SACM Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2021

H. Birkholz
Fraunhofer SIT
J. Fitzgerald-McKay
Department of Defense
C. Schmidt
The MITRE Corporation
D. Waltermire
NIST
November 02, 2020

Concise Software Identification Tags
draft-ietf-sacm-coswid-16

Abstract

ISO/IEC 19770-2:2015 Software Identification (SWID) tags provide an extensible XML-based structure to identify and describe individual software components, patches, and installation bundles. SWID tag representations can be too large for devices with network and storage constraints. This document defines a concise representation of SWID tags: Concise SWID (CoSWID) tags. CoSWID supports a similar set of semantics and features as SWID tags, as well as new semantics that allow CoSWIDs to describe additional types of information, all in a more memory efficient format.

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 May 6, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
1.1. The SWID and CoSWID Tag Lifecycle 4
1.2. Concise SWID Format 8
1.3. Requirements Notation 8
2. Concise SWID Data Definition 8
2.1. Character Encoding 9
2.2. Concise SWID Extensions 10
2.3. The concise-swid-tag Map 12
2.4. concise-swid-tag Co-constraints 16
2.5. The global-attributes Group 16
2.6. The entity-entry Map 17
2.7. The link-entry Map 19
2.8. The software-meta-entry Map 23
2.9. The Resource Collection Definition 27
2.9.1. The hash-entry Array 27
2.9.2. The resource-collection Group 27
2.9.3. The payload-entry Map 30
2.9.4. The evidence-entry Map 31
2.10. Full CDDL Specification 31
3. Determining the Type of CoSWID 37
4. CoSWID Indexed Label Values 38
4.1. Version Scheme 38
4.2. Entity Role Values 40
4.3. Link Ownership Values 42
4.4. Link Rel Values 42
4.5. Link Use Values 44
5. IANA Considerations 45
5.1. CoSWID Items Registry 45
5.2. Software Tag Values Registries 48
5.2.1. Registration Procedures 48
5.2.2. Private Use of Index and Name Values 49
5.2.3. Expert Review Guidelines 49
5.2.4. Software Tag Version Scheme Values Registry 50
5.2.5. Software Tag Entity Role Values Registry 51
5.2.6. Software Tag Link Ownership Values Registry 52
5.2.7. Software Tag Link Relationship Values Registry . . . 53

5.2.8. Software Tag Link Use Values Registry 55
5.3. swid+cbor Media Type Registration 56
5.4. CoAP Content-Format Registration 58
5.5. CBOR Tag Registration 58
5.6. URI Scheme Registrations 58
5.6.1. "swid" URI Scheme Registration 59
5.6.2. "swid" URI Scheme Specification [TODO: FIXME: has to move out of registration] 59
5.6.3. "swidpath" URI Scheme Registration 60
5.6.4. "swidpath" URI Scheme Specification [TODO: FIXME: has to move out of registration] 60
5.7. CoSWID Model for use in SWIMA Registration 61
6. Security Considerations 61
7. Acknowledgments 63
8. Change Log 63
9. References 68
9.1. Normative References 68
9.2. Informative References 71
Appendix A. Signed Concise SWID Tags using COSE 72
Authors' Addresses 73

1. Introduction

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 software, a patch, or an installation bundle, which are referred to as software components in this document. 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.

SWID tags are used to support a number of processes 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 Attester-produced event logs that complement attestation Evidence [I-D.ietf-rats-architecture].

While there are very few required fields in SWID tags, there are many optional fields that support different uses. 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 usage scenarios, 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 through the use of the Concise Binary Object Representation (CBOR) [RFC7049]. [TODO: Add CoSWID size comparison.]

In a CoSWID, the human-readable labels of SWID data items are replaced with more concise integer labels (indices). This approach allows SWID and CoSWID to share a common implicit information model, with CoSWID providing an alternate data model [RFC3444]. While SWID and CoSWID are intended to share the same implicit information model, this specification does not define this information model, or a mapping between the the two data formats. While an attempt to align SWID and CoSWID tags has been made here, future revisions of ISO/IEC 19770-2:2015 or this specification might cause this implicit information model to diverge, since these specifications are maintained by different standards groups.

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. The following text is paraphrased from these sources.

1. Primary Tag - A SWID or CoSWID tag that identifies and describes an installed software component on an endpoint. 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 that 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 allows tools and users to record their own metadata about a software component without modifying SWID primary or patch tags created by a software provider.

The type of a tag is determined by specific data elements, which are discussed in Section 3, which also provides normative language for CoSWID semantics that implement this lifecycle. The following information helps to explain how these semantics apply to use of a CoSWID tag.

> Corpus, primary, and patch tags have similar functions in that they describe the existence and/or presence of different types of software components (e.g., software installers, software installations, software patches), and, potentially, different states of these software components. Supplemental tags have the same structure as other tags, but are used to provide information not contained in the referenced corpus, primary, and 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            |
  Software      Software        Software     Software      Software
  Deployment -> Installation -> Patching  -> Upgrading  -> Removal

  Corpus        Primary         Primary      xPrimary      xPrimary
  Supplemental  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. A detailed description of the four tags types is provided in Section 2.3. The figure identifies the types of tags that are used in each lifecycle event.

There are many ways in which software tags might be managed for the host the software is installed on. For example, software tags could be made available on the host or to an external software manager when storage is limited on the host.

In these cases the host or external software manager is responsible for management of the tags, including deployment and removal of the tags as indicated by the above lifecycle. Tags are deployed and previously deployed tags that are typically removed (indicated by an "x" prefix) at each lifecycle stage, 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).

Corpus tags are not actually deployed on the target system but are intended to support deployment procedures and their dependencies at install-time, such as to verify the installation media.

* 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. A new patch tag is provided, when a patch is applied to the software component, supplying details about the patch and its dependencies. While not illustrated in the figure, a corpus tag can also provide information about the patch installer and patching dependencies that need to be installed before the patch.

* 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.

Note: In the context of software tagging software patching and updating differ in an important way. When installing a patch, a set of file modifications are made to pre-installed software which do not alter the version number or the descriptive metadata of an installed software component. An update can also make a set of file modifications, but the version number or the descriptive metadata of an installed software component are changed.

* 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.

As 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, which is based on CBOR. CBOR-based CoSWID tags offer a more concise representation of SWID information as compared to the XML-based SWID tag representation in ISO-19770-2:2015. 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 software component information and allows CoSWID tags to support the same uses as a SWID tag.

The vocabulary, i.e., the CDDL names of the types and members used in the CoSWID CDDL specification, are mapped to more concise labels represented as small integer values (indices). The names used in the CDDL specification and the mapping to the CBOR representation using integer indices 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 describes the general rules and processes for encoding data using CDDL representation. Prior familiarity with CBOR and CDDL concepts will be helpful in understanding this CoSWID specification.

This section describes the conventions by which a CoSWID is represented in the CDDL structure. 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 CDDL specification. 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 CBOR CoSWID tags. This approach simplifies the composition of further work that reference both XML SWID and CBOR CoSWID documents.

In most cases, mapping attribute names between SWID and CoSWID can be done automatically by converting between CamelCase and KebabCase attribute names. However, some CoSWID CDDL attribute names show greater variation relative to their corresponding SWID XML Schema attributes. This is done when the change improves clarity in the CoSWID specification. For example the "name" and "version" SWID fields corresponds to the "software-name" and "software-version" CoSWID fields, respectively. As such, it is not always possible to mechanically translate between corresponding attribute names in the two formats. In such cases, a manual mapping will need to be used.

The 57 human-readable text labels of the CDDL-based CoSWID vocabulary are mapped to integer indices via a block of rules at the bottom of the definition. This allows a more concise integer-based form to be stored or transported, as compared to the less efficient text-based form of 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.

```
_name_ = (_label_ => _data_ / [ 2* _data_ ])
```

The CDDL rule above allows either a single data item or an array of 2 or more data values to be provided. When a singleton data value is provided, the CBOR markers for the array, array length, and stop point are not needed, saving bytes. When two or more data values are provided, these values are encoded as an array. This modeling pattern is used frequently in the CoSWID CDDL specification to allow for more efficient encoding of singleton values.

[TODO: Are there any considerations that would need to be made for versioning CoSWID beyond the native support provided with CBOR?]

The following subsections describe the different parts of the CoSWID model.

2.1. Character Encoding

The CDDL "text" type is represented in CBOR as a major type 3, which represents "a string of Unicode characters that [are] encoded as UTF-8 [RFC3629]" (see [RFC7049] Section 2.1). Thus both SWID and CoSWID use UTF-8 for the encoding of characters in text strings.

To ensure that UTF-8 character strings are able to be encoded/decoded and exchanged interoperably, text strings in CoSWID MUST be encoded consistent with the Net-Unicode definition defined in [RFC5198].

All names registered with IANA according to requirements in Section Section 5.2 also MUST be valid according to the XML Schema NMToken data type (see [W3C.REC-xmlschema-2-20041028] Section 3.3.4) to ensure compatibility with the SWID specification where these names are used.

2.2. Concise SWID Extensions

The CoSWID specification contains two features that are not included in the SWID specification on which it is based. These features are:

- The explicit definition of types for some attributes in the ISO-19770-2:2015 XML representation that are typically represented by the "any attribute" in the SWID model. These are covered in Section 2.5.

- The inclusion of extension points in the CoSWID specification 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.

Map Name	CDDL Socket	Defined in
concise-swid-tag	$$coswid-extension	Section 2.3
entity-entry	$$entity-extension	Section 2.6
link-entry	$$link-extension	Section 2.7
software-meta-entry	$$software-meta-extension	Section 2.8
file-entry	$$file-extension	Section 2.9.2
directory-entry	$$directory-extension	Section 2.9.2
process-entry	$$process-extension	Section 2.9.2
resource-entry	$$resource-extension	Section 2.9.2
payload-entry	$$payload-extension	Section 2.9.3
evidence-entry	$$evidence-extension	Section 2.9.4

Table 1: CoSWID CDDL Group Extension Points

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 in the table 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 additional CDDL sockets are defined in this document to allow for adding new values to corresponding type-choices (i.e. to represent enumerations) via custom CDDL specifications.

Enumeration Name	CDDL Socket	Defined in
version-scheme	$version-scheme	Section 4.1
role	$role	Section 4.2
ownership	$ownership	Section 4.3
rel	$rel	Section 4.4
use	$use	Section 4.5

Table 2: CoSWID CDDL Enumeration Extension Points

A number of CoSWID value registries are also defined in Section 5.2 that allow new 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 CoSWID, which can also be used by other software tagging specifications. This registration mechanism allows new standardized enumerated values to be shared between multiple tagging specifications (and associated implementations) over time.

2.3. The concise-swid-tag Map

The CDDL specification 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 = {
  tag-id => text / bstr .size 16,
  tag-version => integer,
  ? corpus => bool,
  ? patch => bool,
  ? supplemental => bool,
  software-name => text,
  ? software-version => text,
  ? version-scheme => $version-scheme,
  ? media => text,
  ? software-meta => software-meta-entry / [ 2* software-meta-entry ],
  entity => entity-entry / [ 2* entity-entry ],
  ? link => link-entry / [ 2* link-entry ],
  ? payload-or-evidence,
  global-attributes,
  * $$coswid-extension,
}
```

```
payload-or-evidence //= ( payload => payload-entry ] )
payload-or-evidence //= ( payload => [ 2* payload-entry )
payload-or-evidence //= ( evidence => evidence-entry )
payload-or-evidence //= ( evidence => [ 2* evidence-entry ] )

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.

- o 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 2.5.

- o tag-id (index 0): A 16 byte binary string or textual identifier uniquely referencing a software component. The tag identifier MUST be globally unique. If represented as a 16 byte binary string, the identifier MUST be a valid universally unique identifier as defined by [RFC4122]. There are no strict guidelines on how 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 could 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 that has made incremental changes to a software component installed on an endpoint. 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". A patch item's value MUST NOT be set to "true" if the installation of the associated software package changes the version of a software component.

- 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. This allows tools and users to record their own metadata about a software component without modifying SWID primary or patch tags created by a software provider. 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 integer or textual value representing the versioning scheme used for the software-version item. If an integer value is used it MUST be an index value in

the range -256 to 65535. Integer values in the range -256 to -1 are reserved for testing and use in closed environments (see Section Section 5.2.2). Integer values in the range 0 to 65535 correspond to registered entries in the IANA "Software Tag Version Scheme Values" registry (see Section Section 5.2.4. If a string value is used it MUST be a private use name as defined in Section Section 5.2.2. String values based on a Version Scheme Name from the IANA "Software Tag Version Scheme Values" registry MUST NOT be used, as these values are less concise than their index value equivalent.

- o media (index 10): This text value is a hint to the tag consumer to understand what target platform this tag applies to. This item item MUST be formatted as a query as defined by the W3C Media Queries Recommendation (see [W3C.REC-css3-mediaqueries-20120619]). Support for media queries are included here for interoperability with [SWID], which does not provide any further requirements for media query use. Thus, this specification does not clarify how a media query is to be used for a CoSWID.

- o 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 2.8.

- o 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 2.6.

- o 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 2.7.

- o payload (index 6): This item represents a collection of software artifacts (described by child items) that compose the target software. For example, these artifacts could be 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 2.9.3.

- o evidence-entry (index 3): This item can be used to record the results of a software discovery process used to identify untagged software on an endpoint or to represent indicators for why software is believed to be installed on the endpoint. In either case, a CoSWID tag can be created by the tool performing an analysis of the software components installed on the endpoint. Described in Section 2.9.4.

- o $$coswid-extension: This CDDL socket is used to add new information structures to the concise-swid-tag root map. See Section 2.2.

2.4. concise-swid-tag Co-constraints

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

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

- o If the patch item is set to "true", the tag SHOULD contain at least one link item (see Section Section 2.7) with both the rel 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 both the rel item value of "supplemental" 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.

2.5. 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/index ("key") value pairs. Labels can be either a single integer or text string. Values can be a single integer, a text string, or an array of integers or text strings.

2.6. The entity-entry Map

The CDDL for the entity-entry map follows:

```
entity-entry = {
  entity-name => text,
  ? reg-id => any-uri,
  role => $role / [ 2* $role ],
  ? thumbprint => hash-entry,
  global-attributes,
  * $$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 /= maintainer
$role /= uint / text
tag-creator=1
software-creator=2
aggregator=3
distributor=4
licensor=5
maintainer=6
```

The following describes each child item of this group.

- o global-attributes: The global-attributes group described in Section 2.5.

- o entity-name (index 31): The textual name of the organizational entity claiming the roles specified by the role item for the CoSWID tag.

- o 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 a registration ID MUST be a RFC 3986 URI. The scope SHOULD be the scope of an organization.

- o role (index 33): An integer or textual value representing the relationship(s) between the entity, and this tag or the referenced software component. If an integer value is used it MUST be an index value in the range -256 to 255. Integer values in the range

-256 to -1 are reserved for testing and use in closed environments (see Section Section 5.2.2). Integer values in the range 0 to 255 correspond to registered entries in the IANA "Software Tag Entity Role Values" registry (see Section Section 5.2.5. If a string value is used it MUST be a private use name as defined in Section Section 5.2.2. String values based on a Role Name from the IANA "Software Tag Entity Role Values" registry MUST NOT be used, as these values are less concise than their index value equivalent.

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 2.9.1 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 2.2.

2.7. The link-entry Map

The CDDL for the link-entry map follows:

```
link-entry = {
  ? artifact => text,
  href => any-uri,
  ? media => text,
  ? ownership => $ownership,
  rel => $rel,
  ? media-type => text,
```

```
    ? use => $use,
    global-attributes,
    * $$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 /= supplemental
  $rel /= -356..65536 / text
  ancestor=1
  component=2
  feature=3
  installationmedia=4
  packageinstaller=5
  parent=6
  patches=7
  requires=8
  see-also=9
  supersedes=10
  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 2.5.

- o artifact (index: 37): To be used with rel="installation-media", this item's value provides 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.

- o href (index 38): A URI-reference [RFC3986] 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-reference is a a relative reference 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 "swid:" as the scheme refers to another SWID or CoSWID by the referenced tag's tag-id. This URI needs to be resolved in the context of the endpoint by software that can lookup other SWID or CoSWID tags. For example, "swid: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 software tag via an XPATH query [W3C.REC-xpath20-20101214]. This scheme is provided for compatibility with [SWID]. This specification does not define how to resolve an XPATH query in the context of CBOR.

- o media (index 10): A hint to the consumer of the link to what target platform the link is applicable to. 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 2.3.

- ownership (index 39): An integer or textual value 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. If an integer value is used it MUST be an index value in the range -256 to 255. Integer values in the range -256 to -1 are reserved for testing and use in closed environments (see Section Section 5.2.2). Integer values in the range 0 to 255 correspond to registered entries in the IANA "Software Tag Link Ownership Values" registry (see Section Section 5.2.6. If a string value is used it MUST be a private use name as defined in Section Section 5.2.2. String values based on a Ownership Type Name from the IANA "Software Tag Link Ownership Values" registry MUST NOT be used, as these values are less concise than their index value equivalent.

- rel (index 40): An integer or textual value that identifies the relationship between this CoSWID and the target resource identified by the "href" item. If an integer value is used it MUST be an index value in the range -256 to 65535. Integer values in the range -256 to -1 are reserved for testing and use in closed environments (see Section Section 5.2.2). Integer values in the range 0 to 65535 correspond to registered entries in the IANA "Software Tag Link Relationship Values" registry (see Section Section 5.2.7. If a string value is used it MUST be either a private use name as defined in Section Section 5.2.2 or a "Relation Name" from the IANA "Link Relation Types" registry: https://www.iana.org/assignments/link-relations/link-relations.xhtml as defined by [RFC8288]. When a string value defined in the IANA "Software Tag Link Relationship Values" registry matches a Relation Name defined in the IANA "Link Relation Types" registry, the index value in the IANA "Software Tag Link Relationship Values" registry MUST be used instead, as this relationship has a specialized meaning in the context of a CoSWID tag. String values based on a Relationship Type Name from the IANA "Software Tag Link Relationship Values" registry MUST NOT be used, as these values are less concise than their index value equivalent.

- media-type (index 41): A link can point to arbitrary resources on the endpoint, local network, or Internet using the href item. Use of this item supplies the 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): An integer or textual value used to determine if the referenced software component has to be installed before

installing the software component identified by the COSWID tag. If an integer value is used it MUST be an index value in the range -256 to 255. Integer values in the range -256 to -1 are reserved for testing and use in closed environments (see Section Section 5.2.2). Integer values in the range 0 to 255 correspond to registered entries in the IANA "Link Use Values" registry (see Section Section 5.2.8. If a string value is used it MUST be a private use name as defined in Section Section 5.2.2. String values based on an Link Use Type Name from the IANA "Software Tag Link Use Values" registry MUST NOT be used, as these values are less concise than their index value equivalent.

- o $$link-extension: This CDDL socket can be used to extend the link-entry map model. See Section 2.2.

2.8. The software-meta-entry Map

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

```
software-meta-entry = {
  ? activation-status => text,
  ? channel-type => text,
  ? colloquial-version => text,
  ? description => text,
  ? edition => text,
  ? entitlement-data-required => bool,
  ? entitlement-key => text,
  ? generator => text,
  ? persistent-id => text,
  ? product => text,
  ? product-family => text,
  ? revision => text,
  ? summary => text,
  ? unspsc-code => text,
  ? unspsc-version => text,
  global-attributes,
  * $$software-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.

- o global-attributes: The global-attributes group described in Section 2.5.

- 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. This value MAY be multiple paragraphs separated by CR LF characters as described by [RFC5198].

- 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 can be used 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.

- o 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.

- o persistent-id (index 51): A globally unique identifier used to identify a set of software components that are related. Software components sharing the same persistent-id can be different versions. This item can be used to relate software components, released at different points in time or through different release channels, that may not be able to be related through use of the link item.

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

- o 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. Use of this item is not intended to represent groups of software that are bundled or installed together. The persistent-id or link items SHOULD be used to relate bundled software components.

- o 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.

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

- o unspsc-code (index 56): An 8 digit UNSPSC classification code for the software component as defined by the United Nations Standard Products and Services Code (UNSPSC, [UNSPSC]).

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

- o $$meta-extension: This CDDL socket can be used to extend the software-meta-entry group model. See Section 2.2.

2.9. The Resource Collection Definition

2.9.1. 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 member (index 7) and the corresponding hash-entry type. This is the equivalent of the namespace qualified "hash" attribute in [SWID].

```
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" with a Status of "current" (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.

2.9.2. 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:

```
path-elements-group = ( ? directory => directory-entry / [ 2* directory-
entry ],
                        ? file => file-entry / [ 2* file-entry ],
                      )

resource-collection = (
  path-elements-group,
  ? process => process-entry / [ 2* process-entry ],
  ? resource => resource-entry / [ 2* resource-entry ],
  * $$resource-collection-extension,
)

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

```
  ? root => text,
  global-attributes,
)

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

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

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

resource-entry = {
  type => text,
  global-attributes,
  * $$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 common items used for representing the filesystem root, relative location, name, and significance of a file or directory item.

- global-attributes: The global-attributes group described in Section 2.5.

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

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

- 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 an endpoint.

- resource (index 19): A resource item can be used to provide details about an artifact or capability expected to be found on an endpoint or evidence collected related to the software component. This can be used to represent concepts not addressed directly by the directory, file, or process items. Examples include: registry keys, bound ports, etc. The equivalent construct in [SWID] is currently under specified. As a result, this item might be further defined through extension in the future.

- size (index 20): The file's size in bytes.

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

- 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.

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

- fs-name (index 24): The name of the directory or file without any path information. This aligns with a file "name" in [SWID].

- 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 allows a hierarchy of directory and file items to be 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. This aligns with a process "name" in [SWID].

- 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 2.2.

- o $$file-extension: This CDDL socket can be used to extend the file-entry group model. See Section 2.2.

- o $$directory-extension: This CDDL socket can be used to extend the directory-entry group model. See Section 2.2.

- o $$process-extension: This CDDL socket can be used to extend the process-entry group model. See Section 2.2.

- o $$resource-extension: This CDDL socket can be used to extend the resource-entry group model. See Section 2.2.

2.9.3. The payload-entry Map

The CDDL for the payload-entry map follows:

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

The following describes each child item of this group.

- o global-attributes: The global-attributes group described in Section 2.5.

- o resource-collection: The resource-collection group described in Section 2.9.2.

- o $$payload-extension: This CDDL socket can be used to extend the payload-entry group model. See Section 2.2.

2.9.4. The evidence-entry Map

The CDDL for the evidence-entry map follows:

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

date = 35
device-id = 36
```

[QUESTION: Is "time" a correct representation of XSD:date?]

The following describes each child item of this group.

- o global-attributes: The global-attributes group described in Section 2.5.

- o resource-collection: The resource-collection group described in Section 2.9.2.

- o date (index 35): The date and time the information was collected pertaining to the evidence item.

- o device-id (index 36): The endpoint's string identifier from which the evidence was collected.

- o $$evidence-extension: This CDDL socket can be used to extend the evidence-entry group model. See Section 2.2.

2.10. Full CDDL Specification

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

```
<CODE BEGINS>
concise-swid-tag = {
  tag-id => text / bstr .size 16,
  tag-version => integer,
  ? corpus => bool,
  ? patch => bool,
  ? supplemental => bool,
  software-name => text,
  ? software-version => text,
  ? version-scheme => $version-scheme,
  ? media => text,
  ? software-meta => software-meta-entry / [ 2* software-meta-entry ],
  entity => entity-entry / [ 2* entity-entry ],
  ? link => link-entry / [ 2* link-entry ],
  ? payload-or-evidence,
  global-attributes,
  * $$coswid-extension,
}

payload-or-evidence //= ( payload => payload-entry )
payload-or-evidence //= ( payload => [ 2* payload-entry ] )
payload-or-evidence //= ( evidence => evidence-entry )
payload-or-evidence //= ( evidence => [ 2* evidence-entry ] )

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

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

link-entry = {
  ? artifact => text,
  href => any-uri,
  ? media => text,
  ? ownership => $ownership,
  rel => $rel,
  ? media-type => text,
  ? use => $use,
  global-attributes,
  * $$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 /= -256..64436 / text

$use /= optional
```

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

software-meta-entry = {
  ? activation-status => text,
  ? channel-type => text,
  ? colloquial-version => text,
  ? description => text,
  ? edition => text,
  ? entitlement-data-required => bool,
  ? entitlement-key => text,
  ? generator => text,
  ? persistent-id => text,
  ? product => text,
  ? product-family => text,
  ? revision => text,
  ? summary => text,
  ? unspsc-code => text,
  ? unspsc-version => text,
  global-attributes,
  * $$software-meta-extension,
}

path-elements-group = ( ? directory => directory-entry / [ 2* directory-
entry ],
                        ? file => file-entry / [ 2* file-entry ],
                      )

resource-collection = (
  path-elements-group,
  ? process => process-entry / [ 2* process-entry ],
  ? resource => resource-entry / [ 2* resource-entry ],
  * $$resource-collection-extension,
)

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

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

```
}
```

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

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

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

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

evidence-entry = {
  resource-collection,
  ? date => time,
  ? device-id => text,
  global-attributes,
  * $$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
entity-name = 31
reg-id = 32
role = 33
thumbprint = 34
date = 35
device-id = 36
artifact = 37
href = 38
ownership = 39
rel = 40
media-type = 41
use = 42
activation-status = 43
channel-type = 44
colloquial-version = 45
description = 46
edition = 47
entitlement-data-required = 48
entitlement-key = 49
generator = 50
persistent-id = 51
product = 52
product-family = 53
revision = 54
summary = 55
unspsc-code = 56
unspsc-version = 57

; "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
maintainer=6

; "ownership" integer indexes
shared=1
private=2
abandon=3

; "rel" integer indexes
ancestor=1
component=2
feature=3
installationmedia=4
packageinstaller=5
parent=6
patches=7
requires=8
see-also=9
supersedes=10
; supplemental=11 ; this is already defined earlier

; "use" integer indexes
optional=1
required=2
recommended=3
<CODE ENDS>
```

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".

Note: Multiple of the corpus, patch, and supplemental items can have values set as "true". The rules above provide a means to determine the tag's type in such a case. For example, a SWID or CoSWID tag for a patch installer might have both corpus and patch items set to "true". In such a case, the tag is a "Corpus Tag". The tag installed by this installer would have only the patch item set to "true", making the installed tag type a "Patch Tag".

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.

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

Table 3: Version Scheme Values

[TODO: What text do we need to include to get a waiver to use SEMVER as a normative requirement?]

The values above are registered in the IANA "Software Tag Version Scheme Values" registry defined in Section Section 5.2.4. Additional entries will likely be registered over time in this registry.

These version schemes have partially overlapping value spaces. The following guidelines help to ensure that the most specific version-scheme is used:

- "decimal" and "multipartnumeric" partially overlap in their value space when a value matches a decimal number. When a corresponding software-version item's value falls within this overlapping value space, the "decimal" version scheme SHOULD be used.

- "multipartnumeric" and "semver" partially overlap in their value space when a "multipartnumeric" value matches the semantic versioning syntax. When a corresponding software-version item's value falls within this overlapping value space, the "semver" version scheme SHOULD be used.

- o "alphanumeric" and other version schemes might overlap in their value space. When a corresponding software-version item's value falls within this overlapping value space, the other version scheme SHOULD be used instead of "alphanumeric".

4.2. Entity Role Values

The following table indicates the index value to use for the entity-entry group's role item (see Section 2.6). 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.

Index	Role Name	Definition
1	tagCreator	The person or organization that created the containing SWID or CoSWID tag
2	softwareCreator	The person or organization entity that created the software component.
3	aggregator	From [SWID], "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)."
4	distributor	From [SWID], "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."
5	licensor	From [SAM] as "software licensor", a "person or organization who owns or holds the rights to issue a software license for a specific software [component]"
6	maintainer	The person or organization that is responsible for coordinating and making updates to the source code for the software component. This SHOULD be used when the "maintainer" is a different person or organization than the original "softwareCreator".

Table 4: Entity Role Values

The values above are registered in the IANA "Software Tag Entity Role Values" registry defined in Section Section 5.2.5. Additional values will likely be registered over time. Additionally, the index values 128 through 255 and the name prefix "x_" 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 2.7). 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.

Index	Ownership Type	Definition
1	abandon	If the software component referenced by the CoSWID tag is uninstalled, then the referenced software SHOULD NOT be uninstalled
2	private	If the software component referenced by the CoSWID tag is uninstalled, then the referenced software SHOULD be uninstalled as well.
3	shared	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.

Table 5: Link Ownership Values

The values above are registered in the IANA "Software Tag Link Ownership Values" registry defined in Section Section 5.2.6. Additional values will likely be registered over time. Additionally, the index values 128 through 255 and the name prefix "x_" 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 2.7). 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.

Index	Relationship Type	Definition
1	ancestor	The link references a software tag for a previous release of this software. This can be useful to define an upgrade path.
2	component	The link references a software tag for a separate component of this software.
3	feature	The link references a configurable feature of this software that can be enabled or disabled without changing the installed files.
4	installationmedia	The link references the installation package that can be used to install this software.
5	packageinstaller	The link references the installation software needed to install this software.
6	parent	The link references a software tag that is the parent of the referencing tag. This relationship can be used when multiple software components are part of a software bundle, where the "parent" is the software tag for the bundle, and each child is a "component". In such a case, each child component can provide a "parent" link relationship to the bundle's software tag, and the bundle can provide a "component" link relationship to each child software component.
7	patches	The link references a software tag that the referencing software patches. Typically only used for patch tags (see Section 1.1).
8	requires	The link references a prerequisite for installing this software. A patch tag (see Section 1.1) can use this to represent base software or another

		patch that needs to be installed first.
9	see-also	The link references other software that may be of interest that relates to this software.
10	supersedes	The link references another software that this software replaces. A patch tag (see Section 1.1) can use this to represent another patch that this patch incorporates or replaces.
11	supplemental	The link references a software tag that the referencing tag supplements. Used on supplemental tags (see Section 1.1).

Table 6: Link Relationship Values

The values above are registered in the IANA "Software Tag Link Relationship Values" registry defined in Section Section 5.2.7. Additional values will likely be registered over time. Additionally, the index values 32768 through 65535 and the name prefix "x_" 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 2.7). 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.

Index	Use Type	Definition
1	optional	From [SWID], "Not absolutely required; the [Link]'d software is installed only when specified."
2	required	From [SWID], "The [Link]'d software is absolutely required for an operation software installation."
3	recommended	From [SWID], "Not absolutely required; the [Link]'d software is installed unless specified otherwise."

Table 7: Link Use Values

The values above are registered in the IANA "Software Tag Link Use Values" registry defined in Section Section 5.2.8. Additional values will likely be registered over time. Additionally, the index values 128 through 255 and the name prefix "x_" have been reserved for private use.

5. IANA Considerations

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

5.1. CoSWID Items Registry

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

This document defines 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

Table 8: CoSWID Items Registration Procedures

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.

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

20	size	RFC-AAAA
21	file-version	RFC-AAAA
22	key	RFC-AAAA
23	location	RFC-AAAA
24	fs-name	RFC-AAAA
25	root	RFC-AAAA
26	path-elements	RFC-AAAA
27	process-name	RFC-AAAA
28	pid	RFC-AAAA
29	type	RFC-AAAA
31	entity-name	RFC-AAAA
32	reg-id	RFC-AAAA
33	role	RFC-AAAA
34	thumbprint	RFC-AAAA
35	date	RFC-AAAA
36	device-id	RFC-AAAA
37	artifact	RFC-AAAA
38	href	RFC-AAAA
39	ownership	RFC-AAAA
40	rel	RFC-AAAA
41	media-type	RFC-AAAA
42	use	RFC-AAAA
43	activation-status	RFC-AAAA
44	channel-type	RFC-AAAA

45	colloquial-version	RFC-AAAA
46	description	RFC-AAAA
47	edition	RFC-AAAA
48	entitlement-data-required	RFC-AAAA
49	entitlement-key	RFC-AAAA
50	generator	RFC-AAAA
51	persistent-id	RFC-AAAA
52	product	RFC-AAAA
53	product-family	RFC-AAAA
54	revision	RFC-AAAA
55	summary	RFC-AAAA
56	unspsc-code	RFC-AAAA
57	unspsc-version	RFC-AAAA
58-4294967295	Unassigned	

Table 9: CoSWID Items Inital Registrations

5.2. Software Tag Values Registries

The following IANA registries provide a mechanism for new values to be added over time to common enumerations used by SWID and CoSWID.

5.2.1. Registration Procedures

The following registries allow for the registration of index values and names. New registrations will be permitted through either the Standards Action policy or the Specification Required policy [BCP26]. New index values will be provided on a First Come First Served as defined by [BCP26].

The following registries also reserve the integer-based index values in the range of -1 to -256 for private use as defined by [BCP26] in Section 4.1. This allows values -1 to -24 to be expressed as a

single uint_8t in CBOR, and values -25 to -256 to be expressed using an additional uint_8t in CBOR.

5.2.2. Private Use of Index and Name Values

The integer-based index values in the private use range (-1 to -256) are intended for testing purposes and closed environments; values in other ranges SHOULD NOT be assigned for testing.

For names that correspond to private use index values, an Internationalized Domain Name prefix MUST be used to prevent name conflicts using the form:

" domain.prefix-name "

Where "domain.prefix" MUST be a valid Internationalized Domain Name as defined by [RFC5892], and "name" MUST be a unique name within the namespace defined by the "domain.prefix". Use of a prefix in this way allows for a name to be used initially in the private use range, and to be registered at a future point in time. This is consistent with the guidance in [BCP178].

5.2.3. Expert Review Guidelines

Designated experts MUST ensure that new registration requests meet the following additional guidelines:

- The requesting specification MUST provide a clear semantic definition for the new entry. This definition MUST clearly differentiate the requested entry from other previously registered entries.

- The requesting specification MUST describe the intended use of the entry, including any co-constraints that exist between the use of the entry's index value or name, and other values defined within the SWID/CoSWID model.

- Index values and names outside the private use space MUST NOT be used without registration. This is considered squatting and SHOULD be avoided. Designated experts MUST ensure that reviewed specifications register all appropriate index values and names.

- Standards track documents MAY include entries registered in the range reserved for entries under the Specification Required policy. This can occur when a standards track document provides further guidance on the use of index values and names that are in common use, but were not registered with IANA. This situation SHOULD be avoided.

- o All registered names MUST be valid according to the XML Schema NMTOKEN data type (see [W3C.REC-xmlschema-2-20041028] Section 3.3.4). This ensures that registered names are compatible with the SWID format [SWID] where they are used.

- o Registration of vanity names SHOULD be discouraged. The requesting specification MUST provide a description of how a requested name will allow for use by multiple stakeholders.

5.2.4. Software Tag Version Scheme Values Registry

This document establishes a new registry titled "Software Tag Version Scheme Values". This registry provides index values for use as version-scheme item values in this document and version scheme names for use in [SWID].

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

This registry uses the registration procedures defined in Section 5.2.1 with the following associated ranges:

Range	Registration Procedures
0-16383	Standards Action
16384-65535	Specification Required

Table 10: CoSWID Version Scheme Registration Procedures

Assignments MUST consist of an integer Index value, the Version Scheme Name, and a reference to the defining specification.

Initial registrations for the "Software Tag Version Scheme Values" registry are provided below, which are derived from the textual version scheme names defined in [SWID].

Index	Version Scheme Name	Specification
0	Reserved	
1	multipartnumeric	See Section 4.1
2	multipartnumeric+suffix	See Section 4.1
3	alphanumeric	See Section 4.1
4	decimal	See Section 4.1
5-16383	Unassigned	
16384	semver	[SEMVER]
16385-65535	Unassigned	

Table 11: CoSWID Version Scheme Initial Registrations

Registrations MUST conform to the expert review guidelines defined in Section 5.2.3.

Designated experts MUST also ensure that newly requested entries define a value space for the corresponding version item that is unique from other previously registered entries. Note: The initial registrations violate this requirement, but are included for backwards compatibility with [SWID]. Guidelines on how to deconflict these value spaces are defined in Section Section 4.1.

5.2.5. Software Tag Entity Role Values Registry

This document establishes a new registry titled "Software Tag Entity Role Values". This registry provides index values for use as entity-entry role item values in this document and entity role names for use in [SWID].

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

This registry uses the registration procedures defined in Section 5.2.1 with the following associated ranges:

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

Table 12: CoSWID Entity Role Registration Procedures

Assignments consist of an integer Index value, a Role Name, and a reference to the defining specification.

Initial registrations for the "Software Tag Entity Role Values" registry are provided below, which are derived from the textual entity role names defined in [SWID].

Index	Role Name	Specification
0	Reserved	
1	tagCreator	See Section 4.2
2	softwareCreator	See Section 4.2
3	aggregator	See Section 4.2
4	distributor	See Section 4.2
5	licensor	See Section 4.2
6	maintainer	See Section 4.2
7-255	Unassigned	

Table 13: CoSWID Entity Role Initial Registrations

Registrations MUST conform to the expert review guidelines defined in Section 5.2.3.

5.2.6. Software Tag Link Ownership Values Registry

This document establishes a new registry titled "Software Tag Link Ownership Values". This registry provides index values for use as link-entry ownership item values in this document and link ownership names for use in [SWID].

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

This registry uses the registration procedures defined in Section 5.2.1 with the following associated ranges:

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

Table 14: CoSWID Link Ownership Registration Procedures

Assignments consist of an integer Index value, an Ownership Type Name, and a reference to the defining specification.

Initial registrations for the "Software Tag Link Ownership Values" registry are provided below, which are derived from the textual entity role names defined in [SWID].

Index	Ownership Type Name	Definition
0	Reserved	
1	abandon	See Section 4.3
2	private	See Section 4.3
3	shared	See Section 4.3
4-255	Unassigned	

Table 15: CoSWID Link Ownership Inital Registrations

Registrations MUST conform to the expert review guidelines defined in Section 5.2.3.

5.2.7. Software Tag Link Relationship Values Registry

This document establishes a new registry titled "Software Tag Link Relationship Values". This registry provides index values for use as link-entry rel item values in this document and link ownership names for use in [SWID].

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

This registry uses the registration procedures defined in Section 5.2.1 with the following associated ranges:

Range	Registration Procedures
0-32767	Standards Action
32768-65535	Specification Required

Table 16: CoSWID Link Relationship Registration Procedures

Assignments consist of an integer Index value, the Relationship Type Name, and a reference to the defining specification.

Initial registrations for the "Software Tag Link Relationship Values" registry are provided below, which are derived from the link relationship values defined in [SWID].

Index	Relationship Type Name	Specification
0	Reserved	
1	ancestor	See Section 4.4
2	component	See Section 4.4
3	feature	See Section 4.4
4	installationmedia	See Section 4.4
5	packageinstaller	See Section 4.4
6	parent	See Section 4.4
7	patches	See Section 4.4
8	requires	See Section 4.4
9	see-also	See Section 4.4
10	supersedes	See Section 4.4
11	supplemental	See Section 4.4
12-65535	Unassigned	

Table 17: CoSWID Link Relationship Initial Registrations

Registrations MUST conform to the expert review guidelines defined in Section 5.2.3.

Designated experts MUST also ensure that a newly requested entry documents the URI schemes allowed to be used in an href associated with the link relationship and the expected resolution behavior of these URI schemes. This will help to ensure that applications processing software tags are able to interoperate when resolving resources referenced by a link of a given type.

5.2.8. Software Tag Link Use Values Registry

This document establishes a new registry titled "Software Tag Link Use Values". This registry provides index values for use as link-entry use item values in this document and link use names for use in [SWID].

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

This registry uses the registration procedures defined in Section 5.2.1 with the following associated ranges:

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

Table 18: CoSWID Link Use Registration Procedures

Assignments consist of an integer Index value, the Link Use Type Name, and a reference to the defining specification.

Initial registrations for the "Software Tag Link Use Values" registry are provided below, which are derived from the link relationship values defined in [SWID].

Index	Link Use Type Name	Specification
0	Reserved	
1	optional	See Section 4.5
2	required	See Section 4.5
3	recommended	See Section 4.5
4-255	Unassigned	

Table 19: CoSWID Link Use Initial Registrations

Registrations MUST conform to the expert review guidelines defined in Section 5.2.3.

5.3. swid+cbor Media Type Registration

[TODO: Per Section 5.1 of RFC6838, was a message sent to media-types@iana.org for preliminary review? I didn't see it on that mailing list (did I miss it?). Please kick that off.]

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 RFC7049 Section 7.5.

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]:

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

Table 20: 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:

Tag	Data Item	Semantics
1398229316	map	Concise Software Identifier (CoSWID) [RFC-AAAA]

Table 21: CoSWID CBOR Tag

5.6. URI Scheme Registrations

The ISO 19770-2:2015 SWID specification describes use of the "swid" and "swidpath" URI schemes, which are currently in use in implementations. This document continues this use for CoSWID. The following subsections provide registrations for these schemes in to ensure that a permanent registration exists for these schemes that is suitable for use in the SWID and CoSWID specifications.

[TODO: Per Step 3.2 of Section 7.2 of RFC7595, has this been sent to uri-review@ietf.org? I didn't see it on that mailing list (did I miss it?). Please kick that off.]

5.6.1. "swid" URI Scheme Registration

There is a need for a scheme name that can be used in URIs that point to a specific software tag by that tag's tag-id, such as the use of the link entry as described in Section Section 2.7) of this document. Since this scheme is used in a standards track document and an ISO standard, this scheme needs to be used without fear of conflicts with current or future actual schemes. The scheme "swid" is hereby registered as a 'permanent' scheme for that purpose.

The "swid" scheme is specified as follows:

Scheme name: FIXME

Status: Permanent

Applications/protocols that use this scheme name: FIXME

Contact: FIXME

Change controller: FIXME

References: FIXME

5.6.2. "swid" URI Scheme Specification [TODO: FIXME: has to move out of registration]

Scheme syntax: The scheme specific part consists of a software tag's tag-id that is URI encoded according to [RFC3986] Section 2.1. The following expression is a valid example:

<swid:2df9de35-0aff-4a86-ace6-f7dddd1ade4c>

Scheme semantics: URIs in the "swid" scheme are to be used to reference a software tag by its tag-id. A tag-id referenced in this way can be used to identify the tag resource in the context of where it is referenced from. For example, when a tag is installed on a given device, that tag can reference related tags on the same device using this URI scheme.

Encoding considerations: See Section 2.5 of [RFC3986] for guidelines.

Interoperability considerations: None.

Security considerations: None.

5.6.3. "swidpath" URI Scheme Registration

There is a need for a scheme name that can be used in URIs to identify a collection of specific software tags with data elements that match an XPath expression, such as the use of the link entry as described in Section Section 2.7) of this document. Since this scheme is used in a standards track document and an ISO standard, this scheme needs to be used without fear of conflicts with current or future actual schemes. The scheme "swidpath" is hereby registered as a 'permanent' scheme for that purpose.

The "swidpath" scheme is specified as follows:

Scheme name: FIXME

Status: Permanent

Applications/protocols that use this scheme name: FIXE

Contact: FIXME

Change controller: FIXME

References: FIXME

5.6.4. "swidpath" URI Scheme Specification [TODO: FIXME: has to move out of registration]

Scheme syntax: The scheme specific part consists of an XPath expression as defined by [W3C.REC-xpath20-20101214]. The included XPath expression will be URI encoded according to [RFC3986] Section 2.1.

Scheme semantics: URIs in the "swidpath" scheme are to be used specify the data that must be found in a given software tag for that tag to be considered a matching tag to be included in the identified tag collection. Tags to be evaluated include all tags in the context of where the tag is referenced from. For example, when a tag is installed on a given device, that tag can reference related tags on the same device using this URI scheme. A tag is matching if the XPath evaluation result value has an effective boolean value of "true" according to [W3C.REC-xpath20-20101214] Section 2.4.3. rence related tags on the same device using this URI scheme.

Encoding considerations: See Section 2.5 of [RFC3986] for guidelines.

Interoperability considerations: None.

Security considerations: None.

5.7. CoSWID Model for use in SWIMA Registration

The Software Inventory Message and Attributes (SWIMA) for PA-TNC specification [RFC8412] defines a standardized method for collecting an endpoint device's software inventory. A CoSWID can provide evidence of software installation which can then be used and exchanged with SWIMA. This registration adds a new entry to the IANA "Software Data Model Types" registry defined by [RFC8412] to support CoSWID use in SWIMA as follows:

Pen: 0

Integer: TBD2

Name: Concise Software Identifier (CoSWID)

Defining Specification: RFC-AAAA

Deriving Software Identifiers:

A Software Identifier generated from a CoSWID tag is expressed as a concatenation of the form:

TAG_CREATOR_REGID "_" "_" UNIQUE_ID

Where TAG_CREATOR_REGID is the reg-id item value of the tag's entity item having the role value of 1 (corresponding to "tag creator"), and the UNIQUE_ID is the same tag's tag-id item. If the tag-id item's value is expressed as a 16 byte binary string, the UNIQUE_ID MUST be represented using the UUID string representation defined in [RFC4122] including the "urn:uuid:" prefix.

The TAG_CREATOR_REGID and the UNIQUE_ID are connected with a double underscore (_), without any other connecting character or whitespace.

6. Security Considerations

CoSWID tags contain public information about software components and, as such, do not need to be protected against disclosure on an endpoint. Similarly, CoSWID 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 CoSWID tags focus on the application of CoSWID tags to address security challenges, and the possible disclosure of the results of those applications.

A tag is considered "authoritative" if the CoSWID tag was created by the software provider. An authoritative CoSWID tag contains information about a software component provided by the maintainer of the software component, who is expected to be an expert in their own software. Thus, authoritative CoSWID tags can be trusted to represent authoritative information about the software component.

A signed CoSWID tag (see Appendix A) whose signature has been validated can be relied upon to be unchanged since it was signed. By contrast, the data contained in unsigned tags cannot be trusted to be unmodified.

When an authoritative tag is signed, the software provider can be authenticated as the originator of the signature. A trustworthy association between the signature and the originator of the signature can be established via trust anchors. A certification path between a trust anchor and a certificate including a pub-key enabling the validation of a tag signature can realize the assessment of trustworthiness of an authoritative tag. Having a signed authoritative CoSWID 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.

CoSWID 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, CoSWID tags can be easier to add or remove. On such systems, it can be possible to add or remove CoSWID tags in a way that does not reflect the actual presence or absence of corresponding software components. Similarly, not all software products automatically install CoSWID tags, so products can be present on an endpoint without providing a corresponding SWID tag. As such, any collection of CoSWID tags cannot automatically be assumed to represent either a complete or fully accurate representation of the software inventory of the endpoint. However, especially on endpoint devices that more strictly control the ability to add or remove applications, CoSWID tags are an easy way to provide an preliminary understanding of that endpoint's software inventory.

Any report of an endpoint's CoSWID 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, CoSWID 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 CoSWID tag information that has been collected from an endpoint in transit and at rest, not because those tags individually contain sensitive information, but because the collection of CoSWID tags and their association with an endpoint reveals information about that endpoint's attack surface.

Finally, both the ISO-19770-2:2015 XML schema SWID definition and the CoSWID CDDL specification allow for the construction of "infinite" tags with link item loops or tags that contain malicious content with the intent of creating non-deterministic states during validation or processing of those tags. While software providers are unlikely to do this, CoSWID tags can be created by any party and the CoSWID tags collected from an endpoint could contain a mixture of vendor and non-vendor created tags. For this reason, a CoSWID tag might contain potentially malicious content. Input sanitization and loop detection are two ways that implementations can address this concern.

7. Acknowledgments

This document draws heavily on the concepts defined in the ISO/IEC 19770-2:2015 specification. The authors of this document are grateful for the prior work of the 19770-2 contributors.

We are also grateful to the careful reviews provided by ...

8. Change Log

[THIS SECTION TO BE REMOVED BY THE RFC EDITOR.]

Changes from version 12 to version 14:

- Moved key identifier to protected COSE header
- Fixed index reference for hash
- Removed indirection of CDDL type definition for filesystem-item
- Fixed quantity of resource and process
- Updated resource-collection

- Renamed socket name in software-meta to be consistent in naming
- Aligned excerpt examples in I-D text with full CDDL
- Fixed titels where title was referring to group instead of map
- Added missig date in SEMVER
- Fixed root cardinality for file and directory, etc.
- Transformed path-elements-entry from map to group for re-usability
- Scrubbed IANA Section
- Removed redundant supplemental rule
- Aligned discrepancy with ISO spec.
- Addressed comments on typos.
- Fixed kramdown nits and BCP reference.
- Addressed comments from WGLC reviewers.

Changes in version 12:

- Addressed a bunch of minor editorial issues based on WGLC feedback.
- Added text about the use of UTF-8 in CoSWID.
- Adjusted tag-id to allow for a UUID to be provided as a bstr.
- Cleaned up descriptions of index ranges throughout the document, removing discussion of 8 bit, 16 bit, etc.
- Adjusted discussion of private use ranges to use negative integer values and to be more clear throughout the document.
- Added discussion around resolving overlapping value spaces for version schemes.
- Added a set of expert review guidelines for new IANA registries created by this document.
- Added new registrations for the "swid" and "swidpath" URI schemes, and for using CoSWID with SWIMA.

Changes from version 03 to version 11:

- o Reduced representation complexity of the media-entry type and removed the Section describing the older data structure.

- o Added more signature schemes from COSE

- o Included a minimal required set of normative language

- o Reordering of attribute name to integer label by priority according to semantics.

- o Added an IANA registry for CoSWID items supporting future extension.

- o Cleaned up IANA registrations, fixing some inconsistencies in the table labels.

- o Added additional CDDL sockets for resource collection entries providing for additional extension points to address future SWID/ CoSWID extensions.

- o Updated Section on extension points to address new CDDL sockets and to reference the new IANA registry for items.

- o Removed unused references and added new references to address placeholder comments.

- o Added table with semantics for the link ownership item.

- o Clarified language, made term use more consistent, fixed references, and replacing lowercase RFC2119 keywords.

Changes from version 02 to version 03:

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

Changes from version 01 to version 02:

- o Enforced a more strict separation between the core CoSWID definition and additional usage by moving content to corresponding appendices.

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

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

- o Minor reordering of map members

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

Changes from version 00 to version 01:

- o Ambiguity between evidence and payload eliminated by introducing explicit members (while still

- o allowing for "empty" SWID tags)

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

- o 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:

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

- o Fixed broken multi-map members

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

- o Fixed X.1520 reference

- o 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

- 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.
- 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.

- o Added a Section describing the CoSWID model in detail.
- o Created IANA registries for entity-role and version-scheme

Changes from version 02 to version 03:

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

Changes from version 01 to version 02:

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

Changes from version 00 to version 01:

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

9. References

9.1. Normative References

[BCP178] Saint-Andre, P., Crocker, D., and M. Nottingham, "Deprecating the "X-" Prefix and Similar Constructs in Application Protocols", BCP 178, RFC 6648, DOI 10.17487/RFC6648, June 2012, <https://www.rfc-editor.org/info/rfc6648>.

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, DOI 10.17487/RFC5198, March 2008, <https://www.rfc-editor.org/info/rfc5198>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[RFC5892] Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, DOI 10.17487/RFC5892, August 2010, <https://www.rfc-editor.org/info/rfc5892>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

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

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://www.rfc-editor.org/info/rfc8288>.

[RFC8412] Schmidt, C., Haynes, D., Coffin, C., Waltermire, D., and J. Fitzgerald-McKay, "Software Inventory Message and Attributes (SWIMA) for PA-TNC", RFC 8412, DOI 10.17487/RFC8412, July 2018, <https://www.rfc-editor.org/info/rfc8412>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[SAM] "Information technology - Software asset management - Part 5: Overview and vocabulary", ISO/IEC 19770-5:2015, November 2013.

[SEMVER] Preston-Werner, T., "Semantic Versioning 2.0.0", <https://semver.org/spec/v2.0.0.html>.

[SWID] "Information technology - Software asset management - Part 2: Software identification tag", ISO/IEC 19770-2:2015, October 2015.

[UNSPSC] "United Nations Standard Products and Services Code", October 2020, <https://www.unspsc.org/>.

[W3C.REC-css3-mediaqueries-20120619] Rivoal, F., "Media Queries", World Wide Web Consortium Recommendation REC-css3-mediaqueries-20120619, June 2012, <https://www.w3.org/TR/2012/REC-css3-mediaqueries-20120619>.

[W3C.REC-xmlschema-2-20041028] Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, October 2004, <https://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

[W3C.REC-xpath20-20101214] Berglund, A., Boag, S., Chamberlin, D., Fernandez, M., Kay, M., Robie, J., and J. Simeon, "XML Path Language (XPath) 2.0 (Second Edition)", World Wide Web Consortium Recommendation REC-xpath20-20101214, December 2010, <https://www.w3.org/TR/2010/REC-xpath20-20101214>.

[X.1520] "Recommendation ITU-T X.1520 (2014), Common vulnerabilities and exposures", April 2011.

9.2. Informative References

[CamelCase]
"UpperCamelCase", August 2014, <http://wiki.c2.com/?CamelCase>.

[I-D.ietf-rats-architecture]
Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote Attestation Procedures Architecture", draft-ietf-rats-architecture-07 (work in progress), October 2020.

[KebabCase]
"KebabCase", December 2014, <http://wiki.c2.com/?KebabCase>.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <https://www.rfc-editor.org/info/rfc3444>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC8322] Field, J., Banghart, S., and D. Waltermire, "Resource-Oriented Lightweight Information Exchange (ROLIE)", RFC 8322, DOI 10.17487/RFC8322, February 2018, <https://www.rfc-editor.org/info/rfc8322>.

[RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <https://www.rfc-editor.org/info/rfc8520>.

[SWID-GUIDANCE]
Waltermire, D., Cheikes, B., Feldman, L., and G. Witte, "Guidelines for the Creation of Interoperable Software Identification (SWID) Tags", NISTIR 8060, April 2016, <https://doi.org/10.6028/NIST.IR.8060>.

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.

```
<CODE BEGINS>
signed-coswid = #6.18(COSE-Sign1-coswid)

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

protected-signed-coswid-header = {
    1 => int,                       ; algorithm identifier
    3 => "application/swid+cbor",
    4 => bstr,                      ; key identifier
    * cose-label => cose-values,
}

unprotected-signed-coswid-header = {
    * cose-label => cose-values,
}

COSE-Sign1-coswid = [
    protected: bstr .cbor protected-signed-coswid-header,
    unprotected: unprotected-signed-coswid-header,
    payload: bstr .cbor concise-swid-tag,
    signature: bstr,
]
<CODE ENDS>
```

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