

SACM Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 12, 2019

H. Birkholz
Fraunhofer SIT
J. Fitzgerald-McKay
Department of Defense
C. Schmidt
The MITRE Corporation
D. Waltermire
NIST
November 08, 2018

Concise Software Identifiers
draft-ietf-sacm-coswid-08

Abstract

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 12, 2019.

Copyright Notice

Copyright (c) 2018 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 Tag Lifecycle	4
1.2.	Concise SWID Extensions	6
1.3.	Requirements Notation	7
2.	Concise SWID Data Definition	7
2.1.	The concise-software-identity Object	8
2.1.1.	Determining the tag type	12
2.1.2.	concise-software-identity Co-constraints	12
2.2.	The global-attributes Group	13
2.3.	The media Object	13
2.4.	The entity Object	14
2.5.	The link Object	15
2.6.	The software-meta Object	17
2.7.	The Resource Collection Definition	19
2.7.1.	The hash-entry Array	19
2.7.2.	The resource-collection Group	20
2.7.3.	The payload Object	23
2.7.4.	The evidence Object	23
2.8.	Full CDDL Definition	24
3.	CoSWID Indexed Label Values	29
3.1.	Version Scheme	29
3.2.	Entity Role Values	30
3.3.	Use Values	30
4.	IANA Considerations	31
4.1.	SWID/CoSWID Version Schema Values Registry	31
4.2.	SWID/CoSWID Entity Role Values Registry	32
4.3.	SWID/CoSWID Link Use Values Registry	33
4.4.	Media Type Registration	34
4.4.1.	swid+cbor Media Type Registration	34
4.5.	CoAP Content-Format Registration	35
4.6.	CBOR Tag Registration	36
5.	Security Considerations	36
6.	Acknowledgments	38
7.	Change Log	38
8.	Contributors	41
9.	References	41

9.1.	Normative References	41
9.2.	Informative References	42
Appendix A.	CoSWID Attributes for Firmware (label 60)	43
Appendix B.	Signed Concise SWID Tags using COSE	46
	Authors' Addresses	47

[1.](#) Introduction

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

- o Software Inventory Management, a part of a Software Asset Management [[SAM](#)] process, which requires an accurate list of discernible deployed software components.
- o Vulnerability Assessment, which requires a semantic link between standardized vulnerability descriptions and software components installed on IT-assets [[X.1520](#)].
- o Remote Attestation, which requires a link between reference integrity measurements (RIM) and security logs of measured software components [[I-D.birkholz-tuda](#)].

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

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

1.1. The SWID 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 following is an excerpt (with some modifications and reordering) from NIST Interagency Report (NISTIR) 8060: Guidelines for the Creation of Interoperable SWID Tags [[SWID-GUIDANCE](#)], which describes the tag types used within the lifecycle defined in ISO-19770-2:2015. This information is included here to provide a concise reference for the use of CoSWIDs in the software lifecycle.

The SWID specification defines four types of SWID tags: primary, patch, corpus, and supplemental.

1. Primary Tag - A SWID or CoSWID tag that identifies and describes a software component is installed on a computing device. A primary tag is intended to be installed on an endpoint along with the corresponding software component.
2. Patch Tag - A SWID or CoSWID tag that identifies and describes an installed patch which has made incremental changes to a software component installed on an endpoint. A patch tag is intended to be installed on an endpoint along with the corresponding software component patch.
3. Corpus Tag - A SWID or CoSWID tag that identifies and describes an installable software component in its pre-installation state. A corpus tag can be used to represent metadata about an installation package or installer for a software component, a software update, or a patch.
4. Supplemental Tag - A SWID or CoSWID tag that allows additional information to be associated with a referenced SWID tag. This helps to ensure that SWID Primary and Patch Tags provided by a software provider are not modified by software management tools, while allowing these tools to provide their own software metadata.

installation to provide information about software fixes deployed along with the base software installation.

- * **Software Patching.** When a new patch is applied to the software component, a new patch tag is provided, supplying details about the patch and its dependencies. While not illustrated in the 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.
- * **Software Removal.** Upon removal of the software component, relevant SWID tags are removed. This removal event can trigger timely updates to software inventory reflecting the removal of the product and any associated patch or supplemental tags.

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

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 Extensions

This document defines the CoSWID format, a more concise representation of SWID information in the Concise Binary Object Representation (CBOR) [[RFC7049](#)]. The structure of a CoSWID is described via the Concise Data Definition Language (CDDL) [[I-D.ietf-cbor-cddl](#)]. The resulting CoSWID data definition is aligned in the information able to be expressed with the XML schema definition of ISO-19770-2:2015 [[SWID](#)]. The vocabulary, i.e., the CDDL names of the types and members used in the CoSWID data definition, are mapped to more concise labels represented as small integer values. The names used in the CDDL data definition and the mapping to the CBOR representation using integer labels is based on

the vocabulary of the XML attribute and element names defined in ISO/IEC 19770-2:2015.

The corresponding CoSWID data definition includes two kinds of augmentation.

- o The explicit definition of types for attributes that are typically stored in the "any attribute" of an ISO-19770-2:2015 in XML representation. These are covered in [Section 2.2](#) and [model-any-element] of this document.
- o The inclusion of extension points in the CoSWID data definition that allow for additional uses of CoSWID tags that go beyond the original scope of ISO-19770-2:2015 tags. The following extension points are defined in this document:
 - * CoSWID Extension - Used to add new information structures to a CoSWID (defined in [Section 2.1](#)).
 - * Entity Extension - Used to add new new information structures to an entity (defined in [Section 2.4](#)).
 - * Payload Extension - Used to add new new information structures to a payload, such as new payload resource types (defined in [Section 2.7.3](#)).
 - * Evidence Extension - Used to add new new information structures to an evidence, such as new evidence resource types (defined in [Section 2.7.4](#)).

[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 [RFC 2119](#), [BCP 14](#) [[RFC2119](#)].

[2.](#) Concise SWID Data Definition

The following is a CDDL representation for a CoSWID tag. This CDDL representation is intended to be parallel to the XML schema definition in the ISO/IEC 19770-2:2015 [[SWID](#)] specification, allowing both SWID and CoSWID tags to represent a common set of SWID information and to support all SWID tag use cases. To achieve this end, the CDDL representation includes every SWID tag field and attribute. The CamelCase notation used in the XML schema definition is changed to a hyphen-separated notation (e.g. ResourceCollection is named resource-collection) in the CoSWID data definition. This

deviation from the original notation used in the XML representation reduces ambiguity when referencing certain attributes in corresponding textual descriptions. An attribute referred by its name in CamelCase notation explicitly relates to XML SWID tags, an attribute referred by its name in hyphen-separated notation explicitly relates to CoSWID tags. This approach simplifies the composition of further work that reference both XML SWID and CoSWID documents.

Human-readable names of members in the CDDL data definition are mapped to integer indices via a block of rules at the bottom of the definition. The 67 character strings of the SWID vocabulary that would have to be stored or transported in full if using the original vocabulary are replaced.

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

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

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

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

[2.1.](#) The concise-software-identity Object

The CDDL for the main concise-software-identity object is as follows:


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

The following describes each child item of the concise-software-identity object model.

- 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 2.2](#).
- o tag-id (label 0): An textual identifier uniquely referencing a (composite) software component. The tag identifier MUST be globally unique. There are no strict guidelines on how this identifier is structured, but examples include a 16 byte GUID (e.g. class 4 UUID) [[RFC4122](#)], or a text string appended to a DNS domain name to ensure uniqueness across organizations.
- o tag-version (label 12): An integer value that indicates if a specific release of a software component has more than one tag that can represent that specific release. Typically, the initial value of this field is set to 0, and the value is monotonically increased for subsequent tags produced for the same software component release. This item is used when a CoSWID tag producer creates and releases an incorrect tag that they subsequently want to fix, but no underlying changes have been made to the product the CoSWID tag represents. This could happen if, for example, a patch is distributed that has a link reference that does not cover all the various software releases it can patch. A newer CoSWID tag for that patch can be generated and the tag-version value incremented to indicate that the data has been updated.
- o corpus (label 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".

- o patch (label 9): A boolean value that indicates if the tag identifies and describes an installed patch which has made incremental changes to a software component installed on a computing device. Typically, an installed patch has made a set of file modifications to pre-installed software, and does not alter the version number or the descriptive metadata of an installed software component. If a CoSWID tag is for a patch, it MUST contain the patch item and its value MUST be set to "true". If not provided the default value MUST be considered "false".
- o supplemental (label 11): A boolean value that indicates if the tag is providing additional information to be associated with another referenced SWID or CoSWID tag. Tags using this item help to ensure that primary and patch tags provided by a software provider are not modified by software management tools, while allowing these tools to provide their own software metadata for a software component. If a CoSWID tag is a supplemental tag, it MUST contain the supplemental item and its value MUST be set to "true". If not provided the default value MUST be considered "false".
- o swid-name (label 1): This textual item provides the software component name as it would typically be referenced. For example, what would be seen in the add/remove software dialog in an operating system, or what is specified as the name of a packaged software component or a patch identifier name.
- o software-version (label 13): A textual value representing the specific underlying release or development version of the software component.
- o version-scheme (label 14): An 8-bit integer or textual value representing the versioning scheme used for the software-version item. If an integer value is used it MUST be a value from the registry (see section [Section 4.1](#) or a value in the private use range: 32768-65,535.
- o media (label 10): This text value is a hint to the tag consumer to understand what this tag applies to. This item represents a query as defined by the W3C Media Queries Recommendation (see <http://www.w3.org/TR/css3-mediaqueries/>). A hint to the consumer of the link to what the target item is applicable for. Described in [Section 2.3](#).

- o software-meta-entry (label 5): An open-ended map of key/value data pairs. A number of predefined keys can be used within this item providing for common usage and semantics across the industry. The data definition of this entry allows for any additional attribute to be included, though it is recommended that industry norms for new attributes are defined and followed to the degree possible. Described in [Section 2.6](#).
- o entity-entry (label 2): Provides information about one or more organizations related to the software component referenced by this CoSWID tag. Described in [Section 2.4](#).
- o link-entry (label 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 , MUD resource , software download location, etc). This is modeled after the HTML "link" element. Described in [Section 2.5](#).
- o payload-entry (label 6): This item represents the software artifacts that may be installed on an endpoint when the software component is installed. Note that the payload may represent a superset of the software artifacts installed. Based on user selections at install time, an installation may not include every artifact that could be created or executed on the endpoint when the software component is installed (i.e. if a particular optional sub-component is not installed, the files associated with that software component may be included in payload, but not installed in the system entity). Described in [Section 2.7.3](#).
- o evidence-entry (label 3): This item is used to provide results from a scan of a system where software that does not have a CoSWID tag is discovered. In such a case, a CoSWID tag may be created by the discovery process when the endpoint is scanned. This item represents evidence for why software is believed to be installed on the endpoint. Described in [Section 2.7.4](#).
- o any-element-entry (label 7): A default map that can contain arbitrary map members and even nested maps (which would also be any-elements). In essence, the any-element allows items not defined in this CDDL data definition to be included in a Concise Software Identifier. Described in [model-any-element].

2.1.1. Determining the tag type

The operational model for SWID and CoSWID tags was introduced in [Section 1.1](#). The following rules can be used to determine the type of a CoSWID tag.

- o Corpus Tag: A CoSWID tag MUST be considered a corpus tag if the corpus item is "true".
- o Primary Tag: A CoSWID tag MUST be considered a primary tag if the corpus, patch, and supplemental items are "false".
- o Patch Tag: A CoSWID tag MUST be considered a patch tag if the patch item is "true" and the corpus item is "false".
- o Supplemental Tag: A CoSWID tag MUST be considered a supplemental tag if the supplemental item is set to "true".

A tag that does not match one of the above rules MUST be considered an invalid, unsupported tag type.

If a patch modifies the version number or the descriptive metadata of the software, then a new tag representing these details SHOULD be installed, and the old tag SHOULD be removed.

2.1.2. concise-software-identity Co-constraints

- o Only one of the patch and supplemental items MUST be set to "true", or the patch and supplemental items MUST be set to "false" or be omitted.
- o If the patch item is set to "true", the tag SHOULD contain at least one link with the rel(ation) item value of "patches" and an href item specifying an association with the software that was patched.
- o If the supplemental item is set to "true", the tag SHOULD contain at least one link with the rel(ation) item value of "supplements" and an href item specifying an association with the software that is supplemented.
- o If all of the corpus, patch, and supplemental items are "false", or if the corpus item is set to "true", then a software-version item MUST be included with a value set to the version of the software component. This ensures that primary and corpus tags have an identifiable software version.

2.2. 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 is as follows:

```
global-attributes = (  
  ? lang,  
  * any-attribute,  
)  
  
label = text / int  
  
any-attribute = (  
  label => text / int / [ 2* text ] / [ 2* int ]  
)
```

The following describes each child item of this object.

- o lang (index 15): A textual language tag that conforms with IANA Language Subtag Registry [[RFC5646](#)].
- o any-attribute: This sub-group provides a means to include arbitrary information via label (key) item value pairs where both keys and values can be either a single integer or text string, or an array of integers or text strings.

2.3. The media Object

The CDDL for the entity object is as follows:


```

media = (10: [ + [ media-expression,
                  ? [ media-operation,
                      media-expression,
                    ]
                ]
        ])
media-operation = text
media-expression = media-environment / [ media-prefix,
                                         media-environment,
                                         media-attribute,
                                         media-value,
                                       ]

media-prefix = text
media-environment = text
media-attribute = text
media-value = text

```

The following describes each child item of this object.

- o TBD

2.4. The entity Object

The CDDL for the entity object is as follows:

```

entity = {
  global-attributes,
  entity-name,
  ? reg-id,
  role,
  ? thumbprint,
  * $$entity-extension,
}

```

The following describes each child item of this object.

- o global-attributes: The global-attributes group described in [Section 2.2](#).
- o entity-name (index 32): The text-string name of the organization claiming a particular role in the CoSWID tag.
- o reg-id (index 32): The registration id is intended to uniquely identify a naming authority in a given scope (e.g. global, organization, vendor, customer, administrative domain, etc.) for the referenced naming authority. The value of an registration ID MUST be a [RFC 3986](#) URI. The scope SHOULD be the scope of an

organization. In a given scope, the registration id MUST be used consistently.

- o role (index 33): The relationship(s) between this organization and this tag. The role of an entity MAY include any role value, but the pre-defined roles include: "aggregator", "distributor", "licensor", "software-creator", and "tag-creator". These pre-defined role index and text values are defined in [Section 3.2](#). Use of index values instead of text for these pre-defined roles allows a CoSWID to be more concise.

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

- 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 entities certificate(s). 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 2.7.1](#) for more details on the use of the hash-entry data structure.
- o \$\$entity-extension: This CDDL socket (see [[I-D.ietf-cbor-cddl](#)] [section 3.9](#)) can be used to extend the entity model, allowing well-formed extensions to be defined in additional CDDL descriptions.

[2.5.](#) The link Object

The CDDL for the link object is as follows:

```
link = {  
  global-attributes,  
  ? artifact,  
  href,  
  ? media  
  ? ownership,  
  rel,  
  ? media-type,  
  ? use,  
}
```

The following describes each child item of this object.

- o global-attributes: The global-attributes group described in [Section 2.2](#).
- o artifact (index: 37): For installation media (rel="installation-media"), this item value indicates the path of the installer executable or script that can be run to launch the referenced installation. Items with the same artifact name should be considered mirrors of each other, allowing the installation media to be downloaded from any of the described sources.
- o href (index 38): The link to the item being referenced. The "href" item's value can point to several different things, and can be any of the following:
 - * If no URI scheme is provided, then the URI is to be interpreted as being relative to the URI of the CoSWID tag. For example, `./folder/supplemental.coswid`.
 - * a physical resource location with any acceptable URI scheme (e.g., `file://` `http://` `https://` `ftp://`)
 - * a URI with "coswid:" as the scheme, which refers to another CoSWID by tag-id. This URI would need to be resolved in the context of the endpoint by software that can lookup other CoSWID tags. For example, `coswid:2df9de35-0aff-4a86-ace6-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 CoSWID via an XPATH query. This URI would need to be resolved in the context of the system entity via software components that can lookup other CoSWID tags and select the appropriate tag based on an XPATH query. Examples include:
 - * `swidpath://SoftwareIdentity[Entity/@regid='http://contoso.com']` would retrieve all CoSWID tags that include an entity where the regid is "Contoso" or `swidpath://SoftwareIdentity[Meta/@persistentId='b0c55172-38e9-4e36-be86-92206ad8eddb']` would match CoSWID tags with the persistent-id value `"b0c55172-38e9-4e36-be86-92206ad8eddb"`.
 - * See XPATH query standard : <http://www.w3.org/TR/xpath20/>
- o media (index 10): See media defined in [Section 2.3](#).
- o ownership (index 39): Determines the relative strength of ownership of the software components. Valid enumerations are: abandon, private, shared

- o rel (index 40): The relationship between this CoSWID and the target file. Relationships can be identified by referencing the IANA registration library: <https://www.iana.org/assignments/link-relations/link-relations.xhtml>.
- o media-type (index 41): The IANA MediaType for the target resource; this provides the consumer with a hint of what type of resource to expect. See <http://www.iana.org/assignments/media-types/media-types.xhtml> for more details.
- o use (index 42): Determines if the target software is a hard requirement or not to be installed before installing the tagged software component. Valid enumerations are: required, recommended, optional, which are defined in [Section 3.3](#).

2.6. The software-meta Object

The CDDL for the software-meta object is as follows:

```
software-meta = {  
    global-attributes,  
    ? activation-status,  
    ? channel-type,  
    ? colloquial-version,  
    ? description,  
    ? edition,  
    ? entitlement-data-required,  
    ? entitlement-key,  
    ? generator,  
    ? persistent-id,  
    ? product,  
    ? product-family,  
    ? revision,  
    ? summary,  
    ? unspsc-code,  
    ? unspsc-version,  
}
```

The following describes each child item of this object.

- o global-attributes: The global-attributes group described in [Section 2.2](#).
- o activation-status (index 43): Identification of the activation status of this software title (e.g. Trial, Serialized, Licensed, Unlicensed, etc). Typically, this is used in supplemental tags.

- o channel-type (index 44): Provides information on which channel this particular software was targeted for (e.g. Volume, Retail, OEM, Academic, etc). Typically used in supplemental tags.
- o colloquial-version (index 45): The informal or colloquial version of the product (i.e. 2013). Note that this version may be the same through multiple releases of a software component where the version specified in entity is much more specific and will change for each software release.

Note that this representation of version is typically used to identify a group of specific software releases that are part of the same release/support infrastructure (i.e. Fabrikam Office 2013). This version is used for string comparisons only and is not compared to be an earlier or later release (that is done via the entity version).

- o description (index 46): A longer, detailed description of the software. This description can be multiple sentences (differentiated from summary, which is a very short, one-sentence description).
- o edition (index 47): The variation of the product (Extended, Enterprise, Professional, Standard, etc).
- o entitlement-data-required (index 48): An indicator to determine if there should be accompanying proof of entitlement when a software license reconciliation is completed.
- o entitlement-key (index 49): A vendor-specific textual key that can be used to reconcile the validity of an entitlement. (e.g. serial number, product or license key).
- o generator (index 50): The name of the software tool that created a CoSWID tag. This item is typically used if tags are created on the fly or via a catalog-based analysis for data found on an endpoint.
- o persistent-id (index 51): A GUID used to represent products installed where the products are related, but may be different versions.
- o product (index 52): The base name of the product.
- o product-family (index 53): The overall product family this software belongs to. Product family is not used to identify that a product is part of a suite, but is instead used when a set of products that are all related may be installed on multiple

different endpoints. For example, an enterprise backup system may consist of a backup services, multiple different backup services that support mail services, databases and ERP systems, as well as individual software components that backup client system entities. In such an usage scenario, all software components that are part of the backup system would have the same product-family name so they can be grouped together in respect to reporting systems.

- o revision (index 54): The informal or colloquial representation of the sub-version of the given product (ie, SP1, R2, RC1, Beta 2, etc). Note that the version item will provide very exact version details, while the revision is intended for use in environments where reporting on the informal or colloquial representation of the software is important (for example, if for a certain business process, an organization recognizes that it must have, for example "ServicePack 1" or later of a specific product installed on all devices, they can use the revision data value to quickly identify any devices that do not meet this requirement).

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

- o summary (index 55): A short (one-sentence) description of the software.
- o unspsc-code (index 56): An 8 digit code that provides UNSPSC classification of the software component this SWID tag identifies. For more information see, <http://www.unspsc.org/>.
- o unspsc-version (index 57): The version of the UNSPSC code used to define the UNSPSC code value. For more information see, <http://www.unspsc.org/>.

2.7. The Resource Collection Definition

2.7.1. The hash-entry Array

CoSWID add explicit support for the representation of hash entries using algorithms that are registered at the Named Information Hash Algorithm Registry via the hash-entry member (label 58).

```
hash-entry = (58: [ hash-alg-id: int, hash-value: bstr ] )
```

The number used as a value for hash-alg-id MUST refer an ID in the Named Information Hash Algorithm Registry; 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.

[2.7.2.](#) The resource-collection Group

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

The CDDL for the resource-collection group is as follows:

```
resource-collection = (  
    ? directory-entry,  
    ? file-entry,  
    ? process-entry,  
    ? resource-entry  
)
```

```
directory = {  
    filesystem-item,  
    path-elements,  
}
```

```
file = {  
    filesystem-item,  
    ? size,  
    ? file-version,  
    ? hash-entry,  
}
```

```
process = {  
    global-attributes,  
    process-name,  
    ? pid,  
}
```

```
resource = {  
    global-attributes,  
    type,  
}
```

```
filesystem-item = (  
    global-attributes,  
    ? key,  
    ? location,  
    fs-name,  
    ? root,  
)
```

The following describes each child item or group for these groups.

- o filesystem-item: A list of items both used in representing the nodes of a file-system hierarchy, i.e. directory items that allow one or more directories to be defined in the file structure, and file items that allow one or more files to be specified for a given location.
- o global-attributes: The global-attributes group described in [Section 2.2](#).

- o directory-entry (index 16): A directory item allows one or more directories to be defined in the file structure.
- o file-entry (index 17): A file element that allows one or more files to be specified for a given location.
- o process-entry (index 18): Provides process (software component in execution) information for data that will show up in a devices process table.
- o resource-entry (index 19): A set of items that can be used to provide arbitrary resource information about an application installed on a system entity, or evidence collected from a system entity.
- o size (index 20): The file size in bytes of the file.
- o file-version (index 21): The version of the file.
- o key (index 22): Files that are considered important or required for the use of a software component. Typical key files would be those which, if not available on a system entity, would cause the software component not to execute or function properly. Key files will typically be used to validate that a software component referenced by the CoSWID tag document is actually installed on a specific system entity.
- o location (index 23): The directory or location where a file was found or can expected to be located. This text-string is intended to include the filename itself. This SHOULD be the relative path from the location represented by the root item.
- o fs-name (index 24): The file name or directory name without any path characters.
- o root (index 25): A system-specific root folder that the location item is an offset from. If this is not specified the assumption is the root is the same folder as the location of the CoSWID tag. The text-string value represents a path expression relative to the CoSWID tag document location in the (composite) file-system hierarchy.
- o path-elements (index 26): Provides the ability to apply a directory structure to the path expressions for files defined in a payload or evidence item.
- o process-name (index 27): The process name as it will be found in the system entity's process table.

- o pid (index 28): The process ID for the process in execution that can be included in the process item as part of an evidence tag.
- o type (index 29): The type of resource represented via a text-string (typically, registry-key, port or root-uri).

2.7.3. The payload Object

The CDDL for the payload object is as follows:

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

The following describes each child item of this object.

- o global-attributes: The global-attributes group described in [Section 2.2](#).
- o resource-collection: The resource-collection group described in [Section 2.7.2](#).
- o \$\$payload-extension: This CDDL socket (see [[I-D.ietf-cbor-cddl](#)] [section 3.9](#)) can be used to extend the payload model, allowing well-formed extensions to be defined in additional CDDL descriptions.

2.7.4. The evidence Object

The CDDL for the evidence object is as follows:

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

The following describes each child item of this object.

- o global-attributes: The global-attributes group described in [Section 2.2](#).

- o resource-collection: The resource-collection group described in [Section 2.7.2](#).
- o date (index 35): The date and time evidence represented by an evidence item was gathered.
- o device-id (index 36): A text-string identifier for a device evidence was gathered from.
- o \$\$evidence-extension: This CDDL socket (see [[I-D.ietf-cbor-cddl](#)] [section 3.9](#)) can be used to extend the evidence model, allowing well-formed extensions to be defined in additional CDDL descriptions.

2.8. Full CDDL Definition

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

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



```
)

resource-collection = (
    ? directory-entry,
    ? file-entry,
    ? process-entry,
    ? resource-entry
)

file = {
    filesystem-item,
    ? size,
    ? file-version,
    ? hash-entry,
}

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

directory = {
    filesystem-item,
    path-elements,
}

process = {
    global-attributes,
    process-name,
    ? pid,
}

resource = {
    global-attributes,
    type,
}

entity = {
    global-attributes,
    entity-name,
    ? reg-id,
    role,
    ? thumbprint,
    * $$entity-extension,
}
```



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

```
link = {  
  global-attributes,  
  ? artifact,  
  href,  
  ? media  
  ? ownership,  
  rel,  
  ? media-type,  
  ? use,  
}
```

```
software-meta = {  
  global-attributes,  
  ? activation-status,  
  ? channel-type,  
  ? colloquial-version,  
  ? description,  
  ? edition,  
  ? entitlement-data-required,  
  ? entitlement-key,  
  ? generator,  
  ? persistent-id,  
  ? product,  
  ? product-family,  
  ? revision,  
  ? summary,  
  ? unspsc-code,  
  ? unspsc-version,  
}
```

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

```
tag-id = (0: text)  
swid-name = (1: text)  
entity-entry = (2: entity / [ 2* entity ])  
evidence-entry = (3: evidence)
```



```
link-entry = (4: link / [ 2* link ])
software-meta-entry = (5: software-meta / [ 2* software-meta ])
payload-entry = (6: payload)
corpus = (8: bool)
patch = (9: bool)
media = (10: [ + [ media-expression,
                  ? [ media-operation,
                      media-expression,
                      ]
                  ]
        ])
media-operation = text
media-expression = media-environment / [ media-prefix,
                                         media-environment,
                                         media-attribute,
                                         media-value,
                                         ]

media-prefix = text
media-environment = text
media-attribute = text
media-value = text
supplemental = (11: bool)
tag-version = (12: integer)
software-version = (13: text)
version-scheme = (14: version-schemes / extended-value)
version-schemes = multipartnumeric / multipartnumeric-suffix / alphanumeric /
decimal / semver
multipartnumeric = 1
multipartnumeric-suffix = 2
alphanumeric = 3
decimal = 4
semver = 16384
lang = (15: text)
directory-entry = (16: directory / [ 2* directory ])
file-entry = (17: file / [ 2* file ])
process-entry = (18: process / [ 2* process ])
resource-entry = (19: resource / [ 2* resource ])
size = (20: integer)
file-version = (21: text)
key = (22: bool)
location = (23: text)
fs-name = (24: text)
root = (25: text)
path-elements = (26: { * file-entry,
                      * directory-entry,
                      }
                )
process-name = (27: text)
```

```
pid = (28: integer)
```

Birkholz, et al.

Expires May 12, 2019

[Page 27]

```
type = (29: text)
entity-name = (31: text)
reg-id = (32: any-uri)
role = (33: roles / extended-value / [ 2* roles / extended-value ] )
extended-value = text / uint
roles= aggregator / distributor / licensor / software-creator / tag-creator
aggregator=0
distributor=1
licensor=2
software-creator=3
tag-creator=4
thumbprint = (34: [ hash-alg-id: int,
                    hash-value: bstr,
                    ]
              )
date = (35: time)
device-id = (36: text)
artifact = (37: text)
href = (38: any-uri)
ownership = (39: shared / private / abandon / extended-value )
shared=0
private=1
abandon=2
rel = (40: rels / extended-value )
rels = ancestor / component / feature / installationmedia / packageinstaller /
parent / patches / requires / see-also / supersedes / rel-supplemental
ancestor=0
component=1
feature=2
installationmedia=3
packageinstaller=4
parent=5
patches=6
requires=7
see-also=8
supersedes=9
rel-supplemental=10
media-type = (41: text)
use = (42: optional / required / recommended / extended-value )
optional=1
required=2
recommended=3
activation-status = (43: text)
channel-type = (44: text)
colloquial-version = (45: text)
description = (46: text)
edition = (47: text)
entitlement-data-required = (48: bool)
```

entitlement-key = (49: text)

Birkholz, et al.

Expires May 12, 2019

[Page 28]

```

generator = (50: text)
persistent-id = (51: text)
product = (52: text)
product-family = (53: text)
revision = (54: text)
summary = (55: text)
unspsc-code = (56: text)
unspsc-version = (57: text)
hash-entry = (58: [ hash-alg-id: int,
                    hash-value: bstr,
                    ]
              )

```

3. CoSWID Indexed Label Values

3.1. Version Scheme

The following table contains an initial set of values for use in the version-scheme item for the version schemes defined in the ISO/IEC 19770-2:2015 [\[SWID\]](#) specification. Index value in parens indicates the index value to use in the version-scheme item.

Index	Role Name	Definition
0	Reserved	
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 string 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

The values above are registered in the "SWID/CoSWID Version Schema Values" registry defined in section [Section 4.1](#). Additional valid values will likely be registered over time in this registry. Additionally, the index values 32768 through 65535 have been reserved for private use.

3.2. Entity Role Values

The following table indicates the index value to use for the entity roles defined in the ISO/IEC 19770-2:2015 [[SWID](#)] specification.

Index	Role Name	Definition
0	Reserved	
1	tagCreator	The person or organization that created the containing SWID or CoSWID tag
2	softwareCreator	From [SAM], "person or organization that creates a software product (3.46) or package"
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", "person or organization who owns or holds the rights to issue a software license for a specific software package"

The values above are registered in the "SWID/CoSWID Entity Role Values" registry defined in section [Section 4.2](#). Additional valid values will likely be registered over time. Additionally, the index values 128 through 255 have been reserved for private use.

3.3. Use Values

The following table indicates the index value to use for the link use item (see [Section 2.5](#)), which is also defined in the ISO/IEC 19770-2:2015 [[SWID](#)] specification.

Index	Use Type	Definition
0	Reserved	
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."

The values above are registered in the "SWID/CoSWID Link Use Values" registry defined in section [Section 4.3](#). Additional valid values will likely be registered over time. Additionally, the index values 128 through 255 have been reserved for private use.

4. IANA Considerations

This document will include requests to IANA:

- o Integer indices for SWID content attributes and information elements.
- o Content-Type for CoAP to be used in COSE.

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

4.1. SWID/CoSWID Version Schema Values Registry

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

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

Range	Registration Procedures
0-16383	Standards Action
16384-32767	Specification Required
32768-65535	Reserved for Private Use

Initial registrations for the SWID/CoSWID Version Schema Values registry are provided below.

Index	Role Name	Specification
0	Reserved	
1	multipartnumeric	See Section 3.1
2	multipartnumeric+suffix	See Section 3.1
3	alphanumeric	See Section 3.1
4	decimal	See Section 3.1
5-16383	Unassigned	
16384	semver	[SEMVER]
16385-32767	Unassigned	
32768-65535	Reserved for Private Use	

[4.2.](#) SWID/CoSWID Entity Role Values Registry

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

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

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

Initial registrations for the SWID/CoSWID Entity Role Values registry are provided below.

Index	Role Name	Specification
0	Reserved	
1	tagCreator	See Section 3.2
2	softwareCreator	See Section 3.2
3	aggregator	See Section 3.2
4	distributor	See Section 3.2
5	licensor	See Section 3.2
6-31	Unassigned	
32-127	Unassigned	
128-255	Reserved for Private Use	

4.3. SWID/CoSWID Link Use Values Registry

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

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

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

Initial registrations for the SWID/CoSWID Entity Role Values registry are provided below.

Index	Role Name	Specification
0	Reserved	
1	optional	See Section 3.3
2	required	See Section 3.3
3	recommended	See Section 3.3
4-31	Unassigned	
32-127	Unassigned	
128-255	Reserved for Private Use	

[4.4.](#) Media Type Registration

[4.4.1.](#) swid+cbor Media Type Registration

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 5](#) of RFC-AAAA.

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

Published specification: RFC-AAAA

Applications that use this media type: The type is used by Software asset management systems, Vulnerability assessment systems, and in applications that use remote integrity verification.

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

Additional information:

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

File extension(s): coswid

Macintosh file type code(s): none

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

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

Intended usage: COMMON

Restrictions on usage: None

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

Change controller: IESG

4.5. 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	-	TBDcf	RFC-AAAA

Table 1: CoAP Content-Format IDs

4.6. 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]

5. Security Considerations

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

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

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

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

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

6. Acknowledgments

7. Change Log

Changes from version 06 to version 07:

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

Changes from version 05 to version 06:

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

Changes from version 04 to version 05:

- o Clarified language around SWID and CoSWID to make more consistent use of these terms.
- o Added language describing CBOR optimizations for single vs. arrays in the model front matter.

- o Fixed a number of grammatical, spelling, and wording issues.
- o Documented extension points that use CDDL sockets.
- o Converted IANA registration tables to markdown tables, reserving the 0 value for use when a value is not known.
- o Updated a number of references to their current versions.

Changes from version 03 to version 04:

- o 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

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 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 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

8. Contributors

9. References

9.1. Normative References

- [I-D.ietf-ace-cbor-web-token]
Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig,
"CBOR Web Token (CWT)", [draft-ietf-ace-cbor-web-token-15](#)
(work in progress), March 2018.
- [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>>.
- [RFC4108] Housley, R., "Using Cryptographic Message Syntax (CMS) to
Protect Firmware Packages", [RFC 4108](#),
DOI 10.17487/RFC4108, August 2005,
<<https://www.rfc-editor.org/info/rfc4108>>.
- [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>>.
- [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>>.
- [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", n.d., <<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.
- [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>>.
- [X.1520] "Recommendation ITU-T X.1520 (2014), Common vulnerabilities and exposures", April 2011.

9.2. Informative References

- [I-D.birkholz-tuda]
Fuchs, A., Birkholz, H., McDonald, I., and C. Bormann, "Time-Based Uni-Directional Attestation", [draft-birkholz-tuda-04](#) (work in progress), March 2017.
- [I-D.ietf-cbor-cddl]
Birkholz, H., Vigano, C., and C. Bormann, "Concise data definition language (CDDL): a notational convention to express CBOR and JSON data structures", [draft-ietf-cbor-cddl-05](#) (work in progress), August 2018.
- [I-D.ietf-sacm-rolie-softwaredescriptor]
Banghart, S. and D. Waltermire, "Definition of the ROLIE Software Descriptor Extension", [draft-ietf-sacm-rolie-softwaredescriptor-03](#) (work in progress), July 2018.
- [I-D.ietf-sacm-terminology]
Birkholz, H., Lu, J., Strassner, J., Cam-Winget, N., and A. Montville, "Security Automation and Continuous Monitoring (SACM) Terminology", [draft-ietf-sacm-terminology-15](#) (work in progress), June 2018.
- [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>>.
- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, [RFC 4949](#), DOI 10.17487/RFC4949, August 2007, <<https://www.rfc-editor.org/info/rfc4949>>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", [RFC 7228](#), DOI 10.17487/RFC7228, May 2014, <<https://www.rfc-editor.org/info/rfc7228>>.

Appendix A. CoSWID Attributes for Firmware (label 60)

NOTE: this appendix is subject to revision or removal based on potential convergence of:

- o [draft-moran-suit-manifest](#), and
- o [draft-birkholz-suit-coswid-manifest](#)

The ISO-19770-2:2015 specification of SWID tags assumes the existence of a file system a software component is installed and stored in. In the case of constrained-node networks [[RFC7228](#)] or network equipment this assumption might not apply. Concise software instances in the form of (modular) firmware are often stored directly on a block device that is a hardware component of the constrained-node or network equipment. Multiple differentiable block devices or segmented block devices that contain parts of modular firmware components (potentially each with their own instance version) are already common at the time of this writing.

The optional attributes that annotate a firmware package address specific characteristics of pieces of firmware stored directly on a block-device in contrast to software deployed in a file-system. In essence, trees of relative path-elements expressed by the directory and file structure in CoSWID tags are typically unable to represent the location of a firmware on a constrained-node (small thing). The composite nature of firmware and also the actual composition of small things require a set of attributes to address the identification of the correct component in a composite thing for each individual piece of firmware. A single component also potentially requires a number of distinct firmware parts that might depend on each other (versions). These dependencies can be limited to the scope of the component itself or extend to the scope of a larger composite device. In addition, it might not be possible (or feasible) to store a CoSWID tag document (permanently) on a small thing along with the corresponding piece of firmware.

To address the specific characteristics of firmware, the extension points "\$\$payload-extension" and "\$\$evidence-extension" are used to allow for an additional type of resource description--firmware-entry--thereby increasing the self-descriptiveness and flexibility of CoSWID. The optional use of the extension points "\$\$payload-

extension" and "\$\$evidence-extension" in respect to firmware MUST adhere to the following CDDL data definition.

<CODE BEGINS>

```
$$payload-extension  //= (suit.manifest-entry,)
$$evidence-extension  //= (suit.manifest-entry,)

suit-manifest = {
    suit.manifest-version,
    suit.digest-info,
    suit.text-reference,
    suit.nonce,
    suit.sequence-number,
    ? suit.pre-condition,
    ? suit.post-condition,
    ? suit.directives,
    ? suit.resources,
    ? suit.processors,
    ? suit.targets,
    ? suit.extensions,
}

suit.manifest-entry = (59: suit-manifest / [ 2* suit-manifest ] )
suit.manifest-version = (60: 1)
suit.digest-info = (61: [ suit.digest-algorithm,
                        ? suit.digest-parameters,
                        ]
                    )
suit.digest-algorithm = uint
suit.digest-parameters = bytes
suit.text-reference = (62: bytes)
suit.nonce = (63: bytes)
suit.sequence-number = (64: uint)
suit.pre-condition = (suit.id-condition // suit.time-condition // suit.image-
condition // suit.custom-condition)
suit.post-condition = (suit.image-condition // suit.custom-condition)
suit.id-condition = (65: [ + [ suit.vendor / suit.class / suit.device,
                            suit.uuid,
                            ]
                        ]
                    )

suit.vendor = 0
suit.class = 1
suit.device = 2
suit.uuid = bstr .size 16
suit.time-condition = (66: [ + [ suit.install-after / suit.best-before,
                            suit.timestamp,
                            ]
                        ]
                    )
```

]

Birkholz, et al.

Expires May 12, 2019

[Page 44]

```

    )
    suit.install-after = 0
    suit.best-before = 1
    suit.timestamp = uint .size 8
    suit.image-condition = (67: [ + [ suit.current-content / suit.not-current-
content,
                                suit.storage-identifier,
                                ? suit.digest,
                                ]
                                ]
    )
    suit.current-content = 0
    suit.not-current-content = 1
    suit.digest = bytes
    suit.storage-identifier = bytes
    suit.custom-condition = (68: [ nint,
                                suit.condition-parameters,
                                ]
    )
    suit.condition-parameters = bytes
    suit.directives = (69: { + int => bytes } )
    suit.resources = (70: [ + [ suit.resource-type,
                                suit.uri-list,
                                suit.digest,
                                suit.onode,
                                ? suit.size,
                                ]
                                ]
    )
    suit.resource-type = suit.payload / suit.dependency / suit.key / suit.alias
    suit.payload = 0
    suit.dependency = 1
    suit.key = 2
    suit.alias = 3
    suit.uri-list = { + int => text }
    suit.size = uint
    suit.onode = bytes
    suit.processors = (71: [ + [ suit.decrypt / suit.decompress / suit.undiff /
suit.relocate / suit.unrelocate,
                                suit.parameters,
                                suit.inode,
                                suit.onode,
                                ]
                                ]
    )
    suit.decrypt = 0
    suit.decompress = 1
    suit.undiff = 2

```

```
suit.relocate = 3  
suit.unrelocate = 4
```

Birkholz, et al.

Expires May 12, 2019

[Page 45]


```
suit.parameters = bytes
suit.inode = bytes
suit.targets = (72: [ + [ suit.component-id,
                        suit.storage-identifier,
                        suit.inode,
                        ? suit.encoding,
                        ]
                ]
              )
suit.component-id = [ + bytes ]
suit.encoding = bytes
suit.extensions = (73: { + int => bytes } )
<CODE ENDS>
```

The members of the firmware group that constitutes the content of the firmware-entry is based on the metadata about firmware Described in [\[RFC4108\]](#). As with every semantic differentiation that is supported by the resource-collection type, the use of firmware-entry is optional. It is REQUIRED not to instantiate more than one firmware-entry, as the firmware group is used in a map and therefore only allows for unique labels.

The optional cms-firmware-package member allows to include the actual firmware in the CoSWID tag that also expresses its metadata as a byte-string. This option enables a CoSWID tag to be used as a container or wrapper that composes both firmware and its metadata in a single document (which again can be signed, encrypted and/or compressed). In consequence, a CoSWID tag about firmware can be conveyed as an identifying document across endpoints or used as a reference integrity measurement as usual. Alternatively, it can also convey an actual piece of firmware, serve its intended purpose as a SWID tag and then - due to the lack of a location to store it - be discarded.

[Appendix B](#). 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.

<CODE BEGINS>

signed-coswid = #6.997(COSE-Sign1-coswid) ; see TBS7 in current COSE I-D

label = int / tstr ; see COSE I-D 1.4.

values = any ; see COSE I-D 1.4.

unprotected-signed-coswid-header = {

1 => int, ; algorithm identifier

3 => "application/coswid", ; request for CoAP IANA registry to become an

int

* label => values,

}

protected-signed-coswid-header = {

4 => bstr, ; key identifier

* label => values,

}

COSE-Sign1-coswid = [

protected: bstr .cbor protected-signed-coswid-header,

unprotected: unprotected-signed-coswid-header,

payload: bstr .cbor concise-software-identity,

signature: bstr,

]

<CODE ENDS>

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

