Concise Software Identification Tags
draft-ietf-sacm-coswid-09

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 November 13, 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 ................................ 14  
   2.6. The link-entry Map .................................... 16  
   2.7. The software-meta-entry Map ........................... 19  
   2.8. The Resource Collection Definition ................... 22  
      2.8.1. The hash-entry Array .............................. 22  
      2.8.2. The resource-collection Group .................... 23  
      2.8.3. The payload-entry Group .......................... 26  
      2.8.4. The evidence-entry Group ........................ 26  
   2.9. Full CDDL Definition ................................... 27  
3. Determining the Type of CoSWID .............................. 33  
4. CoSWID Indexed Label Values ................................ 33  
   4.1. Version Scheme ......................................... 33  
   4.2. Entity Role Values .................................... 34  
   4.3. Use Values ............................................ 35  
5. IANA Considerations ........................................... 36  
   5.1. CoSWID Items Registry .................................. 36  
   5.2. SWID/CoSWID Version Schema Values Registry ....... 39  
   5.3. SWID/CoSWID Entity Role Values Registry ............ 40  
   5.4. SWID/CoSWID Link Use Values Registry ............... 41  
   5.5. swid+cbor Media Type Registration .................... 42  
   5.6. CoAP Content-Format Registration .................... 43  
   5.7. CBOR Tag Registration ................................ 44  
6. Security Considerations ...................................... 44  
7. Acknowledgments .............................................. 46  
8. Change Log .................................................. 46  
9. Contributors ................................................ 50  
10. References .................................................. 50  
   10.1. Normative References ................................ 50  
   10.2. Informative References ............................... 51
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.

```
+------------+
|            |
| Installation                  Product       Product       Product       Product |
| Media         -> Installed  -> Patched     -> Upgraded   -> Removed |
| Deployed                  |
| Corpus         Primary          Primary       xPrimary       xPrimary |
| Supplemental   Supplemental     xSupplemental xSupplemental |
| Patch          xPatch            Primary         |
|                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) [I-D.ietf-cbor-cddl]. 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 [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 by its name in CamelCase notation explicitly relates to XML SWID tags, an attribute referred 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 that would have to be stored or transported in full if using the original vocabulary are replaced.

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.

_{name_} = {(_label_: _data_ / [ 2* _data_ ])}

The CDDL rule above allows for a more efficient CBOR encoding of the data when a single value is used by avoiding the need to first encode the array. Conversely, an array is used for two or more values. This modeling pattern is used frequently in the CoSWID CDDL data definition in such cases.

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 [I-D.ietf-cbor-cddl] 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 following CDDL sockets defined in this document allow for adding new values to corresponding type-choice (i.e. to represent enumerations) via custom CDDL data definitions.
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 will provide for well-known index values for data items in CoSWID extensions, allowing these index values to be recognized by implementations supporting a given extension.

### 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:

<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>$ownership</td>
<td>Section 2.6</td>
</tr>
<tr>
<td>rel</td>
<td>$rel</td>
<td>Section 4.3</td>
</tr>
<tr>
<td>use</td>
<td>$use</td>
<td>Section 2.6</td>
</tr>
</tbody>
</table>
concise-swid-tag = {
  global-attributes,
  tag-id => text,
  tag-version => integer,
  ? corpus => bool,
  ? patch => bool,
  ? supplemental => bool,
  swid-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
swid-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 indicates if a specific release of a software component has more than one tag that can represent that specific release. 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 is changed when a CoSWID tag producer creates and releases an incorrect tag that they subsequently want to fix, but no underlying changes have been made to the product the CoSWID tag represents. This could happen if, for example, a patch is distributed that has a link reference that does not cover all the various software releases it can patch. A newer CoSWID tag for that patch can be generated and the tag-version value incremented to indicate that the data has been updated.

- 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".

- swid-name (index 1): This textual item provides the software component name as it would typically be referenced. For example, what would be seen in the add/remove software dialog in an operating system, or what is specified as the name of a packaged software component or a patch identifier name.

- 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 Schema Values Registry (see section Section 5.2 or a value in the private use range: 32768-65535.

- 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. The data definition of this entry allows for any additional attribute to be included, though it is recommended that industry norms for new attributes are defined and followed to the degree possible. Described in Section 2.7.

- entity (index 2): Provides information about one or more organizations related to the CoSWID tag or 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. Note that the payload can represent a superset of the software artifacts 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 (i.e. if a particular optional sub-component is not installed, the files associated with that software component might be included in payload, but not installed on the endpoint). Described in Section 2.8.3.

o evidence (index 3): This item is used to provide results from a scan of a system where software that does not have a CoSWID tag is discovered. In such a case, a CoSWID tag can be created by the discovery process when the endpoint is scanned. This item represents evidence for why software is believed to be installed on the endpoint. Described in Section 2.8.4.

o $$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.

o Only one of the patch and supplemental items MUST be set to "true", or the patch and supplemental items MUST be set to "false" or be omitted.

o 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.

o 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.

o 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 text-string name of the organization claiming a particular role, specified by the role item, in 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 this organization and this tag or the referenced software component. The role of an entity MAY include any role value, but the pre-defined roles include: "aggregator", "distributor", "licensor", "software-creator", and "tag-creator". All pre-defined role index and text values are defined in the IANA "SWID/CoSWID Entity Role Values"
Use of index values instead of text for these pre-defined roles allows a CoSWID to be more concise.

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.

- **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.

- **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:

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

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

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

$rel /= ancestor
$rel /= component
$rel /= feature
$rel /= installationmedia
$rel /= packageinstaller
$rel /= parent
$rel /= patches
$rel /= requires
$rel /= see-also
$rel /= supersedes
$rel /= rel-supplemental
$rel /= uint / text
ancestor=1
component=2
feature=3
installationmedia=4
packageinstaller=5
parent=6
patches=7
requires=8
see-also=9
supersedes=10
rel-supplemental=11

$use /= optional
$use /= required
$use /= recommended
$use /= uint / text
optional=1
required=2
recommended=3

The following describes each member of this map.

o global-attributes: The global-attributes group described in Section 2.4.
artifact (index 37): For installation media (rel="installation-media"), this item value indicates the path of the installer executable or script that can be run to launch the referenced installation. Links with the same artifact name SHOULD 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 item being referenced. The "href" item’s value can point to several different things, and can be any of the following:

* 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-f7ddd1ade4c" references the tag with the tag-id value "2df9de35-0aff-4a86-ace6-f7ddd1ade4c".

* 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/@persistId='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): Determines the relative strength of ownership between the software component referenced by the COSWID
tag and the software component referenced by the link. Valid enumerations are: abandon, private, shared.

The enumerated values have the following meanings:

<table>
<thead>
<tr>
<th>ownership</th>
<th>semantics</th>
</tr>
</thead>
<tbody>
<tr>
<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>private</td>
<td>If the software component referenced by the CoSWID tag is uninstalled, then the referenced software SHOULD be uninstalled too.</td>
</tr>
<tr>
<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>

- rel (index 40): The relationship between this CoSWID and the target resource as defined by [RFC8288]. Relationships can be identified by referencing a "Relation Name" from the IANA "Link Relation Types" registry: https://www.iana.org/assignments/link-relations/link-relations.xhtml.

- media-type (index 41): The media type for the target resource. This provides the 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 target software is a hard requirement or not to be installed before installing the tagged software component. Valid enumerations are: required, recommended, optional, which are defined in Section 4.3.

- $$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
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

The following describes each child item of this group.

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

○ activation-status (index 43): A textual value that identifies the activation status of this software title (e.g. Trial, Serialized, Licensed, Unlicensed, etc). Typically, this is used in supplemental tags.

○ channel-type (index 44): A textual value that provides information on which channel this particular software was targeted for (e.g.
 colloquial-version (index 45): A textual value for an informal or colloquial version of the product (i.e. 2013). Note that 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 release. This representation of version is typically used to identify a group of specific software releases that are part of the same release/support infrastructure (i.e. Fabrikam Office 2013). This version is used for string comparisons only and is not compared to be an earlier or later release (that is done via the concise-swid-tag group’s software-version item).

 description (index 46): A longer, detailed textual description of the software. This description can be multiple sentences (differentiated from summary, which is a very short, one-sentence description).

 edition (index 47): A textual value indicating the variation of the product (e.g., Extended, Enterprise, Professional, Standard, etc).

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

 entitlement-key (index 49): A vendor-specific textual key that can be used to reconcile the validity of an entitlement. (e.g., serial number, product or license key).

 generator (index 50): The name of the software tool that created the CoSWID tag.

 persistent-id (index 51): A GUID used to represent products installed where the products are related, but can be different versions.

 product (index 52): The base name of the product.

 product-family (index 53): A textual value indicating the overall product family this software belongs to. Product family is not used to identify that a product is part of a suite, but is instead used when a set of products that are all related can be installed on multiple different endpoints. For example, an enterprise backup system can consist of a backup services, multiple different backup services that support mail services, databases and ERP
systems, as well as individual software components that backup client system entities. In such an usage scenario, all software components that are part of the backup system would have the same product-family name so they can be grouped together in respect to reporting systems.

- revision (index 54): A textual value indicating the informal or colloquial representation of the sub-version of the given product (ie, SP1, R2, RC1, Beta 2, etc). Note that the software-version specified in the concise-swid-tag group will provide very exact version details. Conversely, the revision item is intended for use in environments where reporting on the informal or colloquial representation of the software is important. For example, when an organization needs "ServicePack 1" or later of a specific product installed on all devices, they can use the revision data value to quickly identify any devices that do not meet this requirement.

Depending on how a software organizations distributes revisions, this value could be specified in a primary (if distributed as an upgrade) or supplemental (if distributed as a patch) CoSWID tag.

- summary (index 55): A short (one-sentence) description of the software.

- unspsc-code (index 56): An 8 digit code that provides UNSPSC classification of the software component this SWID tag identifies. For more information see, http://www.unspsc.org/.

- unspsc-version (index 57): The version of the UNSPSC code used to define the UNSPSC code value. For more information see, http://www.unspsc.org/.

- $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-
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:

```cddl
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 = {
```

Internet-Draft                  COSWID                         May 2019

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.
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.

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

- directory (index 16): A directory item allows one or more directories to be defined in the file structure.

- file (index 17): A file item that allows one or more files to be specified for a given location.

- process (index 18): Provides process (software component in execution) information for data that will show up in a devices process list.
- resource (index 19): A set of items that can be used to provide arbitrary resource information about an application installed on an endpoint, or evidence collected from an endpoint.

- size (index 20): The size of the file in bytes.

- file-version (index 21): The version of the file.

- key (index 22): A boolean indicator for when files or directories are considered important or required for the use of the software component referenced by the CoSWID. Typical key files or directories would be those which, if not available on an endpoint, would cause the software component not to execute or function properly. Key files or directories will typically be used to validate that the software component referenced by the CoSWID tag is actually installed on a specific endpoint.

- location (index 23): The location where a file was found or can be expected to be located. This text-string is intended to include the filename itself. This SHOULD be the relative path from the location represented by the root item or if the root item is omitted be relative to the location of the CoSWID tag.

- fs-name (index 24): The file name or directory name without any path characters.

- root (index 25): A system-specific root folder that the location item is an offset from. If this is not specified the assumption is the root is the same folder as the location of the CoSWID tag. The text-string value represents a path expression relative to the CoSWID tag document location in the (composite) file-system hierarchy.

- path-elements (index 26): This group provides the ability to apply a directory structure to the path expressions for files defined in a payload or evidence items.

- process-name (index 27): The process name as it will be found in the endpoint’s process table.

- pid (index 28): The process ID for the process in execution that can be included in the process item as part of an evidence tag.

- type (index 29): The type of resource represented via a text-string (typically, registry-key, port or root-uri).
2.8.3. The payload-entry Group

The CDDL for the payload-entry group follows:

```cddl
payload-entry = {
  global-attributes,
  resource-collection,
  * $$payload-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.
- $$payload-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 evidence represented by an evidence item was gathered.
- device-id (index 36): A textual identifier for a device evidence was gathered from.
- $$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.

cons-sewid-tag = {
  global-attributes,
  tag-id => text,
  tag-version => integer,
  ? corpus => bool,
  ? patch => bool,
  ? supplemental => bool,
  swid-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 /= 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,
  ? unspsc-code => text,
  ? unspsc-version => text,
  * $unspsc-version => 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
}

directory-item = {
  global-attributes,
  ? key => bool,
  ? location => text,
  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
swid-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
device-id = 37
href = 38
ownership = 39
rel = 40
media-type = 41
use = 42
activation-status = 43
channel-type = 44
colloquial-version = 45
description = 46
description = 47
title = 48
title = 49
generator = 50
date = 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

; "rel" integer indexes
ancestor=1
component=2
feature=3
installationmedia=4
3. Determining the Type of CoSWID

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 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 value</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>Index</td>
<td>Version Scheme Name</td>
<td>Definition</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>multipartnumeric</td>
<td>Numbers separated by dots, where the numbers are interpreted as integers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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 Schema Values" registry defined in Section 5.2. 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. 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 Values" registry defined in section Section 5.3. Additional valid values will likely be registered over time. Additionally, the index values 128 through 255 have been reserved for private use.

4.3. 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. 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 Values" registry defined in section Section 5.4. 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:

<table>
<thead>
<tr>
<th>Range</th>
<th>Registration Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-32767</td>
<td>Standards Action</td>
</tr>
<tr>
<td>32768-4294967295</td>
<td>Specification Required</td>
</tr>
</tbody>
</table>

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>swid-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>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></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>----------</td>
<td>----------</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>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>
</tbody>
</table>
5.2. SWID/CoSWID Version Schema Values Registry

This document uses unsigned 16-bit index values to 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 a new registry titled "SWID/CoSWID Version Schema 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 Version Schema Values" 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>

5.3. SWID/CoSWID Entity Role Values Registry

This document uses unsigned 8-bit index values to represent entity-role 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]
### Range | Registration Procedures
---|---
0-31 | Standards Action
32-127 | Specification Required
128-255 | Reserved for Private Use

Initial registrations for the "SWID/CoSWID Entity Role Values" 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>6-31</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>32-127</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>128-255</td>
<td>Reserved for Private Use</td>
<td></td>
</tr>
</tbody>
</table>

### 5.4. SWID/CoSWID Link Use Values Registry

This document uses unsigned 8-bit index values to represent link-use 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]

+---------+--------------------------+
| Range   | Registration Procedures  |
+---------+--------------------------+
| 0-31    | Standards Action         |
| 32-127  | Specification Required   |
| 128-255 | Reserved for Private Use |
+---------+--------------------------+

Initial registrations for the "SWID/CoSWID Entity Role Values" registry are provided below. Assignments consist of an integer index value, the link use name, and a reference to the defining specification.

+---------+--------------------------+-----------------+
<table>
<thead>
<tr>
<th>Index</th>
<th>Link Use 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 4.3</td>
</tr>
<tr>
<td>2</td>
<td>required</td>
<td>See Section 4.3</td>
</tr>
<tr>
<td>3</td>
<td>recommended</td>
<td>See Section 4.3</td>
</tr>
<tr>
<td>4-31</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>32-127</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>128-255</td>
<td>Reserved for Private Use</td>
<td></td>
</tr>
</tbody>
</table>
+---------+--------------------------+-----------------+

5.5. swid+cbor Media Type Registration

IANA is requested 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

5.6. 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

5.7. 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</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Item</td>
<td></td>
</tr>
<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. As such, 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.
- Added enumeration table
- Moved CBOR specific indexes to the end of the CDDL data definition
- Updated Examples

Changes from version 02 to version 03:
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:

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

Changes from version 06 to version 07:

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

Changes from version 05 to version 06:

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

Changes from version 04 to version 05:

o Clarified language around SWID and CoSWID to make more consistent use of these terms.

o 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.

Converted IANA registration tables to markdown tables, reserving the 0 value for use when a value is not known.

Updated a number of references to their current versions.

Changes from version 03 to version 04:

- Re-index label values in the CDDL.
- Added a section describing the CoSWID model in detail.
- Created IANA registries for entity-role and version-scheme

Changes from version 02 to version 03:

- Updated CDDL to allow for a choice between a payload or evidence
- Re-index label values in the CDDL.
- Added item definitions
- Updated references for COSE, CBOR Web Token, and CDDL.

Changes from version 01 to version 02:

- Added extensions for Firmware and CoSWID use as Reference Integrity Measurements (CoSWID RIM)
- Changes meta handling in CDDL from use of an explicit use of items to a more flexible unconstrained collection of items.
- Added sections discussing use of COSE Signatures and CBOR Web Tokens

Changes from version 00 to version 01:

- Added CWT usage for absolute SWID paths on a device
- Fixed cardinality of type-choices including arrays
- Included first iteration of firmware resource-collection
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.

[I-D.ietf-cbor-cddl]

[KebabCase]
"KebabCase", December 2014,
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