexed Label Values ................................ 34
  4.1. Version Scheme ........................................... 34
  4.2. Entity Role Values ....................................... 35
  4.3. Link Ownership Values ................................... 36
  4.4. Link Rel Values .......................................... 37
  4.5. Link Use Values .......................................... 39
5. IANA Considerations ........................................ 39
  5.1. CoSWID Items Registry ................................... 39
  5.2. SWID/CoSWID Value Registries ......................... 42
    5.2.1. SWID/CoSWID Version Scheme Value Registry ....... 43
    5.2.2. SWID/CoSWID Entity Role Value Registry ........... 44
    5.2.3. SWID/CoSWID Link Ownership Value Registry ....... 45
    5.2.4. SWID/CoSWID Link Relationship Value Registry .... 46
    5.2.5. SWID/CoSWID Link Use Value Registry ............... 47
  5.3. swid+cbor Media Type Registration .................... 48
  5.4. CoAP Content-Format Registration ...................... 49
  5.5. CBOR Tag Registration ................................... 50
6. Security Considerations ...................................... 50
7. Acknowledgments .............................................. 52
1. Introduction

SWID tags have several use-applications including but not limited to:

- Software Inventory Management, a part of a Software Asset Management [SAM] process, which requires an accurate list of discernible deployed software components.

- Vulnerability Assessment, which requires a semantic link between standardized vulnerability descriptions and software components installed on IT-assets [X.1520].

- Remote Attestation, which requires a link between reference integrity measurements (RIM) and security logs of measured software components [I-D.birkholz-rats-tuda].

SWID tags, as defined in ISO-19770-2:2015 [SWID], provide a standardized XML-based record format that identifies and describes a specific release of a software component. Different software components, and even different releases of a particular software component, each have a different SWID tag record associated with them. SWID tags are meant to be flexible and able to express a broad set of metadata about a software component.

While there are very few required fields in SWID tags, there are many optional fields that support different use scenarios. A SWID tag consisting of only required fields might be a few hundred bytes in size; however, a tag containing many of the optional fields can be many orders of magnitude larger. Thus, real-world instances of SWID tags can be fairly large, and the communication of SWID tags in use-applications, such as those described earlier, can cause a large amount of data to be transported. This can be larger than acceptable for constrained devices and networks. Concise SWID (CoSWID) tags significantly reduce the amount of data transported as compared to a typical SWID tag. This reduction is enabled through the use of CBOR, which maps the human-readable labels of SWID data items to more concise integer labels (indices). The use of CBOR to express SWID information in CoSWID tags allows both CoSWID and SWID tags to be part of an enterprise security solution for a wider range of endpoints and environments.
1.1. The SWID and CoSWID Tag Lifecycle

In addition to defining the format of a SWID tag record, ISO/IEC 19770-2:2015 defines requirements concerning the SWID tag lifecycle. Specifically, when a software component is installed on an endpoint, that software component’s SWID tag is also installed. Likewise, when the software component is uninstalled or replaced, the SWID tag is deleted or replaced, as appropriate. As a result, ISO/IEC 19770-2:2015 describes a system wherein there is a correspondence between the set of installed software components on an endpoint, and the presence of the corresponding SWID tags for these components on that endpoint. CoSWIDs share the same lifecycle requirements as a SWID tag.

The SWID specification and supporting guidance provided in NIST Internal Report (NISTIR) 8060: Guidelines for the Creation of Interoperable SWID Tags [SWID-GUIDANCE] defines four types of SWID tags: primary, patch, corpus, and supplemental.

1. Primary Tag - A SWID or CoSWID tag that identifies and describes a software component is installed on a computing device. A primary tag is intended to be installed on an endpoint along with the corresponding software component.

2. Patch Tag - A SWID or CoSWID tag that identifies and describes an installed patch which has made incremental changes to a software component installed on an endpoint. A patch tag is intended to be installed on an endpoint along with the corresponding software component patch.

3. Corpus Tag - A SWID or CoSWID tag that identifies and describes an installable software component in its pre-installation state. A corpus tag can be used to represent metadata about an installation package or installer for a software component, a software update, or a patch.

4. Supplemental Tag - A SWID or CoSWID tag that allows additional information to be associated with a referenced SWID tag. This helps to ensure that SWID Primary and Patch Tags provided by a software provider are not modified by software management tools, while allowing these tools to provide their own software metadata.

The type of a tag is determined by specific data elements, which is discussed in Section 3.

Corpus, primary, and patch tags have similar functions in that they describe the existence and/or presence of different types of
software (e.g., software installers, software installations, software patches), and, potentially, different states of software components. In contrast, supplemental tags furnish additional information not contained in corpus, primary, or patch tags. All four tag types come into play at various points in the software lifecycle, and support software management processes that depend on the ability to accurately determine where each software component is in its lifecycle.

```
+------------+                          
v            |                          
Installation Media -> Installed -> Patched -> Upgraded -> Removed Deployed
Corpus       Primary            Primary xPrimary xPrimary
Supplemental Supplemental xSupplemental xSupplemental
Patch        xPatch
Primary      xPrimary
Supplemental
```

Figure 1: Use of Tag Types in the Software Lifecycle

Figure 1 illustrates the steps in the software lifecycle and the relationships among those lifecycle events supported by the four types of SWID and CoSWID tags, as follows:

* Software Deployment. Before the software component is installed (i.e., pre-installation), and while the product is being deployed, a corpus tag provides information about the installation files and distribution media (e.g., CD/DVD, distribution package).

* Software Installation. A primary tag will be installed with the software component (or subsequently created) to uniquely identify and describe the software component. Supplemental tags are created to augment primary tags with additional site-specific or extended information. While not illustrated in the figure, patch tags can also be installed during software installation to provide information about software fixes deployed along with the base software installation.

* Software Patching. When a new patch is applied to the software component, a new patch tag is provided, supplying details about the patch and its dependencies. While not illustrated in the
Software Upgrading. As a software component is upgraded to a new version, new primary and supplemental tags replace existing tags, enabling timely and accurate tracking of updates to software inventory. While not illustrated in the figure, a corpus tag can also provide information about the upgrade installer, and dependencies that need to be installed before the upgrade.

* Software Removal. Upon removal of the software component, relevant SWID tags are removed. This removal event can trigger timely updates to software inventory reflecting the removal of the product and any associated patch or supplemental tags.

Note: While not fully illustrated in the figure, supplemental tags can be associated with any corpus, primary, or patch tag to provide additional metadata about an installer, installed software, or installed patch respectively.

Understanding the use of CoSWIDs in the software lifecycle provides a basis for understanding the information provided in a CoSWID and the associated semantics of this information. Each of the different SWID and CoSWID tag types provide different sets of information. For example, a "corpus tag" is used to describe a software component’s installation image on an installation media, while a "patch tag" is meant to describe a patch that modifies some other software component.

1.2. Concise SWID Format

This document defines the CoSWID tag format, a more concise representation of SWID information in the Concise Binary Object Representation (CBOR) [RFC7049]. The structure of a CoSWID is described via the Concise Data Definition Language (CDDL) [RFC8610]. The resulting CoSWID data definition is aligned to the information able to be expressed with the XML schema definition of ISO-19770-2:2015 [SWID]. This alignment allows both SWID and CoSWID tags to represent a common set of SWID information and to support all SWID tag use cases. To achieve this end, the CDDL representation includes every SWID tag field and attribute.

The vocabulary, i.e., the CDDL names of the types and members used in the CoSWID data definition, are mapped to more concise labels represented as small integer values. The names used in the CDDL data definition and the mapping to the CBOR representation using integer
labels is based on the vocabulary of the XML attribute and element names defined in ISO/IEC 19770-2:2015.

1.3. 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 SWID Data Definition

The following is a CDDL representation for a CoSWID tag. The CamelCase notation used in the XML schema definition is changed to a hyphen-separated notation [KebabCase] (e.g. ResourceCollection is named resource-collection) in the CoSWID data definition. In essence, [KebabCase] "looks-like-this". This deviation from the original notation used in the XML representation reduces ambiguity when referencing certain attributes in corresponding textual descriptions. An attribute referred to by its name in CamelCase notation explicitly relates to XML SWID tags; an attribute referred to by its name in KebabCase notation explicitly relates to CoSWID tags. This approach simplifies the composition of further work that reference both XML SWID and CoSWID documents.

Human-readable labels of members in CDDL map data definitions are mapped to integer indices via a block of rules at the bottom of the definition. The 57 character strings of the SWID vocabulary are replacesm which would have to be stored or transported in full if using the original vocabulary.

In CBOR, an array is encoded using bytes that identify the array, and the array’s length or stop point (see [RFC7049]). To make items that support 1 or more values, the following CDDL notion is used.

_\text{name}_ = (_\text{label}_: _\text{data}_ / \ [ 2* _\text{data}_ \ ])

The CDDL rule above allows for a more efficient CBOR encoding of the data when a single value is used. This is accomplished by avoiding the need to first encode the array. Conversely, hen two or more values are encoded, the bytes identifying the array are used. This modeling pattern is used frequently in the CoSWID CDDL data definition to allow for more efficient encoding of single values.

The following subsections describe the different parts of the CoSWID model.
2.1. Concise SWID Extensions

The corresponding CoSWID data definition includes two kinds of augmentation.

- The explicit definition of types for attributes that are typically stored in the "any attribute" of an ISO-19770-2:2015 in XML representation. These are covered in Section 2.4.

- The inclusion of extension points in the CoSWID data definition using CDDL sockets (see [RFC8610] section 3.9). The use of CDDL sockets allow for well-formed extensions to be defined in supplementary CDDL descriptions that support additional uses of CoSWID tags that go beyond the original scope of ISO-19770-2:2015 tags. This extension mechanism can also be used to update the CoSWID format as revisions to ISO-19770-2 are published.

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

<table>
<thead>
<tr>
<th>Map Name</th>
<th>CDDL Socket</th>
<th>Defined in</th>
</tr>
</thead>
<tbody>
<tr>
<td>concise-swid-tag</td>
<td>$$coswid-extension</td>
<td>Section 2.2</td>
</tr>
<tr>
<td>entity-entry</td>
<td>$$entity-extension</td>
<td>Section 2.5</td>
</tr>
<tr>
<td>link-entry</td>
<td>$$link-extension</td>
<td>Section 2.6</td>
</tr>
<tr>
<td>software-meta-entry</td>
<td>$$meta-extension</td>
<td>Section 2.7</td>
</tr>
<tr>
<td>file-entry</td>
<td>$$file-extension</td>
<td>Section 2.8.2</td>
</tr>
<tr>
<td>directory-entry</td>
<td>$$directory-extension</td>
<td>Section 2.8.2</td>
</tr>
<tr>
<td>process-entry</td>
<td>$$process-extension</td>
<td>Section 2.8.2</td>
</tr>
<tr>
<td>resource-entry</td>
<td>$$resource-extension</td>
<td>Section 2.8.2</td>
</tr>
<tr>
<td>payload-entry</td>
<td>$$payload-extension</td>
<td>Section 2.8.3</td>
</tr>
<tr>
<td>evidence-entry</td>
<td>$$evidence-extension</td>
<td>Section 2.8.4</td>
</tr>
</tbody>
</table>

The CoSWID Items Registry defined in Section 5.1 provides a registration mechanism allowing new items, and their associated index values, to be added to the CoSWID model through the use of the CDDL
sockets described above. This registration mechanism provides for well-known index values for data items in CoSWID extensions, allowing these index values to be recognized by implementations supporting a given extension.

The following CDDL sockets defined in this document allow for adding new values to corresponding type-choices (i.e. to represent enumerations) via custom CDDL data definitions.

<table>
<thead>
<tr>
<th>Enumeration Name</th>
<th>CDDL Socket</th>
<th>Defined in</th>
</tr>
</thead>
<tbody>
<tr>
<td>version-scheme</td>
<td>$version-scheme</td>
<td>Section 4.1</td>
</tr>
<tr>
<td>role</td>
<td>$role</td>
<td>Section 4.2</td>
</tr>
<tr>
<td>ownership</td>
<td>$ownership</td>
<td>Section 4.3</td>
</tr>
<tr>
<td>rel</td>
<td>$rel</td>
<td>Section 4.4</td>
</tr>
<tr>
<td>use</td>
<td>$use</td>
<td>Section 4.5</td>
</tr>
</tbody>
</table>

A number of SWID/CoSWID value registries are also defined in Section 5.2 that allow new valid values to be registered with IANA for the enumerations above. This registration mechanism supports the definition of new well-known index values and names for new enumeration values used by SWID and CoSWID. This registration mechanism allows new standardized enumerated values to be shared between both specifications (and implementations) over time.

### 2.2. The concise-swid-tag Group

The CDDL data definition for the root concise-swid-tag map is as follows and this rule and its constraints MUST be followed when creating or validating a CoSWID tag:
concise-swid-tag = {
  global-attributes,
  tag-id => text,
  tag-version => integer,
  ? corpus => bool,
  ? patch => bool,
  ? supplemental => bool,
  software-name => text,
  ? software-version => text,
  ? version-scheme => $version-scheme,
  ? media => text,
  ? software-meta => software-meta-entry / [ 2* software-meta-entry ],
  entity => entity-entry / [ 2* entity-entry ],
  ? link => link-entry / [ 2* link-entry ],
  ? (( payload => payload-entry ) // ( evidence => evidence-entry )),
  * $$coswid-extension
}

tag-id = 0
software-name = 1
entity = 2
evidence = 3
link = 4
software-meta = 5
payload = 6
corpus = 8
patch = 9
media = 10
supplemental = 11
tag-version = 12
software-version = 13
version-scheme = 14

$version-scheme /= multipartnumeric
$version-scheme /= multipartnumeric-suffix
$version-scheme /= alphanumeric
$version-scheme /= decimal
$version-scheme /= semver
$version-scheme /= uint / text
multipartnumeric = 1
multipartnumeric-suffix = 2
alphanumeric = 3
decimal = 4
semver = 16384

The following describes each member of the concise-swid-tag root map.
- global-attributes: A list of items including an optional language
definition to support the processing of text-string values and an
unbounded set of any-attribute items. Described in Section 2.4.

- tag-id (index 0): A textual identifier uniquely referencing a
  (composite) software component. The tag identifier MUST be
globally unique. There are no strict guidelines on how this
identifier is structured, but examples include a 16 byte GUID
(e.g. class 4 UUID) [RFC4122], or a text string appended to a DNS
domain name to ensure uniqueness across organizations.

- tag-version (index 12): An integer value that indicate the
  specific release revision of the tag. Typically, the initial
  value of this field is set to 0, and the value is monotonically
  increased for subsequent tags produced for the same software
  component release. This value allows a CoSWID tag producer to
correct an incorrect tag previously released, without indicating a
change to the underlying software component the tag represents.
For example, the tag version would be changed to add new metadata,
to correct a broken link, to add a missing payload entry, etc.
When producing a revised tag, the new tag-version value MUST be
greater than the old tag-version value.

- corpus (index 8): A boolean value that indicates if the tag
  identifies and describes an installable software component in its
  pre-installation state. Installable software includes a
  installation package or installer for a software component, a
  software update, or a patch. If the CoSWID tag represents
  installable software, the corpus item MUST be set to "true". If
  not provided the default value MUST be considered "false".

- patch (index 9): A boolean value that indicates if the tag
  identifies and describes an installed patch which has made
  incremental changes to a software component installed on a
  computing device. Typically, an installed patch has made a set of
  file modifications to pre-installed software, and does not alter
  the version number or the descriptive metadata of an installed
  software component. If a CoSWID tag is for a patch, the patch
  item MUST be set to "true". If not provided the default value
  MUST be considered "false".

- supplemental (index 11): A boolean value that indicates if the tag
  is providing additional information to be associated with another
  referenced SWID or CoSWID tag. Tags using this item help to
  ensure that primary and patch tags provided by a software provider
  are not modified by software management tools, while allowing
  these tools to provide their own software metadata for a software
  component. If a CoSWID tag is a supplemental tag, the
supplemental item MUST be set to "true". If not provided the default value MUST be considered "false".

- software-name (index 1): This textual item provides the software component’s name. This name is likely the same name that would appear in a package management tool.

- software-version (index 13): A textual value representing the specific release or development version of the software component.

- version-scheme (index 14): An 8-bit integer or textual value representing the versioning scheme used for the software-version item. If an integer value is used it MUST be a value from the SWID/CoSWID Version Scheme Value Registry (see section Section 5.2.1 or a value in the private use range: 32768-65535.

An initial set of pre-defined version-scheme index and text values are defined in Section 4.1 that are based on the version-scheme values defined in [SWID]. These pre-defined version-scheme values are registered with IANA in the "SWID/CoSWID Version Scheme Value" registry Section 5.2.1, and MAY be expanded in the future.

The value of an ownership item MUST be one of the following:

* The index (preferred) or string value of a role from the IANA in the "SWID/CoSWID Version Scheme Value" registry.

* An index value in the range 32768 through 65535, to indicate that a private use index value is used.

* A string value prefixed with "x_", to indicate that a private use string value is used.

- media (index 10): This text value is a hint to the tag consumer to understand what this tag applies to. This item represents a query as defined by the W3C Media Queries Recommendation (see [W3C.REC-css3-mediaqueries-20120619]).

- software-meta (index 5): An open-ended map of key/value data pairs. A number of predefined keys can be used within this item providing for common usage and semantics across the industry. Use of this map allows any additional attribute to be included in the tag. It is expected that industry groups will use a common set of attribute names to allow for interoperability within their communities. Described in Section 2.7.

- entity (index 2): Provides information about one or more organizations responsible for producing the CoSWID tag, and
producing or releasing the software component referenced by this CoSWID tag. Described in Section 2.5.

- **link (index 4):** Provides a means to establish relationship arcs between the tag and another items. A given link can be used to establish the relationship between tags or to reference another resource that is related to the CoSWID tag, e.g. vulnerability database association, ROLIE feed [RFC8322], MUD resource [RFC8520], software download location, etc). This is modeled after the HTML "link" element. Described in Section 2.6.

- **payload (index 6):** This item represents the software artifacts that compose the target software. For example, the files included with an installer for a corpus tag or installed on an endpoint when the software component is installed for a primary or patch tag. The artifacts listed in a payload may be a superset of the software artifacts that are actually installed. Based on user selections at install time, an installation might not include every artifact that could be created or executed on the endpoint when the software component is installed or run. Described in Section 2.8.3.

- **evidence-entry (index 3):** This item records the results of a software discovery process used to identify untagged software on an endpoint. This item provides a means to represent indicators for why software is believed to be installed on the endpoint. In such a case, a CoSWID tag is created when the endpoint is scanned by the tool performing the scan. Described in Section 2.8.4.

- **coswid-extension:** This CDDL socket is used to add new information structures to the concise-swid-tag root map. See Section 2.1.

### 2.3. concise-swid-tag Co-constraints

The following co-constraints apply to the information provided by in the concise-swid-tag group.

- The patch and supplemental items MUST NOT both be set to "true".

- If the patch item is set to "true", the tag SHOULD contain at least one link item with the rel(ation) item value of "patches" and an href item specifying an association with the software that was patched.

- If the supplemental item is set to "true", the tag SHOULD contain at least one link item with the rel(ation) item value of
"supplements" and an href item specifying an association with the software that is supplemented.

- If all of the corpus, patch, and supplemental items are "false", or if the corpus item is set to "true", then a software-version item MUST be included with a value set to the version of the software component. This ensures that primary and corpus tags have an identifiable software version.

2.4. The global-attributes Group

The global-attributes group provides a list of items including an optional language definition to support the processing of text-string values and an unbounded set of any-attribute items allowing for additional items to be provided as a general point of extension in the model.

The CDDL for the global-attributes follows:

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

any-attribute = (  
  label => text / int / [ 2* text ] / [ 2* int ]  
)

label = text / int

The following describes each child item of this group.

- lang (index 15): A textual language tag that conforms with IANA "Language Subtag Registry" [RFC5646]. 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.

- any-attribute: This sub-group provides a means to include arbitrary information via label ("key") value pairs. Labels can be either a single integer or text string. Values can be either a single integer or text string, or an array of integers or text strings.
2.5. The entity-entry Group

The CDDL for the entity-entry group follows:

entity-entry = {
   global-attributes,
   entity-name => text,
   ? reg-id => any-uri,
   role => $role / [ 2* $role ],
   ? thumbprint => hash-entry,
   * $entity-extension,
}

entity-name = 31
reg-id = 32
role = 33
thumbprint = 34

$role /= tag-creator
$role /= software-creator
$role /= aggregator
$role /= distributor
$role /= licensor
$role /= uint / text
tag-creator=1
software-creator=2
aggregator=3
distributor=4
licensor=5

The following describes each child item of this group.

- global-attributes: The global-attributes group described in Section 2.4.

- entity-name (index 32): The textual name of the organizational entity claiming the roles specified by the role item for the CoSWID tag.

- reg-id (index 32): The registration id value is intended to uniquely identify a naming authority in a given scope (e.g. global, organization, vendor, customer, administrative domain, etc.) for the referenced entity. The value of an registration ID MUST be a RFC 3986 URI. The scope SHOULD be the scope of an organization. In a given scope, the registration id MUST be used consistently for CoSWID tag production.

- role (index 33): The relationship(s) between the entity, and this tag or the referenced software component. Use of index values
instead of text for these pre-defined roles allows a CoSWID to be more concise.

An initial set of pre-defined role index and text values are defined in Section 4.2 that are based on the roles defined in [SWID]. These pre-defined roles are registered with IANA in the "SWID/CoSWID Entity Role Value" registry Section 5.2.2, and MAY be expanded in the future.

The value of a role item MUST be one of the following:

* The index (preferred) or string value of a role from the IANA in the "SWID/CoSWID Entity Role Value" registry.

* An index value in the range 128 through 255, to indicate that a private use index value is used.

* A string value prefixed with "x_", to indicate that a private use string value is used.

The following additional requirements exist for the use of the "role" item:

* An entity item MUST be provided with the role of "tag-creator" for every CoSWID tag. This indicates the organization that created the CoSWID tag.

* An entity item SHOULD be provided with the role of "software-creator" for every CoSWID tag, if this information is known to the tag creator. This indicates the organization that created the referenced software component.

  o thumbprint (index 34): The value of the thumbprint item provides an integer-based hash algorithm identifier (hash-alg-id) and a byte string value (hash-value) that contains the corresponding hash value (i.e. the thumbprint) of the signing entity’s public key certificate. This provides an indicator of which entity signed the CoSWID tag, which will typically be the tag creator. If the hash-alg-id is not known, then the integer value "0" MUST be used. This ensures parity between the SWID tag specification [SWID], which does not allow an algorithm to be identified for this field. See Section 2.8.1 for more details on the use of the hash-entry data structure.

  o $$entity-extension: This CDDL socket can be used to extend the entity-entry group model. See Section 2.1.
2.6. The link-entry Map

The CDDL for the link-entry map follows:

\[
\text{link-entry} = \{
\begin{align*}
&\text{global-attributes}, \\
&? \text{artifact} => \text{text}, \\
&\text{href} => \text{any-uri}, \\
&? \text{media} => \text{text}, \\
&? \text{ownership} => $\text{ownership}, \\
&\text{rel} => $\text{rel}, \\
&? \text{media-type} => \text{text}, \\
&? \text{use} => $\text{use}, \\
&* $$\text{link-extension},
\end{align*}
\}
\]

media = 10
artifact = 37
href = 38
ownership = 39
rel = 40
media-type = 41
use = 42

$\text{ownership} /= \text{shared}
$\text{ownership} /= \text{private}
$\text{ownership} /= \text{abandon}
$\text{ownership} /= \text{uint} / \text{text}
shared=1
private=2
abandon=3

$\text{rel} /= \text{ancestor}
$\text{rel} /= \text{component}
$\text{rel} /= \text{feature}
$\text{rel} /= \text{installationmedia}
$\text{rel} /= \text{packageinstaller}
$\text{rel} /= \text{parent}
$\text{rel} /= \text{patches}
$\text{rel} /= \text{requires}
$\text{rel} /= \text{see-also}
$\text{rel} /= \text{supersedes}
$\text{rel} /= \text{supplemental}
$\text{rel} /= \text{uint} / \text{text}
ancestor=1
component=2
feature=3
installationmedia=4
packageinstaller=5
The following describes each member of this map.

- **global-attributes**: The global-attributes group described in Section 2.4.

- **artifact** (index: 37): To be used with rel="installation-media", this item value can provide the path to the installer executable or script that can be run to launch the referenced installation. Links with the same artifact name MUST be considered mirrors of each other, allowing the installation media to be acquired from any of the described sources.

- **href** (index 38): A URI for the referenced resource. The "href" item’s value can be, but is not limited to, the following (which is a slightly modified excerpt from [SWID]):

  * If no URI scheme is provided, then the URI is to be interpreted as being relative to the URI of the CoSWID tag. For example, "./folder/supplemental.coswid".

  * a physical resource location with any acceptable URI scheme (e.g., file:/// http:// https:// ftp://)

  * a URI with "coswid:" as the scheme, which refers to another CoSWID by tag-id. This URI would need to be resolved in the context of the endpoint by software that can lookup other CoSWID tags. For example, "coswid:2df9de35-0aff-4a86-ace6-f7dddd1ade4c" references the tag with the tag-id value "2df9de35-0aff-4a86-ace6-f7dddd1ade4c".

  * a URI with "swidpath:" as the scheme, which refers to another CoSIWD via an XPATH query. This URI would need to be resolved in the context of the system entity via software components.
that can lookup other CoSWID tags and select the appropriate
tag based on an XPATH query [W3C.REC-xpath20-20101214].
Examples include:

* `swidpath://SoftwareIdentity[Entity/@regid='http://contoso.com']`
  would retrieve all CoSWID tags that include an entity where the
  regid is "Contoso" or `swidpath://SoftwareIdentity[Meta/@persist
  entId='b0c55172-38e9-4e36-be86-92206ad8eddb']` would match
  CoSWID tags with the persistent-id value "b0c55172-38e9-4e36-be86-92206ad8eddb".

- **media** (index 10): A hint to the consumer of the link to what the
  target item is applicable for. This item represents a query as
  defined by the W3C Media Queries Recommendation (see
  [W3C.REC-css3-mediaqueries-20120619]). See also media defined in
  Section 2.2.

- **ownership** (index 39): Used when the "href" item references another
  software component to indicate the degree of ownership between the
  software component referenced by the COSWID tag and the software
  component referenced by the link.

  An initial set of pre-defined ownership index and text values are
  defined in Section 4.3 that are based on the ownership values
  defined in [SWID]. These pre-defined ownership values are
  registered with IANA in the "SWID/CoSWID Link Ownership Value"
  registry Section 5.2.3, and MAY be expanded in the future.

  The value of an ownership item MUST be one of the following:

  * The index (preferred) or string value of a role from the IANA
    in the "SWID/CoSWID Link Ownership Value" registry.

  * An index value in the range 128 through 255, to indicate that a
    private use index value is used.

  * A string value prefixed with "x_", to indicate that a private
    use string value is used.

- **rel** (index 40): Identifies the relationship between this CoSWID
  and the target resource indicated by the "href" item.

  An initial set of pre-defined rel index and text values are
  defined in Section 4.4 that are based on the rel values defined in
  [SWID]. These pre-defined rel values are registered with IANA in
  the "SWID/CoSWID Link Relationship Value" registry Section 5.2.4,
  and MAY be expanded in the future.
The value of a rel item MUST be one of the following:

* The index (preferred) or string value of a role from the IANA in the "SWID/CoSWID Link Relationship Value" registry.

* An index value in the range 128 through 255, to indicate that a private use index value is used.

* A string value prefixed with "x_", to indicate that a private use string value is used.

* A string value, as defined by [RFC8288], corresponding to a "Relation Name" from the IANA "Link Relation Types" registry: https://www.iana.org/assignments/link-relations/link-relations.xhtml. When a string value defined in the IANA "SWID/CoSWID Link Relationship Value" registry matches a Relation Name defined in the IANA "Link Relation Types" registry, the value in the IANA "SWID/CoSWID Link Relationship Value" registry MUST be used instead, as this relationship has a specialized meaning in the context of a SWID/CoSWID tag.

- media-type (index 41): The media type for the target resource, providing a resource consumer with a hint of what type of resource to expect. Media types are identified by referencing a "Name" from the IANA "Media Types" registry: http://www.iana.org/assignments/media-types/media-types.xhtml.

- use (index 42): Determines if the referenced software component has to be installed before installing the tagged software component.

An initial set of pre-defined use index and text values are defined in Section 4.5 that are based on the use values defined in [SWID]. These pre-defined use values are registered with IANA in the "SWID/CoSWID Link Use Value" registry Section 5.2.5, and MAY be expanded in the future.

The value of an ownership item MUST be one of the following:

* The index (preferred) or string value of a role from the IANA in the "SWID/CoSWID Link Use Value" registry.

* An index value in the range 128 through 255, to indicate that a private use index value is used.

* A string value prefixed with "x_", to indicate that a private use string value is used.
o  $$link-extension: This CDDL socket can be used to extend the link-
entry map model.  See Section 2.1.

2.7. The software-meta-entry Map

The CDDL for the software-meta-entry map follows:

software-meta-entry = {
    global-attributes,
    ? 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,
    * $$meta-extension,
}

activation-status = 43
channel-type = 44
colloquial-version = 45
description = 46
dition = 47
titlement-data-required = 48
titlement-key = 49
generator = 50
persistent-id = 51
product = 52
product-family = 53
revision = 54
summary = 55
unspsc-code = 56
unspsc-version = 57

The following describes each child item of this group.

o  global-attributes: The global-attributes group described in
    Section 2.4.
o activation-status (index 43): A textual value that identifies how the software component has been activated, which might relate to specific terms and conditions for its use (e.g. Trial, Serialized, Licensed, Unlicensed, etc) and relate to an entitlement. This attribute is typically used in supplemental tags as it contains information that might be selected during a specific install.

o channel-type (index 44): A textual value that identifies which sales, licensing, or marketing channel the software component has been targeted for (e.g. Volume, Retail, OEM, Academic, etc). This attribute is typically used in supplemental tags as it contains information that might be selected during a specific install.

o colloquial-version (index 45): A textual value for the software component’s informal or colloquial version. Examples may include a year value, a major version number, or similar value that are used to identify a group of specific software component releases that are part of the same release/support cycle. This version can be the same through multiple releases of a software component, while the software-version specified in the concise-swid-tag group is much more specific and will change for each software component release. This version is intended to be used for string comparison only, and is not intended to be used to determine if a specific value is earlier or later in a sequence.

o description (index 46): A textual value that provides a detailed description of the software component, which can be multiple sentences.

o edition (index 47): A textual value indicating that the software component represents a functional variation of the code base used to support multiple software components. For example, this item can be used to differentiate enterprise, standard, or professional variants of a software component.

o entitlement-data-required (index 48): A boolean value that proof of indicator to determine if accompanying proof of entitlement is needed when a software license reconciliation process is performed.

o entitlement-key (index 49): A vendor-specific textual key that can be used to identify and establish a relationship to an entitlement. Examples of an entitlement-key might include a serial number, product key, or license key). For values that relate to a given software component install (i.e., license key), a supplemental tag will typically contain this information. In
other cases where a general-purpose key can be provided that applies to all possible installs of the software component on different endpoints, a primary tag will typically contain this information.

- **generator (index 50):** The name (or tag-id) of the software component that created the CoSWID tag. If the generating software component has a SWID or CoSWID tag, then the tag-id for the generating software component SHOULD be provided.

- **persistent-id (index 51):** A GUID used to identify a related set of software components. Software components sharing the same persistent-id can be different versions.

- **product (index 52):** A basic name for the software component that can be common across multiple tagged software components (e.g., Apache HTTPD).

- **product-family (index 53):** A textual value indicating the software components overall product family. This should be used when multiple related software components form a larger capability that is installed on multiple different endpoints. For example, some software families may consist of server, client, and shared service components that are part of a larger capability. Email systems, enterprise applications, backup services, web conferencing, and similar capabilities are examples of families.

- **revision (index 54):** A string value indicating an informal or colloquial release version of the software. This value can provide a different version value as compared to the software-version specified in the concise-swid-tag group. This is useful when one or more releases need to have an informal version label that differs from the specific exact version value specified by software-version. Examples can include SP1, RC1, Beta, etc.

- **summary (index 55):** A short description of the software component. This MUST be a single sentence suitable for display in a user interface.

- **unspsc-code (index 56):** An 8 digit UNSPSC classification code for the software component. For more information see, [http://www.unspsc.org/](http://www.unspsc.org/).

- **unspsc-version (index 57):** The version of UNSPSC used to define the unspsc-code value.

- **$\text{meta-extension:}** This CDDL socket can be used to extend the software-meta-entry group model. See Section 2.1.
2.8. The Resource Collection Definition

2.8.1. The hash-entry Array

CoSWID adds explicit support for the representation of hash entries using algorithms that are registered in the IANA "Named Information Hash Algorithm Registry" using the hash-entry member (label 58).

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

The number used as a value for hash-alg-id MUST refer an ID in the "Named Information Hash Algorithm Registry" (see https://www.iana.org/assignments/named-information/named-information.xhtml); other hash algorithms MUST NOT be used. The hash-value MUST represent the raw hash value of the hashed resource generated using the hash algorithm indicated by the hash-alg-id.

2.8.2. The resource-collection Group

A list of items both used in evidence (created by a software discovery process) and payload (installed in an endpoint) content of a CoSWID tag document to structure and differentiate the content of specific CoSWID tag types. Potential content includes directories, files, processes, or resources.

The CDDL for the resource-collection group follows:

resource-collection = {
    ? directory => directory-entry,
    ? file => file-entry,
    ? process => process-entry,
    ? resource => resource-entry,
}

filesystem-item = {
    global-attributes,
    ? key => bool,
    ? location => text,
    fs-name => text,
    ? root => text,
}

path-elements-entry = [ [ * file-entry ],
                         [ * directory-entry ],
                      ]

file-entry = {
    filesystem-item,
"}
? size => integer,
? file-version => text,
? hash => hash-entry,
  * $$file-extension
}
directory-entry = {
  filesystem-item,
  path-elements => path-elements-entry,
  * $$directory-extension
}

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

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

The following describes each member of the groups and maps illustrated above.

- filesystem-item: A list of items both used in representing the nodes of a file-system hierarchy (i.e., directory items that allow one or more directories to be defined in the file structure), and file items that allow one or more files to be specified for a given location.
o  global-attributes: The global-attributes group described in Section 2.4.

o  directory (index 16): A directory item allows child directory and file items to be defined within a directory hierarchy for the software component.

o  file (index 17): A file item allows details about a file to be provided for the software component.

o  process (index 18): A process item allows details to be provided about the runtime behavior of the software component, such as information that will appear in a process listing on a device.

o  resource (index 19): A resource item can be used to provide details about a resource expected to be found on an endpoint, or evidence collected related to the software component.

o  size (index 20): The file’s size in bytes.

o  file-version (index 21): The file’s version as reported by querying information on the file from the operating system.

o  key (index 22): A boolean value indicating if a file or directory is significant or required for the software component to execute or function properly. These are files or directories that can be used to affirmatively determine if the software component is installed on an endpoint.

o  location (index 23): The filesystem path where a file is expected to be located when installed or copied. This path SHOULD be relative to the location of the parent directory item, or if no parent is defined, the location MUST be relative to the location of the CoSWID tag. The location MUST not include a file’s name, which is provided by the fs-name item.

o  fs-name (index 24): The name of the directory or file without any path information.

o  root (index 25): A filesystem-specific name for the root of the filesystem. The location item is considered relative to this location if specified. If not provided, the value provided by the location item is expected to be relative to its parent or the location of the CoSWID tag if no parent is provided.

o  path-elements (index 26): This group provides the ability to define a directory structure for files defined in payload or evidence items.
o process-name (index 27): The software component’s process name as it will appear in an endpoint’s process list.

o pid (index 28): The process ID identified for a running instance of the software component in the endpoint’s process list. This is used as part of the evidence item.

o type (index 29): A string indicating the type of resource.

o resource-collection-extension: This CDDL socket can be used to extend the resource-collection group model. This can be used to add new specialized types of resources. See Section 2.1.

o file-extension: This CDDL socket can be used to extend the file-entry group model. See Section 2.1.

o directory-extension: This CDDL socket can be used to extend the directory-entry group model. See Section 2.1.

o process-extension: This CDDL socket can be used to extend the process-entry group model. See Section 2.1.

o resource-extension: This CDDL socket can be used to extend the group model. See Section 2.1.

o -extension: This CDDL socket can be used to extend the resource-entry group model. See Section 2.1.

2.8.3. The payload-entry Group

The CDDL for the payload-entry group follows:

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

The following describes each child item of this group.

- global-attributes: The global-attributes group described in Section 2.4.

- resource-collection: The resource-collection group described in Section 2.8.2.

- resource-collection-extension: This CDDL socket can be used to extend the payload-entry group model. See Section 2.1.
2.8.4. The evidence-entry Group

The CDDL for the evidence-entry group follows:

```
evidence-entry = {
    global-attributes,
    resource-collection,
    ? date => time,
    ? device-id => text,
    * $$evidence-extension
}
date = 35
device-id = 36
```

The following describes each child item of this group.

- global-attributes: The global-attributes group described in Section 2.4.
- resource-collection: The resource-collection group described in Section 2.8.2.
- date (index 35): The date and time the information was collected pertaining to the evidence item.
- device-id (index 36): The endpoint’s string identifier from which the evidence was collected.
- $$evidence-extension: This CDDL socket can be used to extend the evidence-entry group model. See Section 2.1.

2.9. Full CDDL Definition

In order to create a valid CoSWID document the structure of the corresponding CBOR message MUST adhere to the following CDDL data definition.

```
concise-swid-tag = {
    global-attributes,
    tag-id => text,
    tag-version => integer,
    ? corpus => bool,
    ? patch => bool,
    ? supplemental => bool,
    software-name => text,
    ? software-version => text,
    ? version-scheme => $version-scheme,
    ? media => text,
```
? software-meta => software-meta-entry / [ 2* software-meta-entry ],
entity => entity-entry / [ 2* entity-entry ],
? link => link-entry / [ 2* link-entry ],
? (( payload => payload-entry ) // ( evidence => evidence-entry )),
* $$coswid-extension
}

any-uri = text
label = text / int

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

any-attribute = {
  label => text / int / [ 2* text ] / [ 2* int ]
}

global-attributes = {
  ? lang => text,
  * any-attribute,
}

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

entity-entry = {
  global-attributes,
  entity-name => text,
  ? reg-id => any-uri,
  role => $role / [ 2* $role ],
  ? thumbprint => hash-entry,
  * $$entity-extension,
}

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

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

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

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

software-meta-entry = {
global-attributes,
? 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,
resource-collection = (
  ? directory => directory-entry,
  ? file => file-entry,
  ? process => process-entry,
  ? resource => resource-entry,
  * $$resource-collection-extension
)

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

path-elements-entry = [ [ * file-entry ],
  [ * directory-entry ],
]

directory-entry = {
  filesystem-item,
  path-elements => path-elements-entry,
  * $$directory-extension
}

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

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

filesystem-item = {
  global-attributes,
  ? key => bool,
  ? location => text,
  * $$meta-extension
}
fs-name => text,
? root => text,
}

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

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

; "global map member" integer indexes

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
date = 35
device-id = 36
artifact = 37
type = 36
device-id = 36
artifact = 37
device-id = 36
artifact = 37
hhref = 38
ownership = 39
rel = 40
media-type = 41
use = 42
activation-status = 43
channel-type = 44
colloquial-version = 45
description = 46
date = 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

; "version-scheme" integer indexes
multipartnumeric = 1
multipartnumeric-suffix = 2
alphanumeric = 3
decimal = 4
semver = 16384

; "role" integer indexes
tag-creator=1
software-creator=2
aggregator=3
distributor=4
licensor=5

; ownership integer indexes
shared=1
private=2
abandon=3
The operational model for SWID and CoSWID tags was introduced in Section 1.1, which described four different CoSWID tag types. The following additional rules apply to the use of CoSWID tags to ensure that created tags properly identify the tag type.

The first matching rule MUST determine the type of the CoSWID tag.

1. Primary Tag: A CoSWID tag MUST be considered a primary tag if the corpus, patch, and supplemental items are "false".

2. Supplemental Tag: A CoSWID tag MUST be considered a supplemental tag if the supplemental item is set to "true".

3. Corpus Tag: A CoSWID tag MUST be considered a corpus tag if the corpus item is "true".

4. Patch Tag: A CoSWID tag MUST be considered a patch tag if the patch item is "true".

4. CoSWID Indexed Label Values

4.1. Version Scheme

The following table contains a set of values for use in the concise-swid-tag group’s version-scheme item. These values match the version schemes defined in the ISO/IEC 19770-2:2015 [SWID] specification. Index value indicates the value to use as the version-scheme item’s value. The Version Scheme Name provides human-readable text for the
value. The Definition describes the syntax of allowed values for each entry.

<table>
<thead>
<tr>
<th>Index</th>
<th>Version Scheme Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>multipartnumeric</td>
<td>Numbers separated by dots, where the numbers are interpreted as integers (e.g., 1.2.3, 1.4.5, 1.2.3.4.5.6.7)</td>
</tr>
<tr>
<td>2</td>
<td>multipartnumeric+suffix</td>
<td>Numbers separated by dots, where the numbers are interpreted as integers with an additional textual suffix (e.g., 1.2.3a)</td>
</tr>
<tr>
<td>3</td>
<td>alphanumeric</td>
<td>Strictly a string, sorting is done alphanumerically</td>
</tr>
<tr>
<td>4</td>
<td>decimal</td>
<td>A floating point number (e.g., 1.25 is less than 1.3)</td>
</tr>
<tr>
<td>16384</td>
<td>semver</td>
<td>Follows the [SEMVER] specification</td>
</tr>
</tbody>
</table>

The values above are registered in the IANA "SWID/CoSWID Version Scheme Value" registry defined in section Section 5.2.1. Additional entries will likely be registered over time in this registry. Additionally, the index values 32768 through 65535 have been reserved for private use.

4.2. Entity Role Values

The following table indicates the index value to use for the entity-entry group’s role item (see Section 2.5). These values match the entity roles defined in the ISO/IEC 19770-2:2015 [SWID] specification. The "Index" value indicates the value to use as the role item’s value. The "Role Name" provides human-readable text for the value. The "Definition" describes the semantic meaning of each entry.
<table>
<thead>
<tr>
<th>Index</th>
<th>Role Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>tagCreator</td>
<td>The person or organization that created the containing SWID or CoSWID tag</td>
</tr>
<tr>
<td>2</td>
<td>softwareCreator</td>
<td>From [SAM], &quot;person or organization that creates a software product (3.46) or package&quot;</td>
</tr>
<tr>
<td>3</td>
<td>aggregator</td>
<td>From {{SWID}, &quot;An organization or system that encapsulates software from their own and/or other organizations into a different distribution process (as in the case of virtualization), or as a completed system to accomplish a specific task (as in the case of a value added reseller).&quot;</td>
</tr>
<tr>
<td>4</td>
<td>distributor</td>
<td>From [SWID], &quot;An entity that furthers the marketing, selling and/or distribution of software from the original place of manufacture to the ultimate user without modifying the software, its packaging or its labelling.&quot;</td>
</tr>
<tr>
<td>5</td>
<td>licensor</td>
<td>From [SAM] as &quot;software licensor&quot;, a &quot;person or organization who owns or holds the rights to issue a software license for a specific software package&quot;</td>
</tr>
</tbody>
</table>

The values above are registered in the IANA "SWID/CoSWID Entity Role Value" registry defined in section Section 5.2.2. Additional valid values will likely be registered over time. Additionally, the index values 128 through 255 have been reserved for private use.

4.3. Link Ownership Values

The following table indicates the index value to use for the link-entry group’s ownership item (see Section 2.6). These values match the link ownership values defined in the ISO/IEC 19770-2:2015 [SWID] specification. The "Index" value indicates the value to use as the link-entry group ownership item’s value. The "Ownership Type" provides human-readable text for the value. The "Definition" describes the semantic meaning of each entry.
<table>
<thead>
<tr>
<th>Index</th>
<th>Ownership Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>abandon</td>
<td>If the software component referenced by the CoSWID tag is uninstalled, then the referenced software SHOULD not be uninstalled</td>
</tr>
<tr>
<td>2</td>
<td>private</td>
<td>If the software component referenced by the CoSWID tag is uninstalled, then the referenced software SHOULD be uninstalled as well.</td>
</tr>
<tr>
<td>3</td>
<td>shared</td>
<td>If the software component referenced by the CoSWID tag is uninstalled, then the referenced software SHOULD be uninstalled if no other components sharing the software.</td>
</tr>
</tbody>
</table>

The values above are registered in the IANA "SWID/CoSWID Link Ownership Value" registry defined in section 5.2.3. Additional valid values will likely be registered over time. Additionally, the index values 128 through 255 have been reserved for private use.

4.4. Link Rel Values

The following table indicates the index value to use for the link-entry group’s rel item (see Section 2.6). These values match the link rel values defined in the ISO/IEC 19770-2:2015 [SWID] specification. The "Index" value indicates the value to use as the link-entry group ownership item’s value. The "Relationship Type" provides human-readable text for the value. The "Definition" describes the semantic meaning of each entry.

<table>
<thead>
<tr>
<th>Index</th>
<th>Relationship Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ancestor</td>
<td>The link references a SWID/CoSWID tag for an ancestor of this software. This can be useful to define an upgrade path.</td>
</tr>
<tr>
<td>2</td>
<td>component</td>
<td>The link references a SWID/CoSWID tag for a separate component of this software.</td>
</tr>
<tr>
<td>3</td>
<td>feature</td>
<td>The link references a configurable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>feature of this software, that can be enabled or disabled without changing the installed files.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>4</td>
<td>installationmedia</td>
<td>The link references the installation package that can be used to install this software.</td>
</tr>
<tr>
<td>5</td>
<td>packageinstaller</td>
<td>The link references the installation software needed to install this software.</td>
</tr>
<tr>
<td>6</td>
<td>parent</td>
<td>The link references a SWID/CoSWID tag that is the parent of this SWID/CoSWID tag.</td>
</tr>
<tr>
<td>7</td>
<td>patches</td>
<td>The link references a SWID/CoSWID tag that this software patches. Typically only used for patch SWID/CoSWID tags (see Section 1.1).</td>
</tr>
<tr>
<td>8</td>
<td>requires</td>
<td>The link references a prerequisite for installing this software. A patch SWID/CoSWID tag (see Section 1.1) can use this to represent base software or another patch that needs to be installed first.</td>
</tr>
<tr>
<td>9</td>
<td>see-also</td>
<td>The link references other software that may be of interest that relates to this software.</td>
</tr>
<tr>
<td>10</td>
<td>supersedes</td>
<td>The link references another software that this software replaces. A patch SWID/CoSWID tag (see Section 1.1) can use this to represent another patch that this patch incorporates or replaces.</td>
</tr>
<tr>
<td>11</td>
<td>supplemental</td>
<td>The link references a SWID/CoSWID tag that this tag supplements. Used on supplemental SWID/CoSWID tags (see Section 1.1).</td>
</tr>
</tbody>
</table>

The values above are registered in the IANA "SWID/CoSWID Link Relationship Value" registry defined in section Section 5.2.4. Additional valid values will likely be registered over time.
Additionally, the index values 32768 through 65535 have been reserved for private use.

4.5. Link Use Values

The following table indicates the index value to use for the link-entry group’s use item (see Section 2.6). These values match the link use values defined in the ISO/IEC 19770-2:2015 [SWID] specification. The "Index" value indicates the value to use as the link-entry group use item’s value. The "Use Type" provides human-readable text for the value. The "Definition" describes the semantic meaning of each entry.

<table>
<thead>
<tr>
<th>Index</th>
<th>Use Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>optional</td>
<td>From [SWID], &quot;Not absolutely required; the [Link]’d software is installed only when specified.&quot;</td>
</tr>
<tr>
<td>2</td>
<td>required</td>
<td>From [SWID], &quot;The [Link]’d software is absolutely required for an operation software installation.&quot;</td>
</tr>
<tr>
<td>3</td>
<td>recommended</td>
<td>From [SWID], &quot;Not absolutely required; the [Link]’d software is installed unless specified otherwise.&quot;</td>
</tr>
</tbody>
</table>

The values above are registered in the IANA "SWID/CoSWID Link Use Value" registry defined in section Section 5.2.5. Additional valid values will likely be registered over time. Additionally, the index values 128 through 255 have been reserved for private use.

5. IANA Considerations

This document has a number of IANA considerations, as described in the following subsections.

5.1. CoSWID Items Registry

This document uses integer values as index values in CBOR maps.

This document defines a new a new registry titled "CoSWID Items". Future registrations for this registry are to be made based on [RFC8126] as follows:
### Range | Registration Procedures
---|---
0-32767 | Standards Action
32768-4294967295 | Specification Required

All negative values are reserved for Private Use.

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

<table>
<thead>
<tr>
<th>Index</th>
<th>Item Name</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>tag-id</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>1</td>
<td>software-name</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>2</td>
<td>entity</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>3</td>
<td>evidence</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>4</td>
<td>link</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>5</td>
<td>software-meta</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>6</td>
<td>payload</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>7</td>
<td>hash</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>8</td>
<td>corpus</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>9</td>
<td>patch</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>10</td>
<td>media</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>11</td>
<td>supplemental</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>12</td>
<td>tag-version</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>13</td>
<td>software-version</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>14</td>
<td>version-scheme</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>15</td>
<td>lang</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td></td>
<td>Field</td>
<td>RFC-Version</td>
</tr>
<tr>
<td>---</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>16</td>
<td>directory</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>17</td>
<td>file</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>18</td>
<td>process</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>19</td>
<td>resource</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>20</td>
<td>size</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>21</td>
<td>file-version</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>22</td>
<td>key</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>23</td>
<td>location</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>24</td>
<td>fs-name</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>25</td>
<td>root</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>26</td>
<td>path-elements</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>27</td>
<td>process-name</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>28</td>
<td>pid</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>29</td>
<td>type</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>31</td>
<td>entity-name</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>32</td>
<td>reg-id</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>33</td>
<td>role</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>34</td>
<td>thumbprint</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>35</td>
<td>date</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>36</td>
<td>device-id</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>37</td>
<td>artifact</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>38</td>
<td>href</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>39</td>
<td>ownership</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>40</td>
<td>rel</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>41</td>
<td>media-type</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>42</td>
<td>use</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>43</td>
<td>activation-status</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>44</td>
<td>channel-type</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>45</td>
<td>colloquial-version</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>46</td>
<td>description</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>47</td>
<td>edition</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>48</td>
<td>entitlement-data-required</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>49</td>
<td>entitlement-key</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>50</td>
<td>generator</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>51</td>
<td>persistent-id</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>52</td>
<td>product</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>53</td>
<td>product-family</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>54</td>
<td>revision</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>55</td>
<td>summary</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>56</td>
<td>unspsc-code</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>57</td>
<td>unspsc-version</td>
<td>RFC-AAAA</td>
</tr>
<tr>
<td>58-4294967295</td>
<td>Unassigned</td>
<td></td>
</tr>
</tbody>
</table>

### 5.2. SWID/CoSWID Value Registries

The following IANA registries provide a mechanism for new valid values to be added over time to common enumerations used by SWID and CoSWID.
5.2.1. SWID/CoSWID Version Scheme Value Registry

This document uses unsigned 16-bit index values to represent version-scheme item values. The initial set of version-scheme values are derived from the textual version scheme names defined in the ISO/IEC 19770-2:2015 specification [SWID].

This document defines a new registry titled "SWID/CoSWID Version Scheme Values". Future registrations for this registry are to be made based on [RFC8126] as follows:

[TO BE REMOVED: This registration should take place at the following location: https://www.iana.org/assignments/swid]

<table>
<thead>
<tr>
<th>Range</th>
<th>Registration Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-16383</td>
<td>Standards Action</td>
</tr>
<tr>
<td>16384-32767</td>
<td>Specification Required</td>
</tr>
<tr>
<td>32768-65535</td>
<td>Reserved for Private Use</td>
</tr>
</tbody>
</table>

Initial registrations for the "SWID/CoSWID Version Scheme Value" registry are provided below. Assignments consist of an integer Index value, the Version Scheme Name, and a reference to the defining specification.
<table>
<thead>
<tr>
<th>Index</th>
<th>Version Scheme Name</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>multipartnumeric</td>
<td>See Section 4.1</td>
</tr>
<tr>
<td>2</td>
<td>multipartnumeric+suffix</td>
<td>See Section 4.1</td>
</tr>
<tr>
<td>3</td>
<td>alphanumeric</td>
<td>See Section 4.1</td>
</tr>
<tr>
<td>4</td>
<td>decimal</td>
<td>See Section 4.1</td>
</tr>
<tr>
<td>5-16383</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>16384</td>
<td>semver</td>
<td>[SEMVER]</td>
</tr>
<tr>
<td>16385-32767</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>32768-65535</td>
<td>Reserved for Private Use</td>
<td></td>
</tr>
</tbody>
</table>

5.2.2. SWID/CoSWID Entity Role Value Registry

This document uses unsigned 8-bit index values to represent entity-entry role item values. The initial set of Entity roles are derived from the textual role names defined in the ISO/IEC 19770-2:2015 specification [SWID].

This document defines a new a new registry titled "SWID/CoSWID Entity Role Values". Future registrations for this registry are to be made based on [RFC8126] as follows:

[TO BE REMOVED: This registration should take place at the following location: https://www.iana.org/assignments/swid]
Initial registrations for the "SWID/CoSWID Entity Role Value" registry are provided below. Assignments consist of an integer Index value, a Role Name, and a reference to the defining specification.

<table>
<thead>
<tr>
<th>Index</th>
<th>Role Name</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>tagCreator</td>
<td>See Section 4.2</td>
</tr>
<tr>
<td>2</td>
<td>softwareCreator</td>
<td>See Section 4.2</td>
</tr>
<tr>
<td>3</td>
<td>aggregator</td>
<td>See Section 4.2</td>
</tr>
<tr>
<td>4</td>
<td>distributor</td>
<td>See Section 4.2</td>
</tr>
<tr>
<td>5</td>
<td>licensor</td>
<td>See Section 4.2</td>
</tr>
<tr>
<td>6-127</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>128-255</td>
<td>Reserved for Private Use</td>
<td></td>
</tr>
</tbody>
</table>

5.2.3. SWID/CoSWID Link Ownership Value Registry

This document uses unsigned 8-bit index values to represent link-entry ownership item values. The initial set of Link ownership values are derived from the textual ownership names defined in the ISO/IEC 19770-2:2015 specification [SWID].

This document defines a new a new registry titled "SWID/CoSWID Link Ownership Values". Future registrations for this registry are to be made based on [RFC8126] as follows:

[TO BE REMOVED: This registration should take place at the following location: https://www.iana.org/assignments/swid]

<table>
<thead>
<tr>
<th>Range</th>
<th>Registration Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-31</td>
<td>Standards Action</td>
</tr>
<tr>
<td>32-127</td>
<td>Specification Required</td>
</tr>
<tr>
<td>128-255</td>
<td>Reserved for Private Use</td>
</tr>
</tbody>
</table>
Initial registrations for the "SWID/CoSWID Link Ownership Value" registry are provided below. Assignments consist of an integer Index value, an Ownership Type Name, and a reference to the defining specification.

<table>
<thead>
<tr>
<th>Index</th>
<th>Ownership Type Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>abandon</td>
<td>See Section 4.3</td>
</tr>
<tr>
<td>2</td>
<td>private</td>
<td>See Section 4.3</td>
</tr>
<tr>
<td>3</td>
<td>shared</td>
<td>See Section 4.3</td>
</tr>
<tr>
<td>4-16384</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>16385-32767</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>32768-65535</td>
<td>Reserved for Private Use</td>
<td></td>
</tr>
</tbody>
</table>

5.2.4. SWID/CoSWID Link Relationship Value Registry

This document uses unsigned 16-bit index values to represent link-entry rel item values. The initial set of rel values are derived from the textual rel names defined in the ISO/IEC 19770-2:2015 specification [SWID].

This document defines a new a new registry titled "SWID/CoSWID Link Relationship Values". Future registrations for this registry are to be made based on [RFC8126] as follows:

[TO BE REMOVED: This registration should take place at the following location: https://www.iana.org/assignments/swid]

<table>
<thead>
<tr>
<th>Range</th>
<th>Registration Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-16383</td>
<td>Standards Action</td>
</tr>
<tr>
<td>16384-32767</td>
<td>Specification Required</td>
</tr>
<tr>
<td>32768-65535</td>
<td>Reserved for Private Use</td>
</tr>
</tbody>
</table>
Initial registrations for the "SWID/CoSWID Link Relationship Value" registry are provided below. Assignments consist of an integer Index value, the Relationship Type Name, and a reference to the defining specification.

<table>
<thead>
<tr>
<th>Index</th>
<th>Relationship Type Name</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>ancestor</td>
<td>See Section 4.4</td>
</tr>
<tr>
<td>2</td>
<td>component</td>
<td>See Section 4.4</td>
</tr>
<tr>
<td>3</td>
<td>feature</td>
<td>See Section 4.4</td>
</tr>
<tr>
<td>4</td>
<td>installationmedia</td>
<td>See Section 4.4</td>
</tr>
<tr>
<td>5</td>
<td>packageinstaller</td>
<td>See Section 4.4</td>
</tr>
<tr>
<td>6</td>
<td>parent</td>
<td>See Section 4.4</td>
</tr>
<tr>
<td>7</td>
<td>patches</td>
<td>See Section 4.4</td>
</tr>
<tr>
<td>8</td>
<td>requires</td>
<td>See Section 4.4</td>
</tr>
<tr>
<td>9</td>
<td>see-also</td>
<td>See Section 4.4</td>
</tr>
<tr>
<td>10</td>
<td>supersedes</td>
<td>See Section 4.4</td>
</tr>
<tr>
<td>11</td>
<td>supplemental</td>
<td>See Section 4.4</td>
</tr>
<tr>
<td>12-16384</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>16385-32767</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>32768-65535</td>
<td>Reserved for Private Use</td>
<td></td>
</tr>
</tbody>
</table>

5.2.5. SWID/CoSWID Link Use Value Registry

This document uses unsigned 8-bit index values to represent link-entry use item values. The initial set of Link use values are derived from the textual names defined in the ISO/IEC 19770-2:2015 specification [SWID].
This document defines a new registry titled "SWID/CoSWID Link Use Values". Future registrations for this registry are to be made based on [RFC8126] as follows:

[TO BE REMOVED: This registration should take place at the following location: https://www.iana.org/assignments/swid]

<table>
<thead>
<tr>
<th>Range</th>
<th>Registration Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-31</td>
<td>Standards Action</td>
</tr>
<tr>
<td>32-127</td>
<td>Specification Required</td>
</tr>
<tr>
<td>128-255</td>
<td>Reserved for Private Use</td>
</tr>
</tbody>
</table>

Initial registrations for the "SWID/CoSWID Entity Role Value" registry are provided below. Assignments consist of an integer Index value, the Link Use Type Name, and a reference to the defining specification.

<table>
<thead>
<tr>
<th>Index</th>
<th>Link Use Type Name</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>optional</td>
<td>See Section 4.5</td>
</tr>
<tr>
<td>2</td>
<td>required</td>
<td>See Section 4.5</td>
</tr>
<tr>
<td>3</td>
<td>recommended</td>
<td>See Section 4.5</td>
</tr>
<tr>
<td>4-127</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>128-255</td>
<td>Reserved for Private Use</td>
<td></td>
</tr>
</tbody>
</table>

5.3. swid+cbor Media Type Registration

IANA is requested to add the following to the IANA "Media Types" registry.

Type name: application

Subtype name: swid+cbor

Required parameters: none
Optional parameters: none

Encoding considerations: Must be encoded as using [RFC7049]. 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 Software asset management systems, Vulnerability assessment systems, and in applications that use remote integrity verification.

Fragment identifier considerations: Fragment identification for application/swid+cbor is supported by using fragment identifiers as specified by RFC-AAAA. [Section to be defined]

Additional information:

Magic number(s): first five bytes in hex: da 53 57 49 44

File extension(s): coswid

Macintosh file type code(s): none

Macintosh Universal Type Identifier code: org.ietf.coswid 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

5.4. CoAP Content-Format Registration

IANA is requested to assign a CoAP Content-Format ID for the CoSWID 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]:

<table>
<thead>
<tr>
<th>Media type</th>
<th>Encoding</th>
<th>ID</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>application/swid+cbor</td>
<td>-</td>
<td>TBD1</td>
<td>RFC-AAAA</td>
</tr>
</tbody>
</table>

Table 1: CoAP Content-Format IDs

5.5. CBOR Tag Registration

IANA is requested to allocate a tag in the "CBOR Tags" registry", preferably with the specific value requested:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Data Item</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1398229316</td>
<td>map</td>
<td>Concise Software Identifier (CoSWID)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[RFC-AAAA]</td>
</tr>
</tbody>
</table>

6. Security Considerations

SWID and CoSWID tags contain public information about software components and, as such, do not need to be protected against disclosure on an endpoint. Similarly, SWID tags are intended to be easily discoverable by applications and users on an endpoint in order to make it easy to identify and collect all of an endpoint’s SWID tags. As such, any security considerations regarding SWID tags focus on the application of SWID tags to address security challenges, and the possible disclosure of the results of those applications.

A signed SWID tag whose signature has been validated can be relied upon to be unchanged since it was signed. If the SWID tag was created by the software provider, is signed, and the software provider can be authenticated as the originator of the signature, then the tag can be considered authoritative. In this way, an authoritative SWID tag contains information about a software product provided by the maintainer of the product, who is expected to be an expert in their own product. Thus, authoritative SWID tags can be trusted to represent authoritative information about the software product. Having an authoritative SWID tag can be useful when the information in the tag needs to be trusted, such as when the tag is being used to convey reference integrity measurements for software
components. By contrast, the data contained in unsigned tags cannot be trusted to be unmodified.

SWID tags are designed to be easily added and removed from an endpoint along with the installation or removal of software components. On endpoints where addition or removal of software components is tightly controlled, the addition or removal of SWID tags can be similarly controlled. On more open systems, where many users can manage the software inventory, SWID tags can be easier to add or remove. On such systems, it can be possible to add or remove SWID tags in a way that does not reflect the actual presence or absence of corresponding software components. Similarly, not all software products automatically install SWID tags, so products can be present on an endpoint without providing a corresponding SWID tag. As such, any collection of SWID tags cannot automatically be assumed to represent either a complete or fully accurate representation of the software inventory of the endpoint. However, especially on devices that more strictly control the ability to add or remove applications, SWID tags are an easy way to provide an preliminary understanding of that endpoint’s software inventory.

Any report of an endpoint’s SWID tag collection provides information about the software inventory of that endpoint. If such a report is exposed to an attacker, this can tell them which software products and versions thereof are present on the endpoint. By examining this list, the attacker might learn of the presence of applications that are vulnerable to certain types of attacks. As noted earlier, SWID tags are designed to be easily discoverable by an endpoint, but this does not present a significant risk since an attacker would already need to have access to the endpoint to view that information. However, when the endpoint transmits its software inventory to another party, or that inventory is stored on a server for later analysis, this can potentially expose this information to attackers who do not yet have access to the endpoint. For this reason, it is important to protect the confidentiality of SWID tag information that has been collected from an endpoint, not because those tags individually contain sensitive information, but because the collection of SWID tags and their association with an endpoint reveals information about that endpoint’s attack surface.

Finally, both the ISO-19770-2:2015 XML schema definition and the Concise SWID data definition allow for the construction of "infinite" SWID tags or SWID tags that contain malicious content with the intent if creating non-deterministic states during validation or processing of SWID tags. While software product vendors are unlikely to do this, SWID tags can be created by any party and the SWID tags collected from an endpoint could contain a mixture of vendor and non-vendor created tags. For this reason, tools that consume SWID tags
ought to treat the tag contents as potentially malicious and employ
input sanitizing on the tags they ingest.

7. Acknowledgments

TBD

8. Change Log

Changes from version 03 to version 09:

- Reduced representation complexity of the media-entry type and
  removed the section describing the older data structure.
- Added more signature schemes from COSE
- Included a minimal required set of normative language
- Reordering of attribute name to integer label by priority
  according to semantics.
- Added an IANA registry for CoSWID items supporting future
  extension.
- Cleaned up IANA registrations, fixing some inconsistencies in the
  table labels.
- Added additional CDDL sockets for resource collection entries
  providing for additional extension points to address future SWID/
  CoSWID extensions.
- Updated section on extension points to address new CDDL sockets
  and to reference the new IANA registry for items.
- Removed unused references and added new references to address
  placeholder comments.
- Added table with semantics for the link ownership item.
- Clarified language, made term use more consistent, fixed
  references, and replacing lowercase RFC2119 keywords.

Changes from version 02 to version 03:

- Updated core CDDL including the CDDL design pattern according to
  RFC 8428.

Changes from version 01 to version 02:
- Enforced a more strict separation between the core CoSWID definition and additional usage by moving content to corresponding appendices.

- Removed artifacts inherited from the reference schema provided by ISO (e.g. NMTOKEN(S))

- Simplified the core data definition by removing group and type choices where possible

- Minor reordering of map members

- Added a first extension point to address requested flexibility for extensions beyond the any-element

Changes from version 00 to version 01:

- Ambiguity between evidence and payload eliminated by introducing explicit members (while still allowing for "empty" SWID tags)

- Added a relatively restrictive COSE envelope using cose_sign1 to define signed CoSWID (single signer only, at the moment)

- Added a definition how to encode hashes that can be stored in the any-member using existing IANA tables to reference hash-algorithms

Changes since adopted as a WG I-D -00:

- Removed redundant any-attributes originating from the ISO-19770-2:2015 XML schema definition

- Fixed broken multi-map members

- Introduced a more restrictive item (any-element-map) to represent custom maps, increased restriction on types for the any-attribute, accordingly

- Fixed X.1520 reference

- Minor type changes of some attributes (e.g. NMTOKENS)

- Added semantic differentiation of various name types (e.g. fs-name)

Changes from version 06 to version 07:
* Added type choices/enumerations based on textual definitions in 19770-2:2015
* Added value registry request
* Added media type registration request
* Added content format registration request
* Added CBOR tag registration request
* Removed RIM appendix to be addressed in complementary draft
* Removed CWT appendix
* Flagged firmware resource collection appendix for revision
* Made use of terminology more consistent
* Better defined use of extension points in the CDDL
* Added definitions for indexed values
* Added IANA registry for Link use indexed values

Changes from version 05 to version 06:

* Improved quantities
* Included proposals for implicit enumerations that were NMTOKENS
* Added extension points
* Improved exemplary firmware-resource extension

Changes from version 04 to version 05:

* Clarified language around SWID and CoSWID to make more consistent use of these terms.
* Added language describing CBOR optimizations for single vs. arrays in the model front matter.
* Fixed a number of grammatical, spelling, and wording issues.
* Documented extension points that use CDDL sockets.
9. Contributors
10. References

10.1. Normative References


10.2. Informative References

[CamelCase]  "UpperCamelCase", August 2014,

[I-D.birkholz-rats-tuda]  Fuchs, A., Birkholz, H., McDonald, I., and C. Bormann,
"Time-Based Uni-Directional Attestation", draft-birkholz-
rats-tuda-00 (work in progress), March 2019.

[KebabCase]  "KebabCase", December 2014,

[RFC4122]  Leach, P., Mealling, M., and R. Salz, "A Universally
Unique IDentifier (UUID) URN Namespace", RFC 4122,
DOI 10.17487/RFC4122, July 2005,
Appendix A. Signed Concise SWID Tags using COSE

SWID tags, as defined in the ISO-19770-2:2015 XML schema, can include cryptographic signatures to protect the integrity of the SWID tag. In general, tags are signed by the tag creator (typically, although not exclusively, the vendor of the software component that the SWID tag identifies). Cryptographic signatures can make any modification of the tag detectable, which is especially important if the integrity of the tag is important, such as when the tag is providing reference integrity measurements for files.

The ISO-19770-2:2015 XML schema uses XML DSIG to support cryptographic signatures. CoSWID tags require a different signature scheme than this. COSE (CBOR Object Signing and Encryption) provides the required mechanism [RFC8152]. Concise SWID can be wrapped in a COSE Single Signer Data Object (COSE_Sign1) that contains a single signature. The following CDDL defines a more restrictive subset of header attributes allowed by COSE tailored to suit the requirements of Concise SWID tags.
Optionally, the COSE_Sign structure that allows for more than one signature to be applied to a CoSWID tag MAY be used. The corresponding usage scenarios are domain-specific and require well-defined application guidance. Representation of the corresponding guidance is out-of-scope of this document.

Additionally, the COSE Header counter signature MAY be used as an attribute in the unprotected header map of the COSE envelope of a CoSWID. The application of counter signing enables second parties to provide a signature on a signature allowing for a proof that a signature existed at a given time (i.e., a timestamp).

Authors’ Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt  64295
Germany

Email: henk.birkholz@sit.fraunhofer.de
Jessica Fitzgerald-McKay
Department of Defense
9800 Savage Road
Ft. Meade, Maryland
USA

Email: jmfitz2@nsa.gov

Charles Schmidt
The MITRE Corporation
202 Burlington Road
Bedford, Maryland  01730
USA

Email: cmschmidt@mitre.org

David Waltermire
National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, Maryland  20877
USA

Email: david.waltermire@nist.gov