```
SACM                                                D. Waltermire, Ed.
Internet-Draft                                                    NIST
Intended status: Standards Track                             K. Watson
Expires: December 10, 2016                                         DHS
                                                               C. Kahn
                                                           L. Lorenzin
                                                     Pulse Secure, LLC
                                                              M. Cokus
                                                             D. Haynes
                                                The MITRE Corporation
                                                          June 8, 2016
```

SACM Information Model
draft-ietf-sacm-information-model-05

Abstract

This document defines the information elements that are transported between SACM components and their interconnected relationships. The primary purpose of the Secure Automation and Continuous Monitoring (SACM) Information Model is to ensure the interoperability of corresponding SACM data models and addresses the use cases defined by SACM. The information elements and corresponding types are maintained as the IANA "SACM Information Elements" registry.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The SACM Information Model (IM) serves multiple purposes:

- o to ensure interoperability between SACM data models that are used as transport encodings,
- o to provide a standardized set of information elements - the SACM Vocabulary - to enable the exchange of content vital to automated security posture assessment, and
- o to enable secure information sharing in a scalable and extensible fashion in order to support the tasks conducted by SACM components.

A complete set of requirements imposed on the IM can be found in [I-D.ietf-sacm-requirements]. The SACM IM is intended to be used for standardized data exchange between SACM components (data in motion). Nevertheless, the information elements (IE) and their relationships defined in this document can be leveraged to create and align corresponding data models for data at rest.

The information model expresses, for example, target endpoint (TE) attributes, guidance, and evaluation results. The corresponding information elements are consumed and produced by SACM components as they carry out tasks.

The primary tasks that this information model supports (on data, control, and management plane) are:

- o TE Discovery

- TE Characterization
- TE Classification
- Collection
- Evaluation
- Information Sharing
- SACM Component Discovery
- SACM Component Authentication
- SACM Component Authorization
- SACM Component Registration

These tasks are defined in [I-D.ietf-sacm-terminology].

2. Conventions used in this document

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2.2. Information Element Examples

The notation used to define the SACM Information Elements (IEs) is based on a customized version of the IPFIX information model syntax [RFC7012] which is described in Section 4.1 and Section 4.2. However, there are several examples presented throughout the document that use a simplified pseudo-code to illustrate the basic structure. It should be noted that while they include actual names of subjects and attributes as well as values, they are not intended to influence

how corresponding SACM IEs should be defined in Section 6. The examples are provided for demonstration purposes only.

3. Information Elements

The IEs defined in this document comprise the building blocks by which all SACM content is composed. They are consumed and provided by SACM components on the data plane. Every information element has a unique label: its name. Every type of IE defined by the SACM IM is registered as a type at the IANA registry. The Integer Index of the IANA SMI number tables can be used by SACM data models.

3.1. Context of Information Elements

The IEs in this information model represent information related to the following areas (based on the use cases described in [RFC7632]):

- Endpoint Management
- Software Inventory Management
- Hardware Inventory Management
- Configuration Management
- Vulnerability Management

3.2. Extensibility of Information Elements

A SACM data model based on this information model MAY include additional information elements that are not defined here. The labels of additional information elements included in different SACM data models MUST NOT conflict with the labels of the information elements defined by this information model, and the names of additional information elements MUST NOT conflict with each other or across multiple data models. In order to avoid naming conflicts, the labels of additional IEs SHOULD be prefixed to avoid collisions across extensions. The prefix MUST include an organizational identifier and therefore, for example, MAY be an IANA enterprise number, a (partial) name space URI, or an organization name abbreviation.

4. Structure of Information Elements

There are two basic types of IEs:

- o Attributes: an instance of an attribute type is the simplest IE structure comprised of a unique attribute name and an attribute value.

- o Subjects: a subject is a richer structure that has a unique subject name and one or more attributes or subjects. In essence, instances of a subject type are defined (and differentiated) by the attribute values and subjects associated with it.

```
hostname = "arbutus"

coordinates = (
latitude = N27.99619,
longitude = E86.92761
)
```

Figure 1: Example instance of an attribute and subject.

In general, every piece of information that enables security posture assessment or further enriches the quality of the assessment process can be associated with metadata. In the SACM IM, metadata is represented by specific subjects and is bundled with other attributes or subjects to provide additional information about them. The IM explicitly defines two kinds of metadata:

- o Metadata focusing on the data origin (the SACM component that provides the information to the SACM domain)

- o Metadata focusing on the data source (the target endpoint that is assessed)

Metadata can also include relationships that refer to other associated IEs (or SACM content in general) by using referencing labels that have to be included in the metadata of the associated IE.

Subjects can be nested and the SACM IM allows for circular or recursive nesting. The association of IEs via nesting results in a tree-like structure wherein subjects compose the root and intermediary nodes and attributes the leaves of the tree. This semantic structure does not impose a specific structure on SACM data models regarding data in motion or data repository schemata for data at rest.

The SACM IM provides two top-level subjects that are used to ensure a homogeneous structure for SACM content and its associated metadata: SACM statements and SACM content-elements. Every set of IEs that is provided by a SACM component in order to be consumed by another SACM component uses these top-level subjects.

The notation the SACM IM is defined in is based on the IP Information Flow Export (IPFIX) Information Model syntax described in [RFC7012]. The customized syntax used by the SACM IM is defined below in

Section 4.1 and Section 4.2.

4.1. Attribute Syntax

Attributes must be defined using the IPFIX Information Element Specification Template as described in Section 2.1 of [RFC7012]. The following is a modified version of the template for Information Elements provided in Section 9.1 of [RFC7013].

```
elementId:          Element identifier goes here
                    if known, or "TBD" if it will
                    be assigned by IANA and filled
                    in at publication time.;
                    obligatory

name:               Name goes here.; obligatory

dataType:           Data type goes here;
                    obligatory

status:             Status goes here; obligatory

description:        Description goes here.;
                    obligatory

                    For Information Elements that
                    represent flags, please
                    include a table that lists
                    each flag value (hexadecimal)
                    and description. The following
                    is a template for that table.

+-------+----------------------------------+
| Value | Description                      |
+-------+----------------------------------+
|       |                                  |
+-------+----------------------------------+
dataTypeSemantics: Data type semantics, if any,
                    go here; optional

units:             Units, if any, go here;
                    optional

range:             Range, if not implied by the
                    data type, goes here; optional

references:        References to other RFCs or
                    documents outside the IETF,
                    in which additional
                    information is given, or which
                    are referenced by the
                    description, go here; optional
```

      Figure 2: Attribute Information Element Specification Template

4.2. Subject Syntax

Subjects are complex Information Elements that can be created using one of the following IPFIX constructs that are defined in Section 3.1 of [RFC7012]. ***TODO: UPDATE WITH REVISED IPFIX-LIKE CONSTRUCTS***

o basicList: a list of zero or more instances of any Information Element

o subTemplateList: a list of zero or more instances of a single, specific Template

o subTemplateMultiList: a list of zero or more instances of any Template

The following is a modified version of the template for Information Elements provided in Section 9.1 of [RFC7013] that can be used to model subjects.

```
                  elementId:         Element identifier goes here
                                     if known, or "TBD" if it
                                     will be assigned by IANA and
                                     filled in at publication
                                     time.; obligatory

                  name:              Name goes here.; obligatory

                  dataType:          Data type goes here;
                                     obligatory

                  status:            Status goes here; obligatory

                  description:       Description goes here.;
                                     obligatory

                                     Please include a high-level
                                     model diagram that uses
                                     the following format which
                                     is a simplified version
                                     of a high-level diagram
                                     format used in [RFC6313].

                                     <IE-Name> =
                                      (<IE-DataType>,
                                       <IE-DataTypeSemantic>,
                                       <IE-1>,
                                       <IE-2>,
                                       <IE-3>,
                                       ...
                                     )

                  dataTypeSemantics: Data type semantics, if any,
                                     go here; optional

                  units:             Units, if any, go here;
                                     optional

                  range:             Range, if not implied by
                                     the data type, goes
                  here; optional

                  references:        References to other RFCs or
                                     documents outside the IETF,
                                     in which additional
                                     information is given, or
                                     which are referenced by the
                                     description, go here;
                                     optional
```

Figure 3: Subject Information Element Specification Template

4.3. SACM Content Elements

Every piece of information that is provided by a SACM component is always associated with a set of metadata, for example, the timestamp at which this set of information was produced (e.g. by a collection task) or what target endpoint this set of information is about (e.g. the data-source or a target endpoint identifier, respectively). The subject that associates content IE with content-metadata IE is called a content-element. Content metadata can also include relationships that express associations with other content-elements.

```
         content-element = (
           content-metadata = (
             collection-timestamp = 146193322,
             data-source = fb02e551-7101-4e68-8dec-1fde6bd10981
           ),
           hostname = "arbutus",
           coordinates = (
             latitude = N27.99619,
             longitude = E86.92761
           )
         )
```

Figure 4: Example set of IEs associated with a timestamp and a target endpoint label.

4.4. SACM Statements

One or more SACM content elements are bundled in a SACM statement. In contrast to content-metadata, statement-metatdata focuses on the providing SACM component instead of the target endpoint that the content is about. The only content-specific metadata included in the SACM statement is the content-type IE. Therefore, multiple content-elements that share the same statement metadata and are of the same content-type can be included in a single SACM statement. A SACM statement functions similar to an envelope or a header. Its purpose is to enable the tracking of the origin of data inside a SACM domain and more importantly to enable the mitigation of conflicting information that my originate from different SACM components. How a consuming SACM component actually deals with conflicting information is out-of-scope of the SACM IM. Semantically, the term statement implies that the SACM content provided by a SACM component might not be correct in every context, but rather is the result of an best-effort to produce correct information.

```
         sacm-statement = (
           statement-metadata = (
             publish-timestamp = 1461934031,
             data-origin = 24e67957-3d31-4878-8892-da2b35e121c2,
             content-type = observation
           ),
           content-element = (
             content-metadata = (
               collection-timestamp = 146193322,
               data-source = fb02e551-7101-4e68-8dec-1fde6bd10981
             ),
             hostname = "arbutus"
           )
         )
```

Figure 5: Example of a simple SACM statement including a single content-element.

```
          sacm-statement = (
            statement-metadata = (
              publish-timestamp = 1461934031,
              data-origin = 24e67957-3d31-4878-8892-da2b35e121c2
              content-type = observation
            ),
            content-element = (
              content-metadata = (
                collection-timestamp = 146193322,
                data-source = fb02e551-7101-4e68-8dec-1fde6bd10981
              ),
              coordinates = (
                latitude = N27.99619,
                longitude = E86.92761
              )
            )
          )

          sacm-statement = (
            statement-metadata = (
              publish-timestamp = 1461934744,
              data-origin = e42885a1-0270-44e9-bb5c-865cf6bd4800,
              content-type = observation
            ),
            content-element = (
              content-metadata = (
                collection-timestamp = 146193821,
                te-label = fb02e551-7101-4e68-8dec-1fde6bd10981
              ),
              coordinates = (
                latitude = N16.67622,
                longitude = E141.55321
              )
            )
          )
```

Figure 6: Example of conflicting information originating from different SACM components.

4.5. Relationships

An IE can be associated with another IE, e.g. a user-name attribute can be associated with a content-authorization subject. These references are expressed via the relationships subject, which can be included in a corresponding content-metadata subject. The relationships subject includes a list of one or more references. The SACM IM does not enforce a SACM domain to use unique identifiers as references. Therefore, there are at least two ways to reference another content-element:

- o The value of a reference represents a specific content-label that is unique in a SACM domain (and has to be included in the corresponding content-element metadata in order to be referenced), or

- o The reference is a subject that includes an appropriate number of IEs in order to identify the referenced content-element by its actual content.

It is recommended to provide unique identifiers in a SACM domain and the SACM IM provides a corresponding naming-convention as a reference in section FIXME. The alternative highlighted above summarizes a valid approach that does not require unique identifiers and is similar to the approach of referencing target endpoints via identifying attributes included in a characterization record (FIXME REF arch).

```
        content-element = (
          content-metadata = (
            collection-timestamp = 1461934031,
            te-label =
              fb02e551-7101-4e68-8dec-1fde6bd10981
            relationships = (
              associated-with-user-account =
                f3d70ef4-7e18-42af-a894-8955ba87c95d
            )
          ),
          hostname = "arbutus"
        )

        content-element = (
          content-metadata = (
            content-label = f3d70ef4-7e18-42af-a894-8955ba87c95d
          ),
          user-account = (
            username = romeo
            authentication = local
          )
        )
```

Figure 7: Example instance of a content-element subject associated with another subject via its content metadata.

4.6. Event

Event subjects provide a structure to represent the change of IE values that was detected by a collection task at a specific point of time. It is mandatory to include the new values and the collection timestamp in an event subject and it is recommended to include the past values and a collection timestamp that were replaced by the new IE values. Every event can also be associated with a subject-specific event-timestamp and a lastseen-timestamp that might differ from the corresponding collection-timestamps. If these are omitted the collection-timestamp that is included in the content-metadata subject is used instead.

```
        sacm-statement = (
          statement-metadata = (
            publish-timestamp = 1461934031,
            data-origin = 24e67957-3d31-4878-8892-da2b35e121c2,
            content-type = event
          ),
          event = (
            event-attributes = (
              event-name = "host-name change",
              content-element = (
                content-metadata = (
                  collection-timestamp = 146193322,
                  data-source =
                    fb02e551-7101-4e68-8dec-1fde6bd10981,
                  event-component = past-state
                ),
                hostname = "arbutus"
             ),
            content-element = (
              content-metadata = (
                collection-timestamp = 146195723,
                data-source =
                  fb02e551-7101-4e68-8dec-1fde6bd10981,
                event-component = current-state
              ),
              hostname = "lilac"
           )
          )
        )
```

Figure 8: Example of a SACM statement containing an event.

4.7. Categories

Categories are special IEs that enable to refer to multiple types of IE via just one name. Therefore, they are similar to a type-choice. A prominent example of a category is network-address. Network-address is a category that every kind of network address is associated with, e.g. mac-address, ipv4-address, ipv6-address, or typed-network-address. If a subject includes network-address as one of its components, any of the category members are valid to be used in its place.

Another prominent example is EndpointIdentifier. Some IEs can be used to identify (and over time re-recognize) target endpoints - those are associated with the category endpoint-identifier.

4.8. Designation

TODO: In the IETF, there are privacy concerns with respect to endpoint identity and monitoring. As a result, the Endpoint ID Design Team proposes that "endpoint identity" be changed to "endpoint designation". Designation attributes can be used to correlate endpoints, information about endpoints, events, etc. NOTE: Designation attributes are just those that are mandatory-to-implement. In practice, organizations may need to select additional attributes beyond the mandatory-to-implement attributes to successfully identify an endpoint on their network. Operational and privacy concerns will be covered in Operational Considerations and Privacy Considerations sections respectively. A proposal outlining various options for representing designation attributes/objects in the IPFIX syntax is being discussed on the mailing list. See IM issue #39 at https://github.com/sacmwg/draft-ietf-sacm-information-model/issues/39 for more information.

4.9. Privacy

TODO: In the IETF, there are privacy concerns with respect to endpoint identity and monitoring. As a result, it was proposed that a privacy property be included to denote when a information element represents a privacy concern. A proposal outlining various options for representing privacy attributes/objects in the IPFIX syntax is being discussed on the mailing list. See IM issue #39 at https://github.com/sacmwg/draft-ietf-sacm-information-model/issues/39 for more information.

4.10. Type Space

This section describes the abstract data types that can be used for the specification of the SACM Information Elements in Section 6. Section 4.10.1 describes the set of abstract data types. This section used Section 3 of [RFC7012] as a starting point and was modified to address the needs of SACM.

4.10.1. Abstract Data Types

This section describes the set of valid abstract data types of the SACM information model, independent of how they are implemented in a data model. Note that further abstract data types may be specified by future extensions of the SACM information model.

4.10.1.1. unsigned8

The type "unsigned8" represents a non-negative integer value in the range of 0 to 255.

4.10.1.2. unsigned16

The type "unsigned16" represents a non-negative integer value in the range of 0 to 65535.

4.10.1.3. unsigned32

The type "unsigned32" represents a non-negative integer value in the range of 0 to 4294967295.

4.10.1.4. unsigned64

The type "unsigned64" represents a non-negative integer value in the range of 0 to 18446744073709551615.

4.10.1.5. signed8

The type "signed8" represents an integer value in the range of -128 to 127.

4.10.1.6. signed16

The type "signed16" represents an integer value in the range of -32768 to 32767.

4.10.1.7. signed32

The type "signed32" represents an integer value in the range of -2147483648 to 2147483647.

4.10.1.8. signed64

The type "signed64" represents an integer value in the range of -9223372036854775808 to 9223372036854775807.

4.10.1.9. float32

The type "float32" corresponds to an IEEE single-precision 32-bit floating point type as defined in [IEEE.754.1985].

4.10.1.10. float64

The type "float64" corresponds to an IEEE double-precision 64-bit floating point type as defined in [IEEE.754.1985].

4.10.1.11. boolean

The type "boolean" represents a binary value. The only allowed values are "true" and "false".

4.10.1.12. macAddress

The type "macAddress" represents a MAC-48 address as defined in [IEEE.802-3.2012].

4.10.1.13. octetArray

The type "octetArray" represents a finite-length string of octets.

4.10.1.14. string

The type "string" represents a finite-length string of valid characters from the Unicode coded character set [ISO.10646]. Unicode incorporates ASCII [RFC20] and the characters of many other international character sets.

4.10.1.15. dateTimeSeconds

The type "dateTimeSeconds" represents a time value expressed with second-level precision.

4.10.1.16. dateTimeMilliseconds

The type "dateTimeMilliseconds" represents a time value expressed with millisecond-level precision.

4.10.1.17. dateTimeMicroseconds

The type "dateTimeMicroseconds" represents a time value expressed with microsecond-level precision.

4.10.1.18. dateTimeNanoseconds

The type "dateTimeNanoseconds" represents a time value expressed with nanosecond-level precision.

4.10.1.19. ipv4Address

The type "ipv4Address" represents an IPv4 address as defined in [RFC0791].

4.10.1.20. ipv6Address

The type "ipv6Address" represents an IPv6 address as defined in [RFC3587].

4.10.1.21. ciscoTrainSoftwareVersion

The type "ciscoTrainSoftwareVersion" represents a software version that conforms to the Cisco IOS Train string format.

4.10.1.22. rpmSoftwareVersion

The type "rpmSoftwareVersion" represents a software version that conforms to the EPOCH:VERSION-RELEASE format.

4.10.1.23. simpleSoftwareVersion

The type "simpleSoftwareVersion" represents a software version that is a hierarchical list of non-negative integers separated by a single character delimiter.

4.10.1.24. basicList

The type "basicList" represents a list of zero or more instances of any Information Element as defined in [RFC6313].

4.10.1.25. subTemplateList

The type "subTemplateList" represents a list of zero or more instances of a structured data type, where the data type of each list element is the same and corresponds with a single Template Record as defined in [RFC6313].

4.10.1.26. subTemplateMultiList

The type "subTemplateMultiList" represents a list of zero or more instances of a structured data type, where the data type of each list element can be different and corresponds with different Template definitions as defined in [RFC6313].

4.10.2. Data Type Semantics

This section describes the set of valid data type semantics of the IPFIX information model.

Further data type semantics may be specified by future updates to this document. Changes to the associated "IPFIX Information Element Semantics" subregistry [IANA-IPFIX] require a Standards Action [RFC5226].

4.10.3. quantity

"quantity" is a numeric (integral or floating point) value representing a measured value pertaining to the record. This is distinguished from counters that represent an ongoing measured value whose "odometer" reading is captured as part of a given record. This is the default semantic type of all numeric data types.

4.10.4. totalCounter

"totalCounter" is an integral value reporting the value of a counter. Counters are unsigned and wrap back to zero after reaching the limit of the type. For example, an unsigned64 with counter semantics will continue to increment until reaching the value of 2**64 - 1. At this point, the next increment will wrap its value to zero and continue counting from zero. The semantics of a total counter is similar to the semantics of counters used in the Simple Network Management Protocol (SNMP), such as Counter32 as defined in [RFC2578]. The only difference between total counters and counters used in SNMP is that the total counters have an initial value of 0. A total counter counts independently of the export of its value.

4.10.5. deltaCounter

"deltaCounter" is an integral value reporting the value of a counter. Counters are unsigned and wrap back to zero after reaching the limit of the type. For example, an unsigned64 with counter semantics will continue to increment until reaching the value of 2**64 - 1. At this point, the next increment will wrap its value to zero and continue counting from zero. The semantics of a delta counter is similar to the semantics of counters used in SNMP, such as Counter32 as defined in [RFC2578]. The only difference between delta counters and counters used in SNMP is that the delta counters have an initial value of 0. A delta counter is reset to 0 each time it is exported and/or expires without export.

4.10.6. identifier

"identifier" is an integral value that serves as an identifier. Specifically, mathematical operations on two identifiers (aside from the equality operation) are meaningless. Identifiers MUST be one of the signed or unsigned data types.

4.10.7. flags

"flags" is an integral value that represents a set of bit fields. Logical operations are appropriate on such values, but other mathematical operations are not. Flags MUST always be of an unsigned data type.

4.10.8. list

A list represents an arbitrary-length sequence of zero or more Information Elements, either composed of regular Information Elements or composed of data conforming to a Template Record. See [RFC6313].

5. Information Model Assets

TODO: Explain the different SACM assets. Right now, we have distilled this down to an endpoint, hardware, software, and identity. Previously, this diagram also included account, location, address, and network inteface, but, these things are not assets and can either be consolidated into one of the existing asset types (e.g. network interface => hardware, account => identity, etc.) or are just metadata about the assets (e.g. location => endpoint). We should also explain the types of assets below rather than just referencing out to the Terminology draft.

TODO: The figure below needs to be updated to show the relationships between the different types of assets.

```
           +---------+*______in>_______*+-----+
           |Hardware |                  |!   !|
           |Component|   +---------+    |!   !|
           +---------+   |Software |in> |!   !|
               1|        |Component|____|!   !|
                |        +---------+*  *|!   !|
                |            1|         |!   !|
                |            *|         |     |       +----------+
                |        +---------+    |End- |*_____*| Identity |
               *|        |Software |in> |point| acts  +----------+
           +---------+   |Instance |____|     | for>
           |Hardware |   +---------+*  1|!   !|
           |Instance |__________________|!   !|
           +---------+*      in>       1|!   !|
                                        |!   !|
                                        |!   !|____
                                        |!   !|0..1|
                                        +-----+    |
                                           |*      |
                                           |_______|
                                              in>
```

Figure 9: Model of an Endpoint

5.1. Asset

TODO: Define Asset here in the context of the information model.

[I-D.ietf-sacm-terminology] defines an asset as: Defined in {{RFC4949}} as "a system resource that is (a) required to be protected by an information system's security policy, (b) intended to be protected by a countermeasure, or (c) required for a system's mission". In the scope of SACM, an asset can be composed of other assets. Examples of Assets include: Endpoints, Software, Guidance, or X.509 public key certificates. An asset is not necessarily owned by an organization.

5.2. Endpoint

TODO: Define an Endpoint asset. Explain how it is made up of HW components, SW components, asset identity, etc.

An endpoint is the hollow center of the model. An endpoint is an abstract ideal. Any endpoint attribute assertion that mentions an endpoint mentions it by specifying identifying attributes. Even if there is one preferred endpoint identity, that is modeled as an

identity. We do not anticipate any AVP whose attribute type is "endpoint".

5.3. Hardware Component

TODO: Define a Hardware Component asset. Explain how it is things like motherboards, network cards, etc.

Hardware components may also be assets and/or harmful. For example, a USB port on a system may be disabled to prevent information flow into our out of a particular system; this provides an additional layer of protection that can complement software based protections. Other such assets may include access to or modification of storage media, hardware key stores, microphones and cameras. Like software assets, we can consider these hardware components both from the perspective of (a) an asset that needs protection and (b) an asset that can be compromised in some way to do harm.

A data model MAY designate a hardware component by its manufacturer and a part number.

5.3.1. Hardware Instance

A hardware instance is just an instance of a particular component.

A data model MUST support the following relationships:

- A hardware instance is an "instance of" a hardware component.

- A hardware instance is "in" an endpoint.

Hardware instances may need to be modeled because (a) an endpoint may have multiple instances of a hardware component, (b) a hardware instance may be compromised, whereas other instances may remain intact.

A data model MAY designate a hardware instance by its component and a unique serial number.

5.4. Software Component

TODO: Define a Software Component asset. Explain how it is the software installed on the endpoint including the operating system.

An endpoint contains and runs software components.

Relationship:

o If an endpoint has an instance of a software component, we say that the software component is "in" the endpoint. This is a shorthand.

Some software components are assets. "Asset" is defined in RFC4949 [RFC4949] as "a system resource that is (a) required to be protected by an information system's security policy, (b) intended to be protected by a countermeasure, or (c) required for a system's mission."

An examination of software needs to consider both (a) software assets and (b) software that may do harm. A posture attribute collector may not know (a) from (b). It is useful to define Software Component as the union of (a) and (b).

Examples of Software Assets:

o An application

o A patch

o The operating system kernel

o A boot loader

o Firmware that controls a disk drive

o A piece of JavaScript found in a web page the user visits

Examples of harmful software components:

o A malicious entertainment app

o A malicious executable

o A web page that contains malicious JavaScript

o A business application that shipped with a virus

Software components SHOULD be disjoint from each other. In other words, software componennts SHOULD be so defined that a given byte of software on an endpoint belongs to only one software component.

Different versions of the same piece of software MUST be modeled as different components. Software versioning is not built into the information model.

Each separately installable piece of software SHOULD be modeled as a component. Sometimes it may be better to divide more finely: what an installer installs MAY be modeled as several components.

A data model MAY identify a software component by parts of an ISO SWID tag.

5.4.1. Software Instance

Each copy of a piece of software is called a software instance. The configuration of a software instance is regarded as part of the software instance. Configuration can strongly affect security posture.

A data model MUST support the following relationships:

o A software instance is an "instance of" a software component.

o A software instance is "in" an endpoint.

A data model MAY use ISO SWID tags to describe software instances.

5.5. Asset Identity

TODO: Define an Asset Identity asset. Explain how it is things like user, device, etc. where certificates, usernames, etc. come into place since they are not really hardware or software. NOTE: Make sure it is clear that this is not identity in the sense of what we have been saying endpoint identity (now designation).

5.6. Asset Relationships

TODO: Define the relationships between assets (endpoints, hardware, software, etc.). These will depicted in the overview diagram.

6. Information Model Elements

TODO: Define specific subjects, attributes, and metadata. We may want to consider adding small diagrams showing the relationships between each (see Lisa's notes: https://mailarchive.ietf.org/arch/msg/sacm/kWxlnboHAXD87cned9WavwPZy5w). This may be too much work, but, not sure yet.

The SACM Information Model contains several elements of the architecture, including:

o SACM Components, which may be Collectors, Evaluators, etc. Collectors may be internal (performed within the endpoint itself) or external (performed outside of the endpoint, such as by a hypervisor or remote sensor)

o Guidance, which tells SACM components what to do

o Posture, in the form of posture attributes and evaluation results

o Additional information about the endpoint, such as a representation of a software component, endpoint identity, user identity, address, location, and authorization constraining the endpoint

The SACM Information Model does not (in this draft) specify how long information is retained. Historical information is modeled the same way as current information. Historical information may be represented differently in an implementation, but that difference would be in data models, not in the information model.

Figure 10 introduces the endpoint attributes and their relationships.

```
   +---------+*____in>______*+-----+
   |Hardware |               |!   !|
   |Component| +---------+   |!   !|      +--------+*______________
   +---------+ |Software |in>|!   !|*____*|Location|__________  <in|
       1|      |Component|___|!   !|  in> +--------+*  <in  *|     |
        |      +---------+* *|!   !|                    +-------+  |
        |          1|        |!   !|                    |Account|  |
        |          *|        |    |      +-----------+ +-------+  |
        |     +--------+     |End- |*____*| Identity |______|0..1  |
       *|     |Software|in>  |point| acts +-----------+* belongs    |
   +--------+ |Instance|_____|     | for>   0..1|^        to>       |
   |Hardware| +--------+*   1|!   !|            |acts              |
   |Instance|________________|!   !|           *|for               |*
   +--------+*    in>       1|!   !|______+---------+          +-------+
                             |!   !|1 <in *|Network  |1_______*|Address|
                             |!   !|____  |Interface| <bound   +-------+
                             |!   !|0..1| +---------+   to
                             +-----+    |   *|   |0..1
                                |*      |    |___|
                                |_______|     in>
                                    in>
```

Figure 10: Model of an Endpoint

ISSUE (CEK): we agreed to remove location and account from the model, did we not? TODO: Remove Network Interface, Location, Address, and Account from this diagram if we end up removing the corresponding sections from the information model.

Figure 11 is the core of the information model. It represents the information elements and their relationships.

```
            +-----+            +---------+
            | AVP |____________|Endpoint |
            +-----+1..*       1|Attribute|
                               |Assertion|
                               +---------+
                                 |*                      +-------+
                                 |                       |Summary|
                                 |                       +-------+
                                 |produced-by                *|
                                 |V                           |
                                1|                            |
    +--------+              +-----------+                     |
    |        |              | SACM      |_____________________|
    |Guidance|              | Component |1      <produced-by
    +--------+*____________1+-----------+
               <produced-by
```

Figure 11: Information Elements

Figure 12 is a potential alternative structure for assertions. It is inspired by triple stores. See http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

```
    +-----+______________+---------+                +---------+
    | AVP |1  <subject  *|assertion|________________|predicate|
    |     |______________|         |*  predicate>  1+---------+
    +-----+1  <object   *+---------+
       1^                     |*
        |_____________________|
               <asserter
```

Figure 12: Information Elements, Take 2

Note: UML 2 is specified by [UML].

TODO: update text to match new figure:

Need to be clear in the description that ???

For some of the relationships, will need some language and guidance to the interfaces and relationships we expect to have happen, MUSTs and SHOULDs, as well as explaining the extensibility that other relationships can exist, show examples of how that can happen. Others that we haven't thought of yet, might be added by another RFC or in another way

6.1. Identifying Attributes

TODO: Need to rename this section to align with new "designation" term.

Identifying attributes let a consumer identify an endpoint, for two purposes:

- o To tell whether two endpoint attribute assertions concern the same endpoint o To respond to compliance measurements, for example by reporting, remediating, and quarantining (SACM does not specify these responses, but SACM exists to enable them.)

Out of scope of this section: *classifying* an endpoint so as to apply appropriate collection guidance to it. We don't call this "identification".

6.1.1. How Known

Each attribute-value pair or triple MUST be marked with how the provider knows. There MUST be at least one marking. The possible markings follow.

"Self" means that the endpoint furnished the information: it is self-reported. "Self" does not (necessarily) mean that the provider runs on the the monitored endpoint. Self-reported information is generally subject to the Lying Endpoint Problem. (TODO: citation)

"Authority" means that the provider got the information, directly or indirectly, from an authority that assigned it. For example, the producer got an IP-MAC association from a DHCP server (or was itself the DHCP server).

"Observation" means that the provider got the information from observations of network traffic. For example, the producer saw the source address in an IP packet.

"Verification" means that the provider has verified the information. For example:

* The provider does IP communication with the endpoint and knows the IP address with which it communicates.

* The provider makes an SSH connection to the endpoint and knows the endpoint's public key by virtue of authenticating it.

* The monitored endpoint is a virtual machine and the provider knows by peeking into it.

TODO: Explain security considerations and how consumers are meant to use these markings.

6.1.2. Whether to Include

When publishing an endpoint attribute assertion, the provider MUST publish at least all common identifying AVPs that it knows through verification. If the provider knows none through verification but it knows at least one in another way, it MUST publish at least one. The provider SHOULD publish all common identifying AVPs it knows.

6.1.3. Certificate

6.1.3.1. Range of values

MUST be X.509 certificate, per [RFC5280].

6.1.3.2. Meaning

Throughout the time interval of the AVP, the endpoint had the private key corresponding to the specified certificate.

Throughout the time interval, the certificate was valid: it had a valid certificate chain from a CA certificate that the asserter trusted; every certificate in the chain was time-valid; no certificate in in the chain (excluding the CA certificate) was revoked. ISSUE (CEK): Do we want to get this PKI-ish? If so, would we include the CA certificate as well?

6.1.3.3. Multiplicity

An endpoint may use, or have the right to use, one or more certificates.

Some certificates may be used on more than one endpoint. Other certificates are (by intent) bound to a single endpoint. ISSUE

(CEK): Is there a standard way to distinguish the two? We could perhaps provide a configurable criterion, as an information element. Should we?

6.1.3.4. Stability

Certificates are replaced, due to expiration and other reasons. By and large, they are not replaced often. A year is a typical interval. In sum, they are persistent.

A private key is baked into hardware is almost immutable. But again, hardware can be replaced.

6.1.3.5. Accuracy

If a certificate is known by verification, the attribute is highly accurate.

6.1.3.6. Data model requirements

All SACM data models MUST support this entire subsection.

6.1.4. Public Key

TODO

6.1.5. Username?

ISSUE (CEK): If a user certificate can be an identifying attribute, why not a username also? At an earlier stage of our discussions, usernames were considered common identifying attributes. Did we decide they should not be? Or just forget them?

Many endpoints do not have client certificates. An authenticated username is a useful clue for identifying such an endpoint. I log in only to a handful of personal endpoints. I also present my username and password to many multi-user servers. We would have to distinguish personal endpoints from server endpoints somehow.

6.1.6. Tool-Specific Identifier

TODO

TODO: "Tool-specific identifier" suggests that two tools could never agree on a tool-specific identifier. But a community may agree on an identifier notation, and might even create a formal standard. All that's important is that each of these attributes has a type and

meaning *not* specified by the SACM internet drafts. "Vendor-specific identifier?" "Custom identifier?"

6.1.7. Identification of Endpoints where SACM Components Reside

Every information element needs identifying attributes of its producer's endpoint. (TODO: Provide normative language. SHOULD? MUST?)

Specifically, in an endpoint attribute assertion, we need identifying attributes of the asserter's endpoint. If the asserter is external, the assertion will contain identifying attributes of two endpoints. (TODO: Discuss what this information is for.)

6.1.8. Security Considerations

Effects of misidentification

Things that can cause misidentification

How minimize misidentification

6.2. Identity

TODO: Delete this section?

An identity is the non-secret part of a credential. Examples are a username, an X.500 distinguished name, and a public key. Passwords, private keys, and other secrets are not considered part of an identity.

A data model MUST support the following relationships:

o An endpoint may "act for" an identity. This SHALL mean that the endpoint claims or proves that it has this identity. For example, if the endpoint is part of an Active Directory domain and Alice logs into the endpoint with her AD username (alice) and password, the endpoint "acts for" alice. An endpoint MAY "act for" more than one identity, such as a machine identity and a user identity.

o A identity may "belong to" an account. For example, an enterprise may have a database that maps identities to accounts. CEK: Is this relevant? I don't see how we'd use the notion of an account in identifying an endpoint or in specifying compliance measurements to be taken.

6.3. Location

TODO: Delete this section?

Location can be logical or physical. Location can be a clue to an endpoint's identity.

A data model MUST support the following relationships:

o One or more endpoints may be "in" a location

o A location may be "in" one or more locations

o A network address may be "in" a location

o An account may be "in" a location; this would happen if the account represents a user, and a physical access control system reports on the user's location

Examples of location:

o The switch, access point, VPN gateway, or cell tower to which the endpoint is linked

o The switch port where the endpoint is plugged in

o The location of the endpoint's IP address in the network topology

o The geographic location of the endpoint (which is often self-reported)

o A user location (may be reported by a physical access control system)

CEK: The last three examples seem too advanced for the first set of SACM RFCs. I do not know a notation that would be interoperable and useful for endpoint identification. Should we drop them?

CEK: If we do drop them, all we have left is the device and port at which the endpoint is linked to the network. Maybe we should regard that as a kind of address.

A data model MUST support switch + port number, access point, and VPN gateway as locations. The other examples are optional.

More than one of kind of location may pertain to an endpoint. Endpoint has a many-to-many relationship with Location.

6.4. Endpoint Attribute Assertion

TODO: Integrate into the Section 4 as appropriate.

6.4.1. Form and Precise Meaning

An endpoint attribute assertion has:

- o One or more attribute-value pairs (AVPs)
- o Time intervals over which the AVPs hold
- o Endpoint uniquely identified? True or false
- o Provenance, including:
  - * The SACM component that made the assertion
  - * Information about the method used to derive the assertion

It means that over the specified time interval, there was an endpoint for which all of the listed attribute-value pairs were true.

If the "Endpoint uniquely identified" is true, the set of attributes-value pairs together make this assertion apply to only one endpoint.

The attributes can include posture attributes and identification attributes. The model does not make a rigid distinction between the two uses of attributes.

Some of the attributes may be multi-valued.

One of the AVPs may be a unique endpoint identifier. Not every endpoint will have one. If there is one, the SACM component that produces the Endpoint Attribute Assertion will not necessarily know what it is.

6.4.2. Asserter

An Endpoint Attribute Assertion may come from an attribute collector or an evaluator. It may come from a SACM component that derives it from out-of-band sources, such as a physical inventory system. A SACM component may derive it from other Endpoint Attribute Assertions.

6.4.3. Example

For example, an attribute assertion might have these attribute-value pairs:

mac-address = 01:23:45:67:89:ab

os = OS X

os-version = 10.6.8

This asserts that an endpoint with MAC address 01:23:45:67:89:ab ran OS X 10.6.8 throughout the specified time interval. A profiler might have provided this assertion.

6.4.4. A Use Case

For example, Endpoint Attribute Assertions should help SACM components to track an endpoint as it roams or stays stationary. They must track endpoints because they must track endpoints' postures over time. Tracking of an endpoint can employ many clues, such as:

The endpoint's MAC address

The authenticated identity (even if it identifies a user)

The location of the endpoint and the user

6.4.5. Difference between Attribute and Event

Author: Henk Birkholz

"Attribute" and "event" are often used fairly interchangeably. A clear distinction makes the words more useful.

An *attribute* tends not to change until something causes a change. In contrast, an *event* occurs at a moment in time.

For a nontechnical example, let us consider "openness" as an attribute of a door, with two values, "open" and "closed". A closed door tends to stay closed until something opens it (a breeze, a person, or a dog).

The door's opening or closing is an event.

Similarly, "Host firewall enabled" may be modeled as a true/false attribute of an endpoint. Enabling or disabling the host firewall

may be modeled as an event. An endpoint's crashing also may be modeled as an event.

Although events are not attributes, we use one kind of information element, the "Endpoint Attribute Assertion", to describe both attributes and events.

6.5. Attribute-Value Pair

TODO: Integrate into the Section 4 as appropriate.

The set of attribute types must be extensible, by other IETF standards, by other standards groups, and by vendors. How to express attribute types is not defined here, but is left to data models.

The value may be structured. For example, it may something like XML.

The information model requires a standard attribute type (or possibly more than one) for each box in Figure 10:

o Hardware Component: the value identifies the hardware type. For example, it may consist of the make and model number.

o Hardware Instance: the value, together with the Hardware Component value, uniquely identifies the hardware instance. For example, it may be a manufacturer-assigned serial number. This notion might not apply to all virtual hardware components.

o Software Component: the value identifies a unit of software. Each installable piece of software should be separately identifiable.

For example, this might be a Software Identifier (SWID). Therefore, a software inventory for an endpoint should be expressed as an Endpoint Attribute Assertion.

- o Software Instance: the value describes how the software component is installed and configured.
- o Endpoint: The value is a unique endpoint identifier.
- o Location
- o Identity: The value is the non-secret part of a credential. For example, it may be a certificate, or just a subject Distinguished Name extracted from a certificate. It may be a username.
- o Network Interface: TBD

- o User: [cek: Do we want this? If one user uses different credentials at different times, do we think SACM components will need know that it's the same user?]
- o Address: The value is an IP, MAC, or other network address, possibly qualified with its scope.

6.5.1. Unique Endpoint Identifier

An organization should try to uniquely identify and label an endpoint, whether the endpoint is enrolled or is discovered in the operational environment. The identifier should be assigned by or used in the enrollment process.

Here "unique" means one-to-one. In practice, uniqueness is not always attainable. Even if an endpoint has a unique identifier, an attribute collector may not always know it.

If the attribute type of an AVP is "endpoint", the value is a unique identifier of the endpoint.

6.5.2. Posture Attribute

Some AVPs will be posture attributes.

See the definition in the SACM Terminology for Security Assessment [I-D.ietf-sacm-terminology].

Some potential kinds of posture attributes are:

- o A NEA posture attribute (PA) [RFC5209]
- o A YANG model [RFC6020]
- o An IF-MAP device-characteristics metadata item [TNC-IF-MAP-NETSEC-METADATA]

6.6. Evaluation Result

Evaluation Results (see [I-D.ietf-sacm-terminology]) are modeled as Endpoint Attribute Assertions.

An Evaluation Result derives from one or more other Endpoint Attribute Assertions.

An example is: a NEA access recommendation [RFC5793]

An evaluator may be able to evaluate better if history is available. This is a use case for retaining Endpoint Attribute Assertions for a time.

An Evaluation Result may be retained longer than the Endpoint Attribute Assertions from which it derives. (Figure 10 does not show this.) In the limiting case, Endpoint Attribute Assertions are not retained. When as an Endpoint Attribute Assertion arrives, an evaluator produces an Evaluation Result. These mechanics are out of the scope of the Information Model.

6.7. Organization?

[kkw-from a reporting standpoint there needs to be some concept like organization or system. without this, there is no way to produce result reports that can be acted upon to provide the insight or accountability that almost all continuous monitoring instances are trying to achieve. from a scoring or grading standpoint, an endpoint needs to be associated with exactly one organization or system. it can have a many to many relationship with other types of results reporting "bins". is this important to include here? we had organization as a core asset type for this reason, so i think it is a key information element. but i also know that i do not want to define all the different reporting types, so i am unsure.]

[cek-I had not thought of this at all. Would it make sense to treat the organization and the bins as part of the guidance for creating reports? Maybe not. We should discuss.]

6.8. Guidance

[jmf- the guidance sections need more detail. . .]

[cek - What is missing? We would welcome a critique or text.]

Guidance is generally configurable by human administrators.

6.8.1. Internal Collection Guidance

An internal collector may need guidance to govern what it collects and when.

6.8.2. External Collection Guidance

An external collector may need guidance to govern what it collects and when.

6.8.3. Evaluation Guidance

An evaluator typically needs Evaluation Guidance to govern what it considers to be a good or bad security posture.

6.8.4. Retention Guidance

A SACM deployment may retain posture attributes, events, or evaluation results for some time. Retention supports ad hoc reporting and other use cases.

If information is retained, retention guidance controls what is retained and for how long.

If two or more pieces of retention guidance apply to a piece of information, the guidance calling for the longest retention should take precedence.

6.9. Endpoint

See the definition in the SACM Terminology for Security Assessment [I-D.ietf-sacm-terminology].

In the model, an endpoint can be part of another endpoint. This covers cases where multiple physical endpoints act as one endpoint. The constituent endpoints may not be distinguishable by external observation of network behavior.

For example, a hosting center may maintain a redundant set (redundancy group) of multi-chassis setups to provide active redundancy and load distribution on network paths to WAN gateways. Multi-chassis link aggregation groups make the chassis appear as one endpoint. Traditional security controls must be applied either to physical endpoints or the redundancy groups they compose (and occasionally both). Loss of redundancy is difficult to detect or mitigate without specific posture information about the current state of redundancy groups. Even if a physical endpoint (e.g. router) that is part of a redundancy group is replaced, the redundancy group can remain the same.

6.9.1. Endpoint Identity

An endpoint identity provides both identification and authentication of the endpoint. For example, an identity may be an X.509 certificate [RFC5280] and corresponding private key. [jmf- this example should be formatted like the other examples in this section]

Not all kinds of identities are guaranteed to be unique.

6.9.2. Software Component

An endpoint contains and runs software components.

Some of the software components are assets. "Asset" is defined in RFC4949 [RFC4949] as "a system resource that is (a) required to be protected by an information system's security policy, (b) intended to be protected by a countermeasure, or (c) required for a system's mission."

An examination of software needs to consider both (a) software assets and (b) software that may do harm. A posture attribute collector may not know (a) from (b). It is useful to define Software Component as the union of (a) and (b).

Examples of Software Assets:

- An application
- A patch
- The operating system kernel
- A boot loader
- Firmware that controls a disk drive
- A piece of JavaScript found in a web page the user visits

Examples of harmful software components:

- A malicious entertainment app
- A malicious executable
- A web page that contains malicious JavaScript
- A business application that shipped with a virus

6.9.2.1. Unique Software Identifier

Organizations need to be able to uniquely identify and label software installed or run on an endpoint. Specifically, they need to know the name, publisher, unique ID, and version; and any related patches. In some cases the software's identity might be known a priori by the organization; in other cases, a software identity might be first detected by an organization when the software is first inventoried in an operational environment. Due to this, it is important that an organization have a stable and consistent means to identify software found during collection.

A piece of software may have a unique identifier, such as a SWID tag (ISO/IEC 19770).

6.10. User

6.10.1. User Identity

An endpoint is often - but not always - associated with one or more users.

A user's identity provides both identification and authentication of the user. @@@ Eh?

6.11. hardwareSerialNumber

```
elementId: TBD
name: hardwareSerialNumber
dataType: string
dataTypeSemantics: default
status: current
description: A globally unique identifier for a particular
             piece of hardware assigned by the vendor.
```

6.12. interfaceName

```
elementId: TBD
name: interfaceName
dataType: string
dataTypeSemantics: default
status: current
description: A short name uniquely describing an interface,
             eg "Eth1/0". See [RFC2863] for the definition
             of the ifName object.
```

6.13. interfaceIndex

```
elementId: TBD
name: interfaceIndex
dataType: unsigned32
dataTypeSemantics: identifier
status: current
description: The index of an interface installed on an endpoint.
             The value matches the value of managed object
             'ifIndex' as defined in [RFC2863]. Note that ifIndex
             values are not assigned statically to an interface
             and that the interfaces may be renumbered every time
             the device's management system is re-initialized,
             as specified in [RFC2863].
```

6.14. interfaceMacAddress

```
elementId: TBD
name: interfaceMacAddress
dataType: macAddress
```

dataTypeSemantics: default
status: current
description: The IEEE 802 MAC address associated with a network interface on an endpoint.

6.15. interfaceType

elementId: TBD
name: interfaceType
dataType: unsigned32
dataTypeSemantics: identifier
status: current
description: The type of a network interface. The value matches the value of managed object 'ifType' as defined in [IANA registry ianaiftype-mib].

6.16. interfaceFlags

elementId: TBD
name: interfaceFlags
dataType: unsigned16
dataTypeSemantics: flags
status: current
description: This information element specifies the flags associated with a network interface. Possible values include:

Value	Description
0x1	interface is up
0x2	broadcast address valid
0x4	turn on debugging
0x8	is a loopback net
0x10	interface is point-to-point link
0x20	avoid use of trailers
0x40	resources allocated
0x80	no address resolution protocol
0x100	receive all packets

6.17. networkInterface

elementId: TBD
name: networkInterface
dataType: basicList
dataTypeSemantics: default
status: current
description: Information about a network interface installed on an endpoint. The following high-level digram describes the structure of networkInterface information element.

```
networkInterface = (basicList, allof,
                    interfaceName,
                    interfaceIndex,
                    macAddress,
                    ifType,
                    flags
                    )
```

6.18. softwareIdentifier

elementId: TBD
name: softwareIdentifier
dataType: string
dataTypeSemantics: default
status: current

```
   description: A globally unique identifier for a particular
                software application.
```

6.19. softwareTitle

```
   elementId: TBD
   name: softwareTitle
   dataType: string
   dataTypeSemantics: default
   status: current
   description: The title of the software application.
```

6.20. softwareCreator

```
   elementId: TBD
   name: softwareCreator
   dataType: string
   dataTypeSemantics: default
   status: current
   description: The software developer (e.g., vendor or author).
```

6.21. simpleSoftwareVersion

```
   elementId: TBD
   name: simpleSoftwareVersion
   dataType: simpleVersion
   dataTypeSemantics: default
   status: current
   description: The version string for a software application that
                follows the simple versioning scheme.
```

6.22. rpmSoftwareVersion

```
   elementId: TBD
   name: rpmSoftwareVersion
   dataType: rpmVersion
   dataTypeSemantics: default
   status: current
   description: The version string for a software application that
                follows the RPM versioning scheme.
```

6.23. ciscoTrainSoftwareVersion

```
   elementId: TBD
   name: ciscoTrainSoftwareVersion
   dataType: ciscoTrainVersion
   dataTypeSemantics: default
   status: current
   description: The version string for a software application that
                follows the Cisco Train Release versioning scheme.
```

6.24. softwareVersion

```
   elementId: TBD
   name: softwareVerison
   dataType: basicList
   dataTypeSemantics: default
   status: current
   description: The version of the software application. Software
                applications may be versioned using a number of
                schemas. The following high-level digram describes
                the structure of the softwareVersion information
                element.
   softwareVersion(basicList, exactlyOneOf,
                   simpleSoftwareVersion,
                   rpmSoftwareVersion,
                   ciscoTrainSoftwareVersion,
                   ...
                  )
```

6.25. lastUpdated

```
   elementId: TBD
   name: lastUpdated
   dataType: dateTimeSeconds
   dataTypeSemantics: default
   status: current
   description: The date and time when the software instance
                was last updated on the system (e.g., new
                version instlalled or patch applied)
```

6.26. softwareInstance

```
   elementId: TBD
   name: softwareInstance
   dataType: subTemplateMultiList
   dataTypeSemantics: default
   status: current
   description: Information about an instance of software
                installed on an endpoint. The following
                high-level digram describes the structure of
                softwareInstance information element.

                softwareInstance = (subTemplateMultiList, allof,
                                    softwareIdentifier,
                                    title,
                                    creator,
                                    softwareVersion,
                                    lastUpdated
                                   )
```

6.27. globallyUniqueIdentifier

```
   elementId: TBD
   name: globallyUniqueIdentifier
   dataType: unsigned8
   dataTypeSemantics: identifier
   status: current
   metadata: true
   description: TODO.
```

6.28. dataOrigin

```
   elementId: TBD
   name: dataOrigin
   dataType: string
   dataTypeSemantics: default
   status: current
   metadata: true
   description: The origin of the data. TODO make a better
```

                description.

6.29. dataSource

```
   elementId: TBD
   name: dataSource
   dataType: string
   dataTypeSemantics: default
   status: current
   metadata: true
   description: The source of the data. TODO make a better
                description.
```

6.30. creationTimestamp

```
   elementId: TBD
   name: creationTimestamp
   dataType: dateTimeSeconds
   dataTypeSemantics: default
   status: current
   metadata: true
   description: The date and time when the posture
                information was created by a SACM Component.
```

6.31. collectionTimestamp

```
   elementId: TBD
   name: collectionTimestamp
   dataType: dateTimeSeconds
   dataTypeSemantics: default
   status: current
   metadata: true
   description: The date and time when the posture
                information was collected or observed by a SACM
                Component.
```

6.32. publicationTimestamp

```
   elementId: TBD
   name: publicationTimestamp
   dataType: dateTimeSeconds
   dataTypeSemantics: default
   status: current
   metadata: true
   description: The date and time when the posture
                information was published.
```

6.33. relayTimestamp

```
   elementId: TBD
   name: relayTimestamp
   dataType: dateTimeSeconds
   dataTypeSemantics: default
   status: current
   metadata: true
```

description: The date and time when the posture
             information was relayed to another SACM Component.

6.34. storageTimestamp

elementId: TBD
name: storageTimestamp
dataType: dateTimeSeconds
dataTypeSemantics: default
status: current
metadata: true
description: The date and time when the posture
             information was stored in a Repository.

6.35. type

elementId: TBD
name: type
dataType: unsigned16
dataTypeSemantics: flags
status: current
metadata: true
description: The type of data model use to represent
             some set of endpoint information. The following
             table lists the set of data models supported by SACM.

Value	Description
0x00	Data Model 1
0x01	Data Model 2
0x02	Data Model 3
...	...

6.36. protocolIdentifier

elementId: TBD
name: protocolIdentifier
dataType: unsigned8
dataTypeSemantics: identifier
status: current
description: The value of the protocol number in the IP packet
             header. The protocol number identifies the IP packet
             payload type. Protocol numbers are defined in the
             IANA Protocol Numbers registry.

In Internet Protocol version 4 (IPv4), this is
carried in the Protocol field. In Internet Protocol
version 6 (IPv6), this is carried in the Next Header
field in the last extension header of the packet.

6.37. sourceTransportPort

elementId: TBD
name: sourceTransportPort
dataType: unsigned16
dataTypeSemantics: identifier
status: current
description: The source port identifier in the transport header.
             For the transport protocols UDP, TCP, and SCTP, this
             is the source port number given in the respective
             header. This field MAY also be used for future
             transport protocols that have 16-bit source port
             identifiers.

6.38. sourceIPv4PrefixLength

```
elementId: TBD
name: sourceIPv4PrefixLength
dataType: unsigned8
dataTypeSemantics:
status: current
description: The number of contiguous bits that are relevant in
             the sourceIPv4Prefix Information Element.
```

6.39. ingressInterface

```
elementId: TBD
name: ingressInterface
dataType: unsigned32
dataTypeSemantics: identifier
status: current
description: The index of the IP interface where packets of this
             Flow are being received.  The value matches the
             value of managed object 'ifIndex' as defined in
             [RFC2863]. Note that ifIndex values are not assigned
             statically to an interface and that the interfaces
             may be renumbered every time the device's management
             system is re-initialized, as specified in [RFC2863].
```

6.40. destinationTransportPort

```
elementId: TBD
name: destinationTransportPort
dataType: unsigned16
dataTypeSemantics: identifier
status: current
description: The destination port identifier in the transport
             header. For the transport protocols UDP, TCP, and
             SCTP, this is the destination port number given in
             the respective header. This field MAY also be used
             for future transport protocols that have 16-bit
             destination port identifiers.
```

6.41. sourceIPv6PrefixLength

```
elementId: TBD
name: sourceIPv6PrefixLength
dataType: unsigned8
dataTypeSemantics:
status: current
description: The number of contiguous bits that are relevant in
             the sourceIPv6Prefix Information Element.
```

6.42. sourceIPv4Prefix

```
elementId: TBD
name: sourceIPv4Prefix
dataType: ipv4Address
dataTypeSemantics: default
status: current
description: IPv4 source address prefix.
```

6.43. destinationIPv4Prefix

elementId: TBD
name: destinationIPv4Prefix
dataType: ipv4Address
dataTypeSemantics: default
status: current
description: IPv4 destination address prefix.

6.44. sourceMacAddress

elementId: TBD
name: sourceMacAddress
dataType: macAddress
dataTypeSemantics: default
status: current
description: The IEEE 802 source MAC address field.

6.45. ipVersion

elementId: TBD
name: ipVersion
dataType: unsigned8
dataTypeSemantics: identifier
status: current
description: The IP version field in the IP packet header.

6.46. interfaceDescription

elementId: TBD
name: interfaceDescription
dataType: string
dataTypeSemantics: default
status: current
description: The description of an interface, eg "FastEthernet 1/0" or "ISP connection".

6.47. applicationDescription

elementId: TBD
name: applicationDescription
dataType: string
dataTypeSemantics: default
status: current
description: Specifies the description of an application.

6.48. applicationId

elementId: TBD
name: applicationId
dataType: octetArray
dataTypeSemantics: default
status: current
description: Specifies an Application ID per [RFC6759].

6.49. applicationName

elementId: TBD
name: applicationName
dataType: string
dataTypeSemantics: default
status: current
description: Specifies the name of an application.

6.50. exporterIPv4Address

elementId: TBD
name: exporterIPv4Address
dataType: ipv4Address
dataTypeSemantics: default
status: current

```
   description: The IPv4 address used by the Exporting Process.
                This is used by the Collector to identify the
                Exporter in cases where the identity of the Exporter
                may have been obscured by the use of a proxy.
```

6.51. exporterIPv6Address

```
   elementId: TBD
   name: exporterIPv6Address
   dataType: ipv6Address
   dataTypeSemantics: default
   status: current
   description: The IPv6 address used by the Exporting Process.
                This is used by the Collector to identify the
                Exporter in cases where the identity of the
                Exporter may have been obscured by the use of a
                proxy.
```

6.52. portId

```
   elementId: TBD
   name: portId
   dataType: unsigned32
   dataTypeSemantics: identifier
   status: current
   description: An identifier of a line port that is unique per
                IPFIX Device hosting an Observation Point.
                Typically, this Information Element is used for
                limiting the scope of other Information Elements.
```

6.53. templateId

```
   elementId: TBD
   name: templateId
   dataType: unsigned16
   dataTypeSemantics: identifier
   status: current
   description: An identifier of a Template that is locally unique
                within a combination of a Transport session and an
                Observation Domain.

                Template IDs 0-255 are reserved for Template Sets,
                Options Template Sets, and other reserved Sets yet
                to be created. Template IDs of Data Sets are
                numbered from 256 to 65535.

                Typically, this Information Element is used for
                limiting the scope of other Information Elements.
                Note that after a re-start of the Exporting Process
                Template identifiers may be re-assigned.
```

6.54. collectorIPv4Address

```
   elementId: TBD
   name: collectorIPv4Address
   dataType: ipv4Address
   dataTypeSemantics: default
   status: current
   description: An IPv4 address to which the Exporting Process sends
                Flow information.
```

6.55. collectorIPv6Address

```
   elementId: TBD
   name: collectorIPv6Address
   dataType: ipv6Address
   dataTypeSemantics: default
   status: current
   description: An IPv6 address to which the Exporting Process sends
                Flow information.
```

6.56. informationElementIndex

```
   elementId: TBD
   name: informationElementIndex
   dataType: unsigned16
   dataTypeSemantics: identifier
   status: current
   description: A zero-based index of an Information Element
                referenced by informationElementId within a Template
                referenced by templateId; used to disambiguate
                scope for templates containing multiple identical
                Information Elements.
```

6.57. informationElementId

```
   elementId: TBD
   name: informationElementId
   dataType: unsigned16
   dataTypeSemantics: identifier
   status: current
   description: This Information Element contains the ID of another
                Information Element.
```

6.58. informationElementDataType

```
   elementId: TBD
   name: informationElementDataType
   dataType: unsigned8
   dataTypeSemantics:
   status: current
   description: A description of the abstract data type of an IPFIX
                information element.These are taken from the
                abstract data types defined in section 3.1 of the
                IPFIX Information Model [RFC5102]; see that section
                for more information on the types described in the
                informationElementDataType sub-registry.

                These types are registered in the IANA IPFIX
                Information Element Data Type subregistry.  This
                subregistry is intended to assign numbers for type
                names, not to provide a mechanism for adding data
                types to the IPFIX Protocol, and as such requires a
                Standards Action [RFC5226] to modify.
```

6.59. informationElementDescription

```
elementId: TBD
name: informationElementDescription
dataType: string
dataTypeSemantics: default
status: current
description: A UTF-8 [RFC3629] encoded Unicode string containing
             a human-readable description of an Information
             Element.  The content of the
             informationElementDescription MAY be annotated with
             one or more language tags [RFC4646], encoded
             in-line [RFC2482] within the UTF-8 string, in order
             to specify the language in which the description is
             written.  Description text in multiple languages MAY
             tag each section with its own language tag; in this
             case, the description information in each language
             SHOULD have equivalent meaning.  In the absence of
             any language tag, the "i-default" [RFC2277] language
             SHOULD be assumed.  See the Security Considerations
             section for notes on string handling for Information
             Element type records.
```

6.60. informationElementName

```
elementId: TBD
name: informationElementName
dataType: string
dataTypeSemantics: default
status: current
description: A UTF-8 [RFC3629] encoded Unicode string containing
             the name of an Information Element, intended as a
             simple identifier.  See the Security Considerations
             section for notes on string handling for Information
             Element type records.
```

6.61. informationElementRangeBegin

```
elementId: TBD
name: informationElementRangeBegin
dataType: unsigned64
dataTypeSemantics: quantity
status: current
description: Contains the inclusive low end of the range of
             acceptable values for an Information Element.
```

6.62. informationElementRangeEnd

```
elementId: TBD
name: informationElementRangeEnd
dataType: unsigned64
dataTypeSemantics: quantity
status: current
description: Contains the inclusive high end of the range of
             acceptable values for an Information Element.
```

6.63. informationElementSemantics

```
   elementId: TBD
   name: informationElementSemantics
   dataType: unsigned8
   dataTypeSemantics:
   status: current
   description: A description of the semantics of an IPFIX
                Information Element.  These are taken from the data
                type semantics defined in section 3.2 of the IPFIX
                Information Model [RFC5102]; see that section for
                more information on the types defined in the
                informationElementSemantics sub-registry.  This
                field may take the values in Table ; the special
                value 0x00 (default) is used to note that no
                semantics apply to the field; it cannot be
                manipulated by a Collecting Process or File Reader
                that does not understand it a priori.

                These semantics are registered in the IANA IPFIX
                Information Element Semantics subregistry.  This
                subregistry is intended to assign numbers for
                semantics names, not to provide a mechanism for
                adding semantics to the IPFIX Protocol, and as such
                requires a Standards Action [RFC5226] to modify.
```

6.64. informationElementUnits

```
   elementId: TBD
   name: informationElementUnits
   dataType: unsigned16
   dataTypeSemantics:
   status: current
   description: A description of the units of an IPFIX Information
                Element.  These correspond to the units implicitly
                defined in the Information Element definitions in
                section 5 of the IPFIX Information Model [RFC5102];
                see that section for more information on the types
                described in the informationElementsUnits
                sub-registry. This field may take the values in
                Table 3 below; the special value 0x00 (none) is
                used to note that the field is unitless.

                These types are registered in the IANA IPFIX
                Information Element Units subregistry; new types
                may be added on a First Come First Served [RFC5226]
                basis.
```

6.65. userName

```
   elementId: TBD
   name: userName
   dataType: string
   dataTypeSemantics: default
   status: current
   description: User name associated with the flow.
```

6.66. applicationCategoryName

```
   elementId: TBD
   name: applicationCategoryName
   dataType: string
   dataTypeSemantics: default
   status: current
   description: An attribute that provides a first level
                categorization for each Application ID.
```

6.67. mibObjectValueInteger

```
   elementId: TBD
   name: mibObjectValueInteger
   dataType: signed64
   dataTypeSemantics: identifier
   status: current
   description: An IPFIX Information Element which denotes that the
                integer value of a MIB object will be exported.
                The MIB Object Identifier ("mibObjectIdentifier")
                for this field MUST be exported in a MIB Field
                Option or via another means.  This Information
                Element is used for MIB objects with the Base
                Syntax of Integer32 and INTEGER with IPFIX Reduced
                Size Encoding used as required. The value is
                encoded as per the standard IPFIX Abstract Data Type
                of signed64.
```

6.68. mibObjectValueOctetString

```
   elementId: TBD
   name: mibObjectValueOctetString
   dataType: octetArray
   dataTypeSemantics: default
   status: current
   description: An IPFIX Information Element which denotes that an
                Octet String or Opaque value of a MIB object will
                be exported. The MIB Object Identifier
                ("mibObjectIdentifier") for this field MUST be
                exported in a MIB Field Option or via another means.
                This Information Element is used for MIB objects
                with the Base Syntax of OCTET STRING and Opaque. The
                value is encoded as per the standard IPFIX Abstract
                Data Type of octetArray.
```

6.69. mibObjectValueOID

```
   elementId: TBD
   name: mibObjectValueOID
   dataType: octetArray
   dataTypeSemantics: default
   status: current
   description: An IPFIX Information Element which denotes that an
                Object Identifier or OID value of a MIB object will
                be exported. The MIB Object Identifier
                ("mibObjectIdentifier") for this field MUST be
                exported in a MIB Field Option or via another means.
                This Information Element is used for MIB objects
                with the Base Syntax of OBJECT IDENTIFIER.  Note -
                In this case the "mibObjectIdentifier" will define
                which MIB object is being exported while the value
                contained in this Information Element will be an
                OID as a value.  The mibObjectValueOID Information
                Element is encoded as ASN.1/BER [BER] in an
                octetArray.
```

6.70. mibObjectValueBits

```
   elementId: TBD
   name: mibObjectValueBits
   dataType: octetArray
   dataTypeSemantics: flags
   status: current
   description: An IPFIX Information Element which denotes that a
                set of Enumerated flags or bits from a MIB object
                will be exported. The MIB Object Identifier
                ("mibObjectIdentifier") for this field MUST be
                exported in a MIB Field Option or via another means.
                This Information Element is used for MIB objects
                with the Base Syntax of BITS.  The flags or bits are
                encoded as per the standard IPFIX Abstract Data Type
                of octetArray, with sufficient length to accommodate
                the required number of bits.  If the number of bits
                is not an integer multiple of octets then the most
                significant bits at end of the octetArray MUST be
                set to zero.
```

6.71. mibObjectValueIPAddress

```
   elementId: TBD
   name: mibObjectValueIPAddress
   dataType: ipv4Address
   dataTypeSemantics: default
   status: current
   description: An IPFIX Information Element which denotes that the
                IPv4 Address of a MIB object will be exported.  The
                MIB Object Identifier ("mibObjectIdentifier") for
                this field MUST be exported in a MIB Field Option
                or via another means.  This Information Element is
                used for MIB objects with the Base Syntax of
                IPaddress. The value is encoded as per the standard
                IPFIX Abstract Data Type of ipv4Address.
```

6.72. mibObjectValueCounter

```
   elementId: TBD
   name: mibObjectValueCounter
   dataType: unsigned64
   dataTypeSemantics: snmpCounter
   status: current
   description: An IPFIX Information Element which denotes that the
                counter value of a MIB object will be exported.
                The MIB Object Identifier ("mibObjectIdentifier")
                for this field MUST be exported in a MIB Field
                Option or via another means.  This Information
                Element is used for MIB objects with the Base
                Syntax of Counter32 or Counter64 with IPFIX Reduced
                Size Encoding used as required. The value is encoded
                as per the standard IPFIX Abstract Data Type
                of unsigned64.
```

6.73. mibObjectValueGauge

```
   elementId: TBD
   name: mibObjectValueGauge
   dataType: unsigned32
   dataTypeSemantics: snmpGauge
   status: current
   description: An IPFIX Information Element which denotes that the
                Gauge value of a MIB object will be exported.  The
                MIB Object Identifier ("mibObjectIdentifier") for
                this field MUST be exported in a MIB Field Option
                or via another means.  This Information Element is
                used for MIB objects with the Base Syntax of Gauge32.
                The value is encoded as per the standard IPFIX
                Abstract Data Type of unsigned64.  This value will
                represent a non-negative integer, which may increase
                or decrease, but shall never exceed a maximum
                value, nor fall below a minimum value.
```

6.74. mibObjectValueTimeTicks

```
   elementId: TBD
   name: mibObjectValueTimeTicks
   dataType: unsigned32
   dataTypeSemantics: default
   status: current
   description: An IPFIX Information Element which denotes that the
                TimeTicks value of a MIB object will be exported.
                The MIB Object Identifier ("mibObjectIdentifier")
                for this field MUST be exported in a MIB Field
                Option or via another means.  This Information
                Element is used for MIB objects with the Base
                Syntax of TimeTicks. The value is encoded as per
                the standard IPFIX Abstract Data Type of unsigned32.
```

6.75. mibObjectValueUnsigned

```
   elementId: TBD
   name: mibObjectValueUnsigned
   dataType: unsigned64
   dataTypeSemantics: identifier
   status: current
   description: An IPFIX Information Element which denotes that an
                unsigned integer value of a MIB object will be
                exported.  The MIB Object Identifier
                ("mibObjectIdentifier") for this field MUST be
                exported in a MIB Field Option or via another means.
                This Information Element is used for MIB objects
                with the Base Syntax of unsigned64 with IPFIX
                Reduced Size Encoding used as required. The value is
                encoded as per the standard IPFIX Abstract Data Type
                of unsigned64.
```

6.76. mibObjectValueTable

```
   elementId: TBD
   name: mibObjectValueTable
   dataType: subTemplateList
   dataTypeSemantics: list
   status: current
   description: An IPFIX Information Element which denotes that a
                complete or partial conceptual table will be
                exported.  The MIB Object Identifier
                ("mibObjectIdentifier") for this field MUST be
                exported in a MIB Field Option or via another means.
                This Information Element is used for MIB objects
                with a SYNTAX of SEQUENCE.  This is encoded as a
                subTemplateList of mibObjectValue Information
                Elements.  The template specified in the
                subTemplateList MUST be an Options Template and
                MUST include all the Objects listed in the INDEX
                clause as Scope Fields.
```

6.77. mibObjectValueRow

```
   elementId: TBD
   name: mibObjectValueRow
   dataType: subTemplateList
   dataTypeSemantics: list
   status: current
   description: An IPFIX Information Element which denotes that a
                single row of a conceptual table will be exported.
                The MIB Object Identifier ("mibObjectIdentifier")
                for this field MUST be exported in a MIB Field
                Option or via another means.  This Information
                Element is used for MIB objects with a SYNTAX of
                SEQUENCE.  This is encoded as a subTemplateList of
                mibObjectValue Information Elements.  The
                subTemplateList exported MUST contain exactly one
                row (i.e., one instance of the subtemplate).  The
                template specified in the subTemplateList MUST be
                an Options Template and MUST include all the
                Objects listed in the INDEX clause as Scope Fields.
```

6.78. mibObjectIdentifier

```
   elementId: TBD
   name: mibObjectIdentifier
   dataType: octetArray
   dataTypeSemantics: default
   status: current
   description: An IPFIX Information Element which denotes that a
                MIB Object Identifier (MIB OID) is exported in the
                (Options) Template Record.  The mibObjectIdentifier
                Information Element contains the OID assigned to
                the MIB Object Type Definition encoded as
                ASN.1/BER [BER].
```

6.79. mibSubIdentifier

```
   elementId: TBD
   name: mibSubIdentifier
   dataType: unsigned32
   dataTypeSemantics: identifier
   status: current
   description: A non-negative sub-identifier of an Object
                Identifier (OID).
```

6.80. mibIndexIndicator

```
   elementId: TBD
   name: mibIndexIndicator
   dataType: unsigned64
   dataTypeSemantics: flags
   status: current
   description: This set of bit fields is used for marking the
                Information Elements of a Data Record that serve as
                INDEX MIB objects for an indexed Columnar MIB
                object.  Each bit represents an Information Element
                in the Data Record with the n-th bit representing
                the n-th Information Element.  A bit set to value 1
                indicates that the corresponding Information Element
                is an index of the Columnar Object represented by
                the mibFieldValue.  A bit set to value 0 indicates
                that this is not the case.

                If the Data Record contains more than 64
                Information Elements, the corresponding Template
                SHOULD be designed such that all INDEX
                Fields are among the first 64 Information Elements,
                because the mibIndexIndicator only contains 64 bits.
                If the Data Record contains less than 64
                Information Elements, then the extra bits in the
                mibIndexIndicator for which no corresponding
                Information Element exists MUST have the value 0,
                and must be disregarded by the Collector.  This
                Information Element may be exported with
                IPFIX Reduced Size Encoding.
```

6.81. mibCaptureTimeSemantics

```
   elementId: TBD
   name: mibCaptureTimeSemantics
   dataType: unsigned8
   dataTypeSemantics: identifier
   status: current
   description: Indicates when in the lifetime of the flow the MIB
                value was retrieved from the MIB for a
                mibObjectIdentifier.  This is used to indicate if
                the value exported was collected from the MIB
                closer to flow creation or flow export time and
                will refer to the Timestamp fields included in the
                same record.  This field SHOULD be used when
                exporting a mibObjectValue that specifies counters
                or statistics.

                If the MIB value was sampled by SNMP prior to the
                IPFIX Metering Process or Exporting Process
                retrieving the value (i.e., the data is already
                stale) and it's important to know the exact sampling
                time, then an additional observationTime* element
                should be paired with the OID using structured data.
                Similarly, if different mibCaptureTimeSemantics
                apply to different mibObject elements within the
                Data Record, then individual mibCaptureTimeSemantics
                should be paired with each OID using structured data.

                Values:
                0.  undefined
                1.  begin - The value for the MIB object is captured
                from the MIB when the Flow is first observed
                2.  end - The value for the MIB object is captured
                from the MIB when the Flow ends
                3.  export - The value for the MIB object is
                captured from the MIB at export time
                4.  average - The value for the MIB object is an
                average of multiple captures from the MIB over the
                observed life of the Flow
```

6.82. mibContextEngineID

```
   elementId: TBD
   name: mibContextEngineID
   dataType: octetArray
   dataTypeSemantics: default
   status: current
   description: A mibContextEngineID that specifies the SNMP engine
                ID for a MIB field being exported over IPFIX.
                Definition as per [RFC3411] section 3.3.
```

6.83. mibContextName

```
   elementId: TBD
   name: mibContextName
   dataType: string
   dataTypeSemantics: default
   status: current
   description: This Information Element denotes that a MIB Context
                Name is specified for a MIB field being exported
                over IPFIX. Reference [RFC3411] section 3.3.
```

6.84. mibObjectName

```
   elementId: TBD
   name: mibObjectName
   dataType: string
   dataTypeSemantics: default
   status: current
   description: The name (called a descriptor in [RFC2578]
                of an object type definition.
```

6.85. mibObjectDescription

```
   elementId: TBD
   name: mibObjectDescription
   dataType: string
   dataTypeSemantics: default
   status: current
   description: The value of the DESCRIPTION clause of an MIB object
                type definition.
```

6.86. mibObjectSyntax

```
   elementId: TBD
   name: mibObjectSyntax
   dataType: string
   dataTypeSemantics: default
   status: current
   description: The value of the SYNTAX clause of an MIB object type
                definition, which may include a Textual Convention
                or Subtyping. See [RFC2578].
```

6.87. mibModuleName

```
   elementId: TBD
   name: mibModuleName
   dataType: string
   dataTypeSemantics: default
   status: current
   description: The textual name of the MIB module that defines a MIB
                Object.
```

7. SACM Usage Scenario Example

TODO: this section needs to refer out to wherever the operations / generalized workflow content ends up

TODO: revise to eliminate graph references

This section illustrates the proposed SACM Information Model as applied to SACM Usage Scenario 2.2.3, Detection of Posture Deviations [RFC7632]. The following subsections describe the elements (components and elements), graph model, and operations (sample workflow) required to support the Detection of Posture Deviations scenario.

The Detection of Posture Deviations scenario involves multiple elements interacting to accomplish the goals of the scenario. Figure 10 illustrates those elements along with their major communication paths.

7.1. Graph Model for Detection of Posture Deviation

The following subsections contain examples of identifiers and metadata which would enable detection of posture deviation. These lists are by no means exhaustive - many other types of metadata would be enumerated in a data model that fully addressed this usage scenario.

7.1.1. Components

The proposed SACM Information Model contains three components, as defined in the SACM Architecture [I-D.ietf-sacm-architecture]: Posture Attribute Information Provider, Posture Attribute Information Consumer, and Control Plane.

In this example, the components are instantiated as follows:

- o The Posture Attribute Information Provider is an endpoint security service which monitors the compliance state of the endpoint and reports any deviations for the expected posture.

- o The Posture Attribute Information Consumer is an analytics engine which absorbs information from around the network and generates a "heat map" of which areas in the network are seeing unusually high rates of posture deviations.

- o The Control Plane is a security automation broker which receives subscription requests from the analytics engine and authorizes access to appropriate information from the endpoint security service.

7.1.2. Identifiers

To represent the elements listed above, the set of identifiers might include (but is not limited to):

- o Identity - a device itself, or a user operating a device, categorized by type of identity (e.g. username or X.509 certificate [RFC5280])

- o Software asset

- o Network Session

- o Address - categorized by type of address (e.g. MAC address, IP address, Host Identity Protocol (HIP) Host Identity Tag (HIT) [RFC5201], etc.)

- o Task - categorized by type of task (e.g. internal collector, external collector, evaluator, or reporting task)

- o Result - categorized by type of result (e.g. evaluation result or report)

- o Guidance

7.1.3. Metadata

To characterize the elements listed above, the set of metadata types might include (but is not limited to):

- o Authorization metadata attached to an identity identifier, or to a link between a network session identifier and an identity identifier, or to a link between a network session identifier and an address identifier.

- o Location metadata attached to a link between a network session identifier and an address identifier.

- o Event metadata attached to an address identifier or an identity identifier of an endpoint, which would be made available to interested parties at the time of publication, but not stored long-term. For example, when a user disables required security software, an internal collector associated with an endpoint security service might publish guidance violation event metadata attached to the identity identifier of the endpoint, to notify consumers of the change in endpoint state.

- o Posture attribute metadata attached to an identity identifier of an endpoint. For example, when required security software is not running, an internal collector associated with an endpoint security service might publish posture attribute metadata attached to the identity identifier of the endpoint, to notify consumers of the current state of the endpoint.

7.1.4. Relationships between Identifiers and Metadata

Interaction between multiple sets of identifiers and metadata lead to some fairly common patterns, or "constellations", of metadata. For example, an authenticated-session metadata constellation might include a central network session with authorizations and location attached, and links to a user identity, an endpoint identity, a MAC address, an IP address, and the identity of the policy server that authorized the session, for the duration of the network session.

These constellations may be independent of each other, or one constellation may be connected to another. For example, an authenticated-session metadata constellation may be created when a user connects an endpoint to the network; separately, an endpoint-posture metadata constellation may be created when an endpoint security system and other collectors gather and publish posture information related to an endpoint. These two constellations are not necessarily connected to each other, but may be joined if the component publishing the authenticated-session metadata constellation is able to link the network session identifier to the identity identifier of the endpoint.

7.2. Workflow

The workflow for exchange of information supporting detection of posture deviation, using a standard publish/subscribe/query transport model such as available with IF-MAP [TNC-IF-MAP-SOAP-Binding] or XMPP-Grid [I-D.salowey-sacm-xmpp-grid], is as follows:

1. The analytics engine (Posture Assessment Information Consumer) establishes connectivity and authorization with the transport fabric, and subscribes to updates on posture deviations.

2. The endpoint security service (Posture Assessment Information Provider) requests connection to the transport fabric.

3. Transport fabric authenticates and establishes authorized privileges (e.g. privilege to publish and/or subscribe to security data) for the requesting components.

4. The endpoint security service evaluates the endpoint, detects posture deviation, and publishes information on the posture deviation.

5. The transport fabric notifies the analytics engine, based on its subscription of the new posture deviation information.

Other components, such as access control policy servers or remediation systems, may also consume the posture deviation information provided by the endpoint security service.

8. Acknowledgements

Many of the specifications in this document have been developed in a public-private partnership with vendors and end-users. The hard work of the SCAP community is appreciated in advancing these efforts to their current level of adoption.

Over the course of developing the initial draft, Brant Cheikes, Matt Hansbury, Daniel Haynes, Scott Pope, Charles Schmidt, and Steve Venema have contributed text to many sections of this document.

8.1. Contributors

The RFC guidelines no longer allow RFCs to be published with a large number of authors. Some additional authors contributed to specific sections of this document; their names are listed in the individual section headings as well as alphabetically listed with their affiliations below.

Name	Affiliation	Contact
Henk Birkholz	Fraunhofer SIT	henk.birkholz@sit.fraunhofer.de

9. IANA Considerations

This memo includes no request to IANA.

10. Operational Considerations

TODO: Need to include various operational considerations here. Proposed sections include timestamp accuracy and which attributes attributes designate an endpoint.

11. Privacy Considerations

TODO: Need to include various privacy considerations here.

12. Security Considerations

Posture Assessments need to be performed in a safe and secure manner. In that regard, there are multiple aspects of security that apply to the communications between components as well as the capabilities themselves. Due to time constraints, this information model only contains an initial listing of items that need to be considered with respect to security. This list is not exhaustive, and will need to be augmented as the model continues to be developed/refined.

Initial list of security considerations include:

Authentication: Every component and asset needs to be able to identify itself and verify the identity of other components and assets.

Confidentiality: Communications between components need to be protected from eavesdropping or unauthorized collection. Some communications between components and assets may need to be protected as well.

Integrity: The information exchanged between components needs to be protected from modification. some exchanges between assets and components will also have this requirement.

Restricted Access: Access to the information collected, evaluated, reported, and stored should only be viewable/consumable to authenticated and authorized entities.

The TNC IF-MAP Binding for SOAP [TNC-IF-MAP-SOAP-Binding] and TNC IF-MAP Metadata for Network Security [TNC-IF-MAP-NETSEC-METADATA] document security considerations for sharing information via security automation. Most, and possibly all, of these considerations also apply to information shared via this proposed information model.

13. References

13.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <http://www.rfc-editor.org/info/rfc791>.

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

[RFC3587] Hinden, R., Deering, S., and E. Nordmark, "IPv6 Global Unicast Address Format", RFC 3587, DOI 10.17487/RFC3587, August 2003, <http://www.rfc-editor.org/info/rfc3587>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

[RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, DOI 10.17487/RFC6313, July 2011, <http://www.rfc-editor.org/info/rfc6313>.

13.2. Informative References

[CCE] The National Institute of Standards and Technology, "Common Configuration Enumeration", 2014, <http://nvd.nist.gov/CCE/>.

[CCI] United States Department of Defense Defense Information Systems Agency, "Control Correlation Identifier", 2014, <http://iase.disa.mil/cci/>.

[CPE-WEBSITE] The National Institute of Standards and Technology, "Common Platform Enumeration", 2014, <http://scap.nist.gov/specifications/cpe/>.

[CVE-WEBSITE] The MITRE Corporation, "Common Vulnerabilities and Exposures", 2014, <http://cve.mitre.org/about/>.

[I-D.ietf-sacm-architecture] Cam-Winget, N., Ford, B., Lorenzin, L., McDonald, I., and l. loxx@cisco.com, "Secure Automation and Continuous Monitoring (SACM) Architecture", draft-ietf-sacm-architecture-00 (work in progress), October 2014.

[I-D.ietf-sacm-requirements] Cam-Winget, N. and L. Lorenzin, "Secure Automation and

Continuous Monitoring (SACM) Requirements", draft-ietf-sacm-requirements-01 (work in progress), October 2014.

[I-D.ietf-sacm-terminology]
Waltermire, D., Montville, A., Harrington, D., and N. Cam-Winget, "Terminology for Security Assessment", draft-ietf-sacm-terminology-05 (work in progress), August 2014.

[I-D.salowey-sacm-xmpp-grid]
Salowey, J., Lorenzin, L., Kahn, C., Pope, S., Appala, S., Woland, A., and N. Cam-Winget, "XMPP Protocol Extensions for Use in SACM Information Transport", draft-salowey-sacm-xmpp-grid-00 (work in progress), July 2014.

[IM-LIAISON-STATEMENT-NIST]
Montville, A., "Liaison Statement: Call for Contributions for the SACM Information Model to NIST", May 2014, <http://datatracker.ietf.org/liaison/1329/>.

[ISO.18180]
"Information technology -- Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.2", ISO/IEC 18180, 2013, <http://standards.iso.org/ittf/PubliclyAvailableStandards/c061713_ISO_IEC_18180_2013.zip>.

[ISO.19770-2]
"Information technology -- Software asset management -- Part 2: Software identification tag", ISO/IEC 19770-2, 2009.

[NISTIR-7275]
Waltermire, D., Schmidt, C., Scarfone, K., and N. Ziring, "Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.2", NISTIR 7275r4, March 2013, <http://csrc.nist.gov/publications/nistir/ir7275-rev4/nistir-7275r4_updated-march-2012_clean.pdf>.

[NISTIR-7693]
Wunder, J., Halbardier, A., and D. Waltermire, "Specification for Asset Identification 1.1", NISTIR 7693, June 2011, <http://csrc.nist.gov/publications/nistir/ir7693/NISTIR-7693.pdf>.

[NISTIR-7694]
Halbardier, A., Waltermire, D., and M. Johnson, "Specification for the Asset Reporting Format 1.1", NISTIR 7694, June 2011, <http://csrc.nist.gov/publications/nistir/ir7694/NISTIR-7694.pdf>.

[NISTIR-7695]
Cheikes, B., Waltermire, D., and K. Scarfone, "Common Platform Enumeration: Naming Specification Version 2.3", NISTIR 7695, August 2011, <http://csrc.nist.gov/publications/nistir/ir7695/NISTIR-7695-CPE-Naming.pdf>.

[NISTIR-7696]
Parmelee, M., Booth, H., Waltermire, D., and K. Scarfone, "Common Platform Enumeration: Name Matching Specification Version 2.3", NISTIR 7696, August 2011, <http://csrc.nist.gov/publications/nistir/ir7696/NISTIR-7696-CPE-Matching.pdf>.

[NISTIR-7697]
Cichonski, P., Waltermire, D., and K. Scarfone, "Common Platform Enumeration: Dictionary Specification Version 2.3", NISTIR 7697, August 2011, <http://csrc.nist.gov/publications/nistir/ir7697/

NISTIR-7697-CPE-Dictionary.pdf>.

[NISTIR-7698]
Waltermire, D., Cichonski, P., and K. Scarfone, "Common Platform Enumeration: Applicability Language Specification Version 2.3", NISTIR 7698, August 2011, <http://csrc.nist.gov/publications/nistir/ir7698/NISTIR-7698-CPE-Language.pdf>.

[NISTIR-7848]
Davidson, M., Halbardier, A., and D. Waltermire, "Specification for the Asset Summary Reporting Format 1.0", NISTIR 7848, May 2012, <http://csrc.nist.gov/publications/drafts/nistir-7848/draft_nistir_7848.pdf>.

[OVAL-LANGUAGE]
Baker, J., Hansbury, M., and D. Haynes, "The OVAL Language Specification version 5.10.1", January 2012, <https://oval.mitre.org/language/version5.10.1/>.

[RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, DOI 10.17487/RFC3411, December 2002, <http://www.rfc-editor.org/info/rfc3411>.

[RFC3416] Presuhn, R., Ed., "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3416, DOI 10.17487/RFC3416, December 2002, <http://www.rfc-editor.org/info/rfc3416>.

[RFC3418] Presuhn, R., Ed., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, DOI 10.17487/RFC3418, December 2002, <http://www.rfc-editor.org/info/rfc3418>.

[RFC3580] Congdon, P., Aboba, B., Smith, A., Zorn, G., and J. Roese, "IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines", RFC 3580, DOI 10.17487/RFC3580, September 2003, <http://www.rfc-editor.org/info/rfc3580>.

[RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, DOI 10.17487/RFC4287, December 2005, <http://www.rfc-editor.org/info/rfc4287>.

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

[RFC5201] Moskowitz, R., Nikander, P., Jokela, P., Ed., and T. Henderson, "Host Identity Protocol", RFC 5201, DOI 10.17487/RFC5201, April 2008, <http://www.rfc-editor.org/info/rfc5201>.

[RFC5209] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", RFC 5209, DOI 10.17487/RFC5209, June 2008, <http://www.rfc-editor.org/info/rfc5209>.

[RFC5792] Sangster, P. and K. Narayan, "PA-TNC: A Posture Attribute (PA) Protocol Compatible with Trusted Network Connect

(TNC)", RFC 5792, DOI 10.17487/RFC5792, March 2010, <http://www.rfc-editor.org/info/rfc5792>.

[RFC5793] Sahita, R., Hanna, S., Hurst, R., and K. Narayan, "PB-TNC: A Posture Broker (PB) Protocol Compatible with Trusted Network Connect (TNC)", RFC 5793, DOI 10.17487/RFC5793, March 2010, <http://www.rfc-editor.org/info/rfc5793>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6876] Sangster, P., Cam-Winget, N., and J. Salowey, "A Posture Transport Protocol over TLS (PT-TLS)", RFC 6876, DOI 10.17487/RFC6876, February 2013, <http://www.rfc-editor.org/info/rfc6876>.

[RFC7012] Claise, B., Ed. and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <http://www.rfc-editor.org/info/rfc7012>.

[RFC7013] Trammell, B. and B. Claise, "Guidelines for Authors and Reviewers of IP Flow Information Export (IPFIX) Information Elements", BCP 184, RFC 7013, DOI 10.17487/RFC7013, September 2013, <http://www.rfc-editor.org/info/rfc7013>.

[RFC7171] Cam-Winget, N. and P. Sangster, "PT-EAP: Posture Transport (PT) Protocol for Extensible Authentication Protocol (EAP) Tunnel Methods", RFC 7171, DOI 10.17487/RFC7171, May 2014, <http://www.rfc-editor.org/info/rfc7171>.

[RFC7632] Waltermire, D. and D. Harrington, "Endpoint Security Posture Assessment: Enterprise Use Cases", RFC 7632, DOI 10.17487/RFC7632, September 2015, <http://www.rfc-editor.org/info/rfc7632>.

[SP800-117]
Quinn, S., Scarfone, K., and D. Waltermire, "Guide to Adopting and Using the Security Content Automation Protocol (SCAP) Version 1.2", SP 800-117, January 2012, <http://csrc.nist.gov/publications/drafts/800-117-R1/Draft-SP800-117-r1.pdf>.

[SP800-126]
Waltermire, D., Quinn, S., Scarfone, K., and A. Halbardier, "The Technical Specification for the Security Content Automation Protocol (SCAP): SCAP Version 1.2", SP 800-126, September 2011, <http://csrc.nist.gov/publications/nistpubs/800-126-rev2/SP800-126r2.pdf>.

[TNC-Architecture]
Trusted Computing Group, ""TNC Architecture", Specification Version 1.5", May 2012.

[TNC-IF-M-TLV-Binding]
Trusted Computing Group, ""TNC IF-M: TLV Binding", Specification Version 1.0", May 2014.

[TNC-IF-MAP-ICS-METADATA]
Trusted Computing Group, ""TNC IF-MAP Metadata for ICS Security", Specification Version 1.0", May 2014.

[TNC-IF-MAP-NETSEC-METADATA]
Trusted Computing Group, ""TNC IF-MAP Metadata for Network Security", Specification Version 1.1", May 2012.

[TNC-IF-MAP-SOAP-Binding]
Trusted Computing Group, ""TNC IF-MAP Binding for SOAP", Specification Version 2.2", March 2014.

[TNC-IF-T-TLS]
Trusted Computing Group, ""TNC IF-T: Binding to TLS", Specification Version 2.0", February 2013.

[TNC-IF-T-Tunneled-EAP]
Trusted Computing Group, ""TNC IF-T: Protocol Bindings for Tunneled EAP Methods", Specification Version 2.0", May 2014.

[TNC-IF-TNCCS-TLV-Binding]
Trusted Computing Group, ""TNC IF-TNCCS: TLV Binding", Specification Version 2.0", May 2014.

[UML]   Object Management Group, ""Unified Modeling Language TM (UML (R))", Version 2.4.1", August 2011.

[W3C.REC-rdf11-concepts-20140225]
Cyganiak, R., Wood, D., and M. Lanthaler, "RDF 1.1 Concepts and Abstract Syntax", World Wide Web Consortium Recommendation REC-rdf11-concepts-20140225, February 2014, <http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225>.

[W3C.REC-soap12-part1-20070427]
Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., Nielsen, H., Karmarkar, A., and Y. Lafon, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)", World Wide Web Consortium Recommendation REC-soap12-part1-20070427, April 2007, <http://www.w3.org/TR/2007/REC-soap12-part1-20070427>.

Appendix A. Change Log

A.1. Changes in Revision 01

Renamed "credential" to "identity", following industry usage. A credential includes proof, such as a key or password. A username or a distinguished name is called an "identity".

Removed Session, because an endpoint's network activity is not SACM's initial focus

Removed Authorization, for the same reason

Added many-to-many relationship between Hardware Component and Endpoint, for clarity

Added many-to-many relationship between Software Component and Endpoint, for clarity

Added "contains" relationship between Network Interface and Network Interface

Removed relationship between Network Interface and Account. The endpoint knows the identity it used to gain network access. The PDP also knows that. But they probably do not know the account.

Added relationship between Network Interface and Identity. The endpoint and the PDP will typically know the identity.

Made identity-to-account a many-to-one relationship.

A.2. Changes in Revision 02

Added Section 6.1, Identifying Attributes.

Split the figure into Figure 10 and Figure 11.

Added Figure 12, proposing a triple-store model.

Some editorial cleanup

A.3. Changes in Revision 03

Moved Appendix A.1, Appendix A.2, and Appendix B into the Appendix. Added a reference to it in Section 1

Added the Section 4 section. Provided notes for the type of information we need to add in this section.

Added the Section 5 section. Moved sections on Endpoint, Hardware Component, Software Component, Hardware Instance, and Software Instance there. Provided notes for the type of information we need to add in this section.

Removed the Provenance of Information Section. SACM is not going to solve provenance rather give organizations enough information to figure it out.

Updated references to the Endpoint Security Posture Assessment: Enterprise Use Cases document to reflect that it was published as an RFC.

Fixed the formatting of a few figures.

Included references to [RFC3580] where RADIUS is mentioned.

A.4. Changes in Revision 04

Integrated the IPFIX [RFC7012] syntax into Section 4.

Converted many of the existing SACM Information Elements to the IPFIX syntax.

Included existing IPFIX Information Elements and datatypes that could likely be reused for SACM in Section 6 and Section 4 respectively.

Removed the sections related to reports as described in https://github.com/sacmwg/draft-ietf-sacm-information-model/issues/30.

Cleaned up other text throughout the document.

A.5. Changes in Revision 05

Merged proposed changes from the I-D IM into the WG IM (https://github.com/sacmwg/draft-ietf-sacm-information-model/issues/41).

Fixed some formatting warnings.

Removed a duplicate IE and added a few IE datatypes that were missing.

Appendix B. Mapping to SACM Use Cases

TODO: revise

(wandw)This information model directly corresponds to all four use cases defined in the SACM Use Cases draft [RFC7632]. It uses these use cases in coordination to achieve a small set of well-defined tasks.

Sections [removed] thru [removed] address each of the process areas. For each process area, a "Process Area Description" sub-section represent an end state that is consistent with all the General Requirements and many of the Use Case Requirements identified in the requirements draft [I-D.ietf-sacm-requirements].

The management process areas and supporting operations defined in this memo directly support REQ004 Endpoint Discovery; REQ005-006 Attribute and Information Based Queries, and REQ0007 Asynchronous Publication.

In addition, the operations that defined for each business process in this memo directly correlate with the typical workflow identified in the SACM Use Case document.(/wandw)

Appendix C. Security Automation with TNC IF-MAP

C.1. What is Trusted Network Connect?

Trusted Network Connect (TNC) is a vendor-neutral open architecture [TNC-Architecture] and a set of open standards for network security developed by the Trusted Computing Group (TCG). TNC standards integrate security components across end user systems, servers, and network infrastructure devices into an intelligent, responsive, coordinated defense. TNC standards have been widely adopted by vendors and customers; the TNC endpoint assessment protocols [TNC-IF-

M-TLV-Binding][TNC-IF-TNCCS-TLV-Binding][TNC-IF-T-Tunneled-EAP][TNC-I F-T-TLS] were used as the base for the IETF NEA RFCs [RFC5792][RFC5793][RFC7171][RFC6876].

Traditional information security architectures have separate silos for endpoint security, network security, server security, physical security, etc. The TNC architecture enables the integration and categorization of security telemetry sources via the information model contained in its Interface for Metadata Access Points (IF-MAP) [TNC-IF-MAP-SOAP-Binding]. IF-MAP provides a query-able repository of security telemetry that may be used for storage or retrieval of such data by multiple types of security systems and endpoints on a vendor-neutral basis. The information model underlying the IF-MAP repository covers, directly or indirectly, all of the security information types required to serve SACM use-cases.

C.2. What is TNC IF-MAP?

IF-MAP provides a standard client-server protocol for MAP clients to exchange security-relevant information via database server known as the Metadata Access Point or MAP. The data (known as "metadata") stored in the MAP is XML data. Each piece of metadata is tagged with a metadata type that indicates the meaning of the metadata and identifies an XML schema for it. Due to the XML language, the set of metadata types is easily extensible.

The MAP is a graph database, not a relational database. Metadata can be associated with an identifier (e.g. the email address "user@example.com") or with a link between two identifiers (e.g. the link between MAC address 00:11:22:33:44:55 and IPv4 address 192.0.2.1) where the link defines an association (for example: a relation or state) between the identifiers. These links between pairs of identifiers create an ad hoc graph of relationships between identifiers. The emergent structure of this graph reflects a continuously evolving knowledge base of security-related metadata that is shared between various providers and consumers.

C.3. What is the TNC Information Model?

The TNC Information Model underlying IF-MAP relies on the graph database architecture to enable a (potentially distributed) MAP service to act as a shared clearinghouse for information that infrastructure devices can act upon. The IF-MAP operations and metadata schema specifications (TNC IF-MAP Binding for SOAP [TNC-IF-MAP-SOAP-Binding], TNC IF-MAP Metadata for Network Security [TNC-IF-MAP-NETSEC-METADATA], and TNC IF-MAP Metadata for ICS

Security [TNC-IF-MAP-ICS-METADATA]) define an extensible set of identifiers and data types.

Each IF-MAP client may interact with the IF-MAP graph data store through three fundamental types of operation requests:

o Publish, which may create, modify, or delete metadata associated with one or more identifiers and/or links in the graph

o Search, which retrieves a selected sub-graph according to a set of search criteria

o Subscribe, which allows a client to manage a set of search commands which asynchronously return selected sub-graphs when changes to that sub-graph are made by other IF-MAP clients

The reader is invited to review the existing IF-MAP specification [TNC-IF-MAP-SOAP-Binding] for more details on the above graph data store operation requests and their associated arguments.

The current IF-MAP specification provides a SOAP [W3C.REC-soap12-part1-20070427] binding for the above operations, as well as associated SOAP operations for managing sessions, error handling, etc.

Appendix D. Text for Possible Inclusion in the Terminology Draft

D.1. Terms and Definitions

This section describes terms that have been defined by other RFCs and Internet Drafts, as well as new terms introduced in this document.

D.1.1. Pre-defined and Modified Terms

This section contains pre-defined terms that are sourced from other IETF RFCs and Internet Drafts. Descriptions of terms in this section will reference the original source of the term and will provide additional specific context for the use of each term in SACM. For sake of brevity, terms from [I-D.ietf-sacm-terminology] are not repeated here unless the original meaning has been changed in this document.

Asset   For this Information Model it is necessary to change the scope of the definition of asset from the one provided in [I-D.ietf-sacm-terminology]. Originally defined in [RFC4949] and referenced in [I-D.ietf-sacm-terminology] as "a system resource that is (a) required to be protected by an information system's security policy, (b) intended to be protected by a countermeasure, or (c) required for a system's mission." This definition generally relates to an "IT Asset", which in the context of this document is overly

limiting. For use in this document, a broader definition of the term is needed to represent non-IT asset types as well.

In [NISTIR-7693] an asset is defined as "anything that has value to an organization, including, but not limited to, another organization, person, computing device, information technology (IT) system, IT network, IT circuit, software (both an installed instance and a physical instance), virtual computing platform (common in cloud and virtualized computing), and related hardware (e.g., locks, cabinets, keyboards)." This definition aligns better with common dictionary definitions of the term and better fits the needs of this document.

D.1.2. New Terms

IT Asset Originally defined in [RFC4949] as "a system resource that is (a) required to be protected by an information system's security policy, (b) intended to be protected by a countermeasure, or (c) required for a system's mission."

Security Content Automation Protocol (SCAP) According to SP800-126, SCAP, pronounced "ess-cap", is "a suite of specifications that standardize the format and nomenclature by which software flaw and security configuration information is communicated, both to machines and humans." SP800-117 revision 1 [SP800-117] provides a general overview of SCAP 1.2. The 11 specifications that comprise SCAP 1.2 are synthesized by a master specification, SP800-126 revision 2 [SP800-126], that addresses integration of the specifications into a coherent whole. The use of "protocol" in its name is a misnomer, as SCAP defines only data models. SCAP has been adopted by a number of operating system and security tool vendors.

Appendix E. Text for Possible Inclusion in the Architecture or Use Cases

E.1. Introduction

The posture of an endpoint is the status of the endpoint with respect to the security policies and risk models of the organization.

A system administrator needs to be able to determine which elements of an endpoint have a security problem and which do not conform the organization's security policies. The CIO needs to be able to

determine whether endpoints have security postures that conform to the organization's policies to ensure that the organization is complying with its fiduciary and regulatory responsibilities. The regulator or auditor needs to be able to assess the level of due diligence being achieved by an organization to ensure that all regulations and due diligence expectations are being met. The operator needs to understand which assets have deviated from organizational policies so that those assets can be remedied.

Operators will focus on which endpoints are composed of specific assets with problems. CIO and auditors need a characterization of how an organization is performing as a whole to manage the posture of its endpoints. All of these actors need deployed capabilities that implement security automation standards in the form of data formats, interfaces, and protocols to be able to assess, in a timely and secure fashion, all assets on all endpoints within their enterprise. This information model provides a basis to identify the desirable characteristics of data models to support these scenarios. Other SACM specifications, such as the SACM Architecture, will describe the potential components of an interoperable system solution based on the SACM information model to address the requirements for scalability, timeliness, and security.

E.2. Core Principles

This information model is built on the following core principles:

- Collection and Evaluation are separate tasks.

- Collection and Evaluation can be performed on the endpoint, at a local server that communicates directly with the endpoint, or based on data queried from a back end data store that does not communicate directly with any endpoints.

- Every entity (human or machine) that notifies, queries, or responds to any guidance, collection, or evaluator must have a way of identifying itself and/or presenting credentials. Authentication is a key step in all of the processes, and while needed to support the business processes, information needs to support authentication are not highlighted in this information model. There is already a large amount of existing work that defines information needs for authentication.

- o Policies are reflected in guidance for collection, evaluation, and reporting.

- o Guidance will often be generated by humans or through the use of transformations on existing automation data. In some cases,

guidance will be generated dynamically based on shared information or current operational needs. As guidance is created it will be published to an appropriate guidance data store allowing guidance to be managed in and retrieved from convenient locations.

- o Operators of a continuous monitoring or security automation system will need to make decisions when defining policies about what guidance to use or reference. The guidance used may be directly associated with policy or may be queried dynamically based on associated metadata.

- o Guidance can be gathered from multiple data stores. It may be retrieved at the point of use or may be packaged and forwarded for later use. Guidance may be retrieved in event of a collection or evaluation trigger or it may be gathered ahead of time and stored locally for use/reference during collection and evaluation activities.

E.3. Architecture Assumptions

This information model will focus on WHAT information needs to be exchanged to support the business process areas. The architecture document is the best place to represent the HOW and the WHERE this information is used. In an effort to ensure that the data models derived from this information model scale to the architecture, four core architectural components need to be defined. They are producers, consumers, capabilities, and repositories. These elements are defined as follows:

- o Producers (e.g., Evaluation Producer) collect, aggregate, and/or derive information items and provide them to consumers. For this model there are Collection, Evaluation, and Results Producers. There may or may not be Guidance Producers.

- o Consumers (e.g., Collection Consumer) request and/or receive information items from producers for their own use. For this model there are Collection, Evaluation, and Results Consumers. There may or may not be Guidance Consumers.

- o Capabilities (e.g., Posture Evaluation Capability) take the input from one or more producers and perform some function on or with that information. For this model there are Collection Guidance, Collection, Evaluation Guidance, Evaluation, Reporting Guidance, and Results Reporting Capabilities.

- o Repositories (e.g., Enterprise Repository) store information items that are input to or output from Capabilities, Producers, and

Consumers. For this model we refer to generic Enterprise and Guidance Repositories.

Information that needs to be communicated by or made available to any of these components will be specified in each of the business process areas.

In the most trivial example, illustrated in Figure 13, Consumers either request information from, or are notified by, Producers.

```
+----------+     Request     +----------+
|          <-----------------+          |
| Producer |                 | Consumer |
|          +----------------->          |
+----------+    Response     +----------+

+----------+                 +----------+
|          |     Notify      |          |
| Producer +-----------------> Consumer |
|          |                 |          |
+----------+                 +----------+
```

Figure 13: Example Producer/Consumer Interactions

As illustrated in Figure 14, writing and querying from data repositories are a way in which this interaction can occur in an asynchronous fashion.

```
+----------+                +----------+
|          |                |          |
| Producer |                | Consumer |
|          |                |          |
+-----+----+                +----^-----+
      |                          |
Write |     +------------+       | Query
      |     |            |       |
      +-----> Repository +-------+
            |            |
            +------------+
```

Figure 14: Producer/Consumer Repository Interaction

To perform an assessment, these elements are chained together. The diagram below is illustrative of this and process, and is meant to demonstrate WHAT basic information exchanges need to occur, while trying to maintain flexibility in HOW and WHERE they occur.

For example:

- o the collection capability can reside on the endpoint or not.
- o the collection producer can be part of the collection capability or not.
- o a repository can be directly associated with a producer and/or an evaluator or stand on its own.
- o there can be multiple "levels" of producers and consumers.

```
                               +-------------+
                               |Evaluation   |
              +-------------+  |Guidance     +--+
              |Endpoint     |  |Capability   |  |
      +-------+             |  +-------------+  |
      |       |             |                   |
      |       +-------+-----+             +-----v-------+
      | Collection    |                   |Evaluation   |
    +-> Capability +--+--------+          |Capability   |
    | |            |Collection |    +-----------+   +----------+
    | +------------+Producer   |    |           |---|          |
    |              |           |    |Collection |   |Evaluation|
    |              |           |    |Consumer   |   |Producer  |
    |              +----+------+    +----^------+   +---+------+
   ++---------+         |                |              |
   |Collection|   +-----v------+     +---+--------+     |
   |Guidance  |   |            |     |Collection  |     |
   |Capability|   |Collection  |     |Producer    |     |
   |          |   |Consumer    |-----|            |     |
   +----------+   +------------+     +------------+     |
                             | Collection |             |
                             | Repository |             |
                             +------------+             |
                                                        |
       +--------------+           +---------------+     |
       |Evaluation    |           |Evaluation     |     |
       |Results       |           |Consumer       <-----+
       |Producer      |-----------|               |
       +-----+--------+           +---------------+
             |     |Results Reporting|
             |     |Capability       |
             |     +------------^----+
             |                  |
       +-----v--------+    +----+------+
       |Evaluation    |    |Reporting  |
       |Results       |    |Guidance   |
       |Consumer      |    |Repository |
       +---+----------+    +-----------+ +-------------+
           |                             | Results     |
           +-----------------------------> Repository  |
                                         |             |
                                         +-------------+
```

Figure 15: Producer/Consumer Complex Example

This illustrative example in Figure 15 provides a set of information exchanges that need to occur to perform a posture assessment. The rest of this information model is using this set of exchanges based

on these core architectural components as the basis for determining information elements.

Appendix F. Text for Possible Inclusion in the Requirements Draft

F.1. Problem Statement

Scalable and sustainable collection, expression, and evaluation of endpoint information is foundational to SACM's objectives. To secure and defend one's network one must reliably determine what devices are on the network, how those devices are configured from a hardware perspective, what software products are installed on those devices, and how those products are configured. We need to be able to determine, share, and use this information in a secure, timely, consistent, and automated manner to perform endpoint posture assessments.

F.2. Problem Scope

The goal of this iteration of the information model is to define the information needs for an organization to effectively monitor the endpoints operating on their network, the software installed on those endpoints, and the configuration of that software. Once we have those three business processes in place, we can identify vulnerable endpoints in a very efficient manner.

The four business process areas represent a large set of tasks that support endpoint posture assessment. In an effort to address the most basic and foundational needs, we have also narrowed down the scope inside of each of the business processes to a set of defined tasks that strive to achieve specific results in the operational environment and the organization. These tasks are:

1. Define the assets. This is what we want to know about an asset. For instance, organizations will want to know what software is installed and its many critical security attributes such as patch level.

2. Resolve what assets compose an endpoint. This requires populating the data elements and attributes needed to exchange information pertaining to the assets composing an endpoint.

3. Express what expected values for the data elements and attributes need to be evaluated against the actual collected instances of asset data. This is how an organization can express its policy for an acceptable data element or attribute value. A system administrator can also identify specific data elements and

attributes that represent problems, such as vulnerabilities, that need to be detected on an endpoint.

4. Evaluate the collected instances of the asset data against those expressed in the policy.

5. Report the results of the evaluation.

Appendix G. Text With No Clear Home Yet

G.1. Operations

Operations that may be carried out the proposed SACM Information Model are:

o Publish data: Security information is made available in the information model when a component publishes data to it.

o Subscribe to data: A component seeking to consume an on-going stream of security information "subscribes" to such data from the information model.

o Query: This operation enables a component to request a specific set of security data regarding a specific asset (such as a specific user endpoint).

The subscribe capability will allow SACM components to monitor for selected security-related changes in the graph data store without incurring the performance penalties associated with polling for such changes.

G.1.1. Generalized Workflow

The proposed SACM Information Model would be most commonly used with a suitable transport protocol for collecting and distributing security data across appropriate network platforms and endpoints. The information model is transport agnostic and can be used with its native transport provided by IF-MAP or by other data transport protocols such as the recently proposed XMPP-Grid.

1. A Posture Assessment Information Consumer (Consumer) establishes connectivity and authorization with the transport fabric.

2. A Posture Assessment Information Provider (Provider) with a source of security data requests connection to the transport fabric.

3. Transport fabric authenticates and establishes authorized privileges (e.g. privilege to publish and/or subscribe to security data) for the requesting components.

4. Components may either publish security data, subscribe to security data, query for security data, or any combination of these operations.

Any component sharing information - either as Provider or Consumer - may do so on a one-to-one, one-to-many and/or many-to-many meshed basis.

G.2. From Information Needs to Information Elements

The previous sections highlighted information needs for a set of management process areas that use posture assessment to achieve organizational security goals. A single information need may be made up of multiple information elements. Some information elements may be required for two different process areas, resulting in two different requirements. In an effort to support the main idea of collect once and reuse the data to support multiple processes, we try to define a singular set of information elements that will support all the associated information needs.

G.3. Information Model Elements

TODO: Kim to pull up relevant content into section 4 / Elements

Traditionally, one would use the SACM architecture to define interfaces that required information exchanges. Identified information elements would then be based on those exchanges. Because the SACM architecture document is still in the personal draft stage, this information model uses a different approach to the identification of information elements. First it lists the four main endpoint posture assessment activities. Then it identifies management process areas that use endpoint posture assessment to achieve organizational security objectives. These process areas were then broken down into operations that mirrored the typical workflow from the SACM Use Cases draft [RFC7632]. These operations identify architectural components and their information needs. In this section, information elements derived from those information needs are mapped back to the four main activities listed above.

The original liaison statement [IM-LIAISON-STATEMENT-NIST] requested contributions for the SACM information model in the four areas described below. Based on the capabilities defined previously in this document, the requested areas alone do not provide a sufficient enough categorization of the necessary information model elements.

The following sub-sections directly address the requested areas as follows:

1. Endpoint Identification

   A. Appendix G.3.1 Asset Identifiers: Describes identification of many different asset types including endpoints.

2. Endpoint Characterization

   A. Appendix G.3.3 Endpoint characterization: This directly maps to the requested area.

3. Endpoint Attribute Expression/Representation

   A. Appendix G.3.4 Posture Attribute Expression: This corresponds to the first part of "Endpoint Attribute Expression/ Representation."

   B. Appendix G.3.5 Actual Value Representation: This corresponds to the second part of "Endpoint Attribute Expression/ Representation."

4. Policy evaluation expression and results reporting

   A. Appendix G.3.6 Evaluation Guidance: This corresponds to the first part of "Policy evaluation expression and results reporting."

   B. Appendix G.3.7 Evaluation Result Reporting: corresponds to the second part of "Policy evaluation expression and results reporting."

Additionally, Appendix G.3.2 Other Identifiers: describes other important identification concepts that were not directly requested by the liaison statement.

Per the liaison statement, each subsection references related work that provides a basis for potential data models. Some analysis is also included for each area of related work on how directly applicable the work is to the SACM efforts. In general, much of the related work does not fully address the general or use case-based requirements for SACM, but they do contain some parts that can be used as the basis for data models that correspond to the information model elements. In these cases additional work will be required by the WG to adapt the specification. In some cases, existing work can largely be used in an unmodified fashion. This is also indicated in the analysis. Due to time constraints, the work in this section is

very biased to previous work supported by the authors and does not reflect a comprehensive listing. An attempt has been made where possible to reference existing IETF work. Additional research and discussion is needed to include other related work in standards and technology communities that could and should be listed here. The authors intend to continue this work in subsequent revisions of this draft.

Where possible when selecting and developing data models in support of these information model elements, extension points and IANA registries SHOULD be used to provide for extensibility which will allow for future data models to be addressed.

G.3.1. Asset Identifiers

In this context an "asset" refers to "anything that has value to an organization" (see [NISTIR-7693]). This use of the term "asset" is broader than the current definition in [I-D.ietf-sacm-terminology]. To support SACM use cases, a number of different asset types will need to be addressed. For each type of asset, one or more type of asset identifier will be needed for use in establishing contextual relationships within the SACM information model. The following asset types are referenced or implied by the SACM use cases:

Endpoint: Identifies an individual endpoint for which posture is collected and evaluated.

Hardware: Identifies a given type of hardware that may be installed within an endpoint.

Software: Identifies a given type of software that may be installed within an endpoint.

Network: Identifies a network for which a given endpoint may be connected or request a connection to.

Organization: Identifies an organizational unit.

Person: Identifies an individual, often within an organizational context.

G.3.1.1. Related Work

G.3.1.1.1. Asset Identification

The Asset Identification specification [NISTIR-7693] is an XML-based data model that "provides the necessary constructs to uniquely identify assets based on known identifiers and/or known information

about the assets." Asset identification plays an important role in an organization's ability to quickly correlate different sets of information about assets. The Asset Identification specification provides the necessary constructs to uniquely identify assets based on known identifiers and/or known information about the assets. Asset Identification provides a relatively flat and extensible model for capturing the identifying information about a one or more assets, and also provides a way to represent relationships between assets.

The model is organized using an inheritance hierarchy of specialized asset types/classes (see Figure 16), providing for extension at any level of abstraction. For a given asset type, a number of properties are defined that provide for capturing identifying characteristics and the referencing of namespace qualified asset identifiers, called "synthetic IDs."

The following figure illustrates the class hierarchy defined by the Asset Identification specification.

```
      asset
       +-it-asset
       | +-circuit
       | +-computing-device
       | +-database
       | +-network
       | +-service
       | +-software
       | +-system
       | +-website
       +-data
       +-organization
       +-person
```

Figure 16: Asset Identification Class Hierarchy

This table presents a mapping of notional SACM asset types to those asset types provided by the Asset Identification specification.

SACM Asset Type	Asset Identification Type	Notes
Endpoint	computing-device	This is not a direct mapping since a computing device is not required to have network-connectivity. Extension will be needed to define a directly aligned endpoint asset type.
Hardware	Not Applicable	The concept of hardware is not addressed by the asset identification specification. An extension can be created based on the it-asset class to address this concept.
Software	software	Direct mapping.
Network	network	Direct mapping.
Organization	organization	Direct mapping.
Person	person	Direct mapping.

Table 1: Mapping of SACM to Asset Identification Asset Types

This specification has been adopted by a number of SCAP validated products. It can be used to address asset identification and categorization needs within SACM with minor modification.

G.3.1.2. Endpoint Identification

An unique name for an endpoint. This is a foundational piece of information that will enable collected posture attributes to be related to the endpoint from which they were collected. It is important that this name either be created from, provide, or be associated with operational information (e.g., MAC address, hardware certificate) that is discoverable from the endpoint or its communications on the network. It is also important to have a method of endpoint identification that can persist across network sessions to allow for correlation of collected data over time.

G.3.1.2.1. Related Work

The previously introduced asset identification specification (see Appendix G.3.1.1.1 provides a basis for endpoint identification using the "computing-device" class. While the meaning of this class is broader than the current definition of an endpoint in the SACM terminology [I-D.ietf-sacm-terminology], either that class or an appropriate sub-class extension can be used to capture identification information for various endpoint types.

G.3.1.3. Software Identification

A unique name for a unit of installable software. Software names should generally represent a unique release or installable version of software. Identification approaches should allow for identification of commercially available, open source, and organizationally developed custom software. As new software releases are created, a new software identifier should be created by the releasing party (e.g., software creator, publisher, licensor). Such an identifier is useful to:

- o Relate metadata that describes the characteristics of the unit of software, potentially stored in a repository of software information. Typically, the software identifier would be used as an index into such a repository.

- o Indicate the presence of the software unit on a given endpoint.

- o To determine what endpoints are the targets for an assessment based on what software is installed on that endpoint.

- o Define guidance related to a software unit that represents collection, evaluation, or other automatable policies.

In general, an extensible method of software identification is needed to provide for adequate coverage and to address legacy identification approaches. Use of an IANA registry supporting multiple software identification methods would be an ideal way forward.

G.3.1.3.1. Related Work

While we are not aware of a one-size-fits-all solution for software identification, there are two existing specifications that should be considered as part of the solution set. They are described in the following subsections.

G.3.1.3.1.1. Common Platform Enumeration

G.3.1.3.1.1.1. Background

The Common Platform Enumeration (CPE) [CPE-WEBSITE] is composed of a family of four specification that are layered to build on lower-level functionality. The following describes each specification:

1. CPE Naming: A standard machine-readable format [NISTIR-7695] for encoding names of IT products and platforms. This defines the notation used to encode the vendor, software name, edition, version and other related information for each platform or product. With the 2.3 version of CPE, a second, more advanced notation was added to the original colon-delimited notation for CPE naming.

2. CPE Matching: A set of procedures [NISTIR-7696] for comparing names. This describes how to compare two CPE names to one another. It describes a logical method that ensures that automated systems comparing two CPE names would arrive at the same conclusion.

3. CPE Applicability Language: An XML-based language [NISTIR-7698] for constructing "applicability statements" that combine CPE names with simple logical operators.

4. CPE Dictionary: An XML-based catalog format [NISTIR-7697] that enumerates CPE Names and associated metadata. It details how to encode the information found in a CPE Dictionary, thereby allowing multiple organizations to maintain compatible CPE Dictionaries.

The primary use case of CPE is for exchanging software inventory data, as it allows the usage of unique names to identify software platforms and products present on an endpoint. The NIST currently maintains and updates a dictionary of all agreed upon CPE names, and is responsible for ongoing maintenance of the standard. Many of the names in the CPE dictionary have been provided by vendors and other 3rd-parties.

While the effort has seen wide adoption, most notably within the US

Government, a number of critical flaws have been identified. The most critical issues associated with the effort are:

- o Because there is no requirement for vendors to publish their own, official CPE names, CPE necessarily requires one or more organizations for curation. This centralized curation requirement ensures that the effort has difficulty scaling.

- o Not enough primary source vendors provide platform and product naming information. As a result, this pushes too much of the effort out onto third-party groups and non-authoritative organizations. This exacerbates the ambiguity in names used for identical platforms and products and further reduces the utility of the effort.

G.3.1.3.1.1.2. Applicability to Software Identification

The Common Platform Enumeration (CPE) Naming specification version 2.3 defines a scheme for human-readable standardized identifiers of hardware and software products.

CPE names are the identifier format for software and hardware products used in SCAP 1.2 and is currently adopted by a number of SCAP product vendors.

CPE names can be directly referenced in the asset identification software class (see Appendix G.3.1.1.1.)

Although relevant, CPE has an unsustainable maintenance "tail" due to the need for centralized curation and naming-consistency enforcement. Its mention in this document is to support the historic inclusion of CPE as part of SCAP and implementation of this specification in a number of security processes and products. Going forward, software identification (SWID) tags are recommended as a replacement for CPE. To this end, work has been started to align both efforts to provide translation for software units identified using SWID tags to CPE Names. This translation would allow tools that currently use CPE-based identifiers to map to SWID identifiers during a transition period.

G.3.1.3.1.2. Software Identification (SWID) Tags

The software identification tag specification [ISO.19770-2] is an XML-based data model that is used to describe a unit of installable software. A SWID tag contains data elements that:

- o Identify a specific unit of installable software,
- o Enable categorization of the software (e.g., edition, bundle),
- o Identification and hashing of software artifacts (e.g., executables, shared libraries),
- o References to related software and dependencies, and
- o Inclusion of extensible metadata.

SWID tags can be associated with software installation media, installed software, software updates (e.g., service packs, patches, hotfixes), and redistributable components. SWID tags also provide for a mechanism to relate these concepts to each other. For example, installed software can be related back to the original installation media, patches can be related to the software that they patch, and software dependencies can be described for required redistributable components. SWID tags are ideally created at build-time by the software creator, publisher or licensor; are bundled with software installers; and are deployed to an endpoint during software installation.

SWID tags should be considered for two primary uses:

1. As the data format for exchanging descriptive information about software products, and

2. As the source of unique identifiers for installed software.

In addition to usage for software identification, a SWID tag can provide the necessary data needed to target guidance based on included metadata, and to support verification of installed software and software media using cryptographic hashes. This added information increases the value of using SWID tags as part of the larger security automation and continuous monitoring solution space.

G.3.1.4. Hardware Identification

Due to the time constraints, research into information elements and related work for identifying hardware is not included in this revision of the information model.

G.3.2. Other Identifiers

In addition to identifying core asset types, it is also necessary to have stable, globally unique identifiers to represent other core concepts pertaining to posture attribute collection and evaluation. The concept of "global uniqueness" ensures that identifiers provided by multiple organization do not collide. This may be handled by a number of different mechanisms (e.g., use of namespaces).

G.3.2.1. Platform Configuration Item Identifier

A name for a low-level, platform-dependent configuration mechanism as determined by the authoritative primary source vendor. New identifiers will be created when the source vendor makes changes to the underlying platform capabilities (e.g., adding new settings, replacing old settings with new settings). When created each

identifier should remain consistent with regards to what it represents. Generally, a change in meaning would constitute the creation of a new identifier.

For example, if the configuration item is for "automatic execution of code", then the platform vendor would name the low-level mechanism for their platform (e.g., autorun for mounted media).

G.3.2.1.1. Related Work

G.3.2.1.1.1. Common Configuration Enumeration

The Common Configuration Enumeration (CCE) [CCE] is an effort managed by NIST. CCE provides a unique identifier for platform-specific configuration items that facilitates fast and accurate correlation of configuration items across multiple information sources and tools. CCE does this by providing an identifier, a human readable description of the configuration control, parameters needed to implement the configuration control, various technical mechanisms that can be used to implement the configuration control, and references to documentation that describe the configuration control in more detail.

By vendor request, NIST issues new blocks of CCE identifiers. Vendors then populate the required fields and provided the details back to NIST for publication in the "CCE List", a consolidated listing of assigned CCE identifiers and associated data. Many vendors also include references to these identifiers in web pages, SCAP content, and prose configuration guides they produce.

CCE the identifier format for platform specific configuration items in SCAP and is currently adopted by a number of SCAP product vendors.

While CCE is largely supported as a crowd-sourced effort, it does rely on a central point of coordination for assignment of new CCE identifiers. This approach to assignment requires a single organization, currently NIST, to manage allocations of CCE identifiers which doesn't scale well and introduces sustainability challenges for large volumes of identifier assignment. If this approach is used going forward by SACM, a namespaced approach is recommended for identifier assignment that allows vendors to manage their own namespace of CCE identifiers. This change would require additional work to specify and implement.

G.3.2.1.1.2. Open Vulnerability and Assessment Language

G.3.2.1.1.2.1. Background

The Open Vulnerability and Assessment Language (OVAL(R)) is an XML schema-based data model developed as part of a public-private information security community effort to standardize how to assess and report upon the security posture of endpoints. OVAL provides an established framework for making assertions about an endpoint's posture by standardizing the three main steps of the assessment process:

1. representing the current endpoint posture;

2. analyzing the endpoint for the presence of the specified posture; and

3. representing the results of the assessment.

OVAL facilitates collaboration and information sharing among the information security community and interoperability among tools. OVAL is used internationally and has been implemented by a number of operating system and security tools vendors.

The following figure illustrates the OVAL data model.

```
                                             +------------+
                       +-----------------+   | Variables  |
                       | Common          <---+            |
              +-------->                 |   +------------+
              |        |                 |   +------------+
              |        |                 <---+ Directives |
              |        +--------^----^---+   |            |
              |                 |    |       +--------+---+
              |                 |    +-----+          |
              |                 |          |          |
              |        +--------+--------+ |          |
              |        | System          | |          |
              |        | Characteristics | |          |
       +------+------+ |                 | | +--------v---+
       | Definitions | |                 | | | Results    |
       |             | +--------^--------+ +-+            |
       |             |          |            |            |
       |             |          +------------+            |
       +------^------+                       +-------+----+
              |                                      |
              +--------------------------------------+
```

Note: The direction of the arrows indicate a model dependency

Figure 17: The OVAL Data Model

The OVAL data model [OVAL-LANGUAGE], visualized in Figure 17, is composed of a number of different components. The components are:

o Common: Constructs, enumerations, and identifier formats that are used throughout the other model components.

o Definitions: Constructs that describe assertions about system state. This component also includes constructs for internal variable creation and manipulation through a variety of functions. The core elements are:

  * Definition: A collection of logical statements that are combined to form an assertion based on endpoint state.

  * Test(platform specific): A generalized construct that is extended in platform schema to describe the evaluation of expected against actual state.

  * Object(platform specific): A generalized construct that is extended in platform schema to describe a collectable aspect of endpoint posture.

  * State(platform specific): A generalized construct that is extended in platform schema to describe a set of criteria for evaluating posture attributes.

o Variables: Constructs that allow for the parameterization of the elements used in the Definitions component based on externally provided values.

o System Characteristics: Constructs that represent collected posture from one or more endpoints. This element may be embedded with the Results component, or may be exchanged separately to allow for separate collection and evaluation. The core elements of this component are:

  * CollectedObject: Provides a mapping of collected Items to elements defined in the Definitions component.

  * Item(platform specific): A generalized construct that is extended in platform schema to describe specific posture attributes pertaining to an aspect of endpoint state.

o Results: Constructs that represent the result of evaluating expected state (state elements) against actual state (item elements). It includes the true/false evaluation result for each evaluated Definition and Test. Systems characteristics are embedded as well to provide low-level posture details.

- o Directives: Constructs that enable result reporting detail to be declared, allowing for result production to be customized.

End-user organizations and vendors create assessment guidance using OVAL by creating XML instances based on the XML schema implementation of the OVAL Definitions model. The OVAL Definitions model defines a structured identifier format for each of the Definition, Test, Object, State, and Item elements. Each instantiation of these elements in OVAL XML instances are assigned a unique identifier based on the specific elements identifier syntax. These XML instances are used by tools that support OVAL to drive collection and evaluation of endpoint posture. When posture collection is performed, an OVAL Systems Characteristics XML instance is generated based on the collected posture attributes. When this collected posture is evaluated, an OVAL Result XML instance is generated that contains the results of the evaluation. In most implementations, the collection and evaluation is performed at the same time.

Many of the elements in the OVAL model (i.e., Test, Object, State, Item) are abstract, requiring a platform-specific schema implementation, called a "Component Model" in OVAL. These platform schema implementations are where platform specific posture attributes are defined. For each aspect of platform posture a specialized OVAL Object, which appears in the OVAL Definitions model, provides a format for expressing what posture attribute data to collect from an endpoint through the specification of a datatype, operation, and value(s) on entities that uniquely identify a platform configuration item. For example, a hive, key, and name is used to identify a registry key on a Windows endpoint. Each specialized OVAL Object has a corresponding specialized State, which represents the posture attributes that can be evaluated, and an Item which represents the specific posture attributes that can be collected. Additionally, a specialized Test exists that allows collected Items corresponding to a CollectedObject to be evaluated against one or more specialized States of the same posture type.

The OVAL language provides a generalized approach suitable for posture collection and evaluation. While this approach does provide for a degree of extensibility, there are some concerns that should be addressed in order to make OVAL a viable basis for SACM's use. These concerns include:

- o Platform Schema Creation and Maintenance: In OVAL platform schema, the OVAL data model maintains a tight binding between the Test, Object, State, and Item elements used to assess an aspect of endpoint posture. Creating a new platform schema or adding a new posture aspect to an existing platform schema can be a very labor intensive process. Doing so often involves researching and understanding system APIs and can be prone to issues with inconsistency within and between platforms. To simplify platform schema creation and maintenance, the model needs to be evolved to generalize the Test, Object, and State elements, requiring only the definition of an Item representation.

- o Given an XML instance based on the Definitions model, it is not clear in the specification how incremental collection and evaluation can occur. Because of this, typically, OVAL assessments are performed on a periodic basis. The OVAL specification needs to be enhanced to include specifications for performing event-based and incremental assessment in addition to full periodic collection.

- o Defining new functions for manipulating variable values is current handled in the Definitions schema. This requires revision to the core language to add new functions. The OVAL specification needs to be evolved to provide for greater extensibility in this area, allowing extension schema to define new functions.

- The current process for releasing a new version of OVAL, bundle releases of the core language with release of community recognized platform schema. The revision processes for the core and platform schema need to be decoupled. Each platform schema should use some mechanism to declare which core language version it relies on.

If adopted by SCAM, these issues will need to be addressed as part of the SCAM engineering work to make OVAL more broadly adoptable as a general purpose data model for posture collection and evaluation.

G.3.2.1.1.2.2. Applicability to Platform Configuration Item Identification

Each OVAL Object is identified by a globally unique identifier. This globally unique identifier could be used by the SACM community to identify platform-specific configuration items and at the same time serve as collection guidance. If used in this manner, OVAL Objects would likely need to undergo changes in order to decouple it from evaluation guidance and to provide more robust collection capabilities to support the needs of the SACM community.

G.3.2.2. Configuration Item Identifier

An identifier for a high-level, platform-independent configuration control. This identification concept is necessary to allow similar configuration item concepts to be comparable across platforms. For example, a configuration item might be created for the minimum password length configuration control, which may then have a number of different platform-specific configuration settings. Without this type of identification, it will be difficult to perform evaluation of expected versus actual state in a platform-neutral way.

High-level configuration items tend to change much less frequently than the platform-specific configuration items (see Appendix G.3.2.1) that might be associated with them. To provide for the greatest amount of sustainability, collections of configuration item identifiers are best defined by specific communities of interest, while platform-specific identifiers are best defined by the source vendor of the platform. Under this model, the primary source vendors would map their platform-specific configuration controls to the appropriate platform-independent item allowing end-user organizations to make use of these relationships.

To support different communities of interest, it may be necessary to support multiple methods for identification of configuration items

and for associating related metadata. Use of an IANA registry supporting multiple configuration item identification methods would be an ideal way forward. To the extent possible, a few number of configuration item identification approaches is desirable, to maximize the update by vendors who would be maintain mapping of platform-specific configuration identifiers to the more general platform-neutral configuration identifiers.

G.3.2.2.1. Related Work

G.3.2.2.1.1. Control Correlation Identifier

The Control Correlation Identifier (CCI) [CCI] is developed and managed by the United States Department of Defense (US-DoD) Defense Information Systems Agency (DISA). According to their website, CCI "provides a standard identifier and description for each of the singular, actionable statements that comprise an information assurance (IA) control or IA best practice. CCI bridges the gap between high-level policy expressions and low-level technical implementations. CCI allows a security requirement that is expressed in a high-level policy framework to be decomposed and explicitly associated with the low-level security setting(s) that must be assessed to determine compliance with the objectives of that specific security control. This ability to trace security requirements from their origin (e.g., regulations, IA frameworks) to their low-level implementation allows organizations to readily demonstrate compliance to multiple IA compliance frameworks. CCI also provides a means to objectively roll-up and compare related compliance assessment results across disparate technologies."

It is recommended that this approach be analysed as a potential candidate for use as a configuration item identifier method.

Note: This reference to CCI is for informational purposes. Since the editors do not represent DISA's interests, its inclusion in this document does not indicate the presence or lack of desire to contribute aspects of this effort to SACM.

G.3.2.2.1.2. A Potential Alternate Approach

There will likely be a desire by different communities to create different collections of configuration item identifiers. This fracturing may be caused by:

o Different requirements for levels of abstraction,

o Varying needs for timely maintenance of the collection, and

o Differing scopes of technological needs.

Due to these and other potential needs, it will be difficult to standardize around a single collection of configuration identifiers. A workable solution will be one that is scalable and usable for a broad population of end-user organizations. An alternate approach that should be considered is the definition of data model that contains a common set of metadata attributes, perhaps supported by an extensible taxonomy, that can be assigned to platform-specific configuration items. If defined at a necessary level of granularity, it may be possible to query collections of platform-specific configuration items provided by vendors to create groupings at various levels of abstractions. By utilizing data provided by vendors, technological needs and the timeliness of information can be addressed based on customer requirements.

SACM should consider this and other approaches to satisfy the need for configuration item roll-up in a way that provides the broadest benefit, while achieving a sensible degree of scalability and sustainability.

G.3.2.3. Vulnerability Identifier

An unique name for a known software flaw that exists in specific versions of one or more units of software. One use of a vulnerability identifier in the SACM context is to associate a given flaw with the vulnerable software using software identifiers. For this reason at minimum, software identifiers should identify a software product to the patch or version level, and not just to the level that the product is licensed.

G.3.2.3.1. Related Work

G.3.2.3.1.1. Common Vulnerabilities and Exposures

Common Vulnerabilities and Exposures (CVE) [CVE-WEBSITE] is a MITRE led effort to assign common identifiers to publicly known security vulnerabilities in software to facilitate the sharing of information related to the vulnerabilities. CVE is the industry standard by which software vendors, tools, and security professionals identify vulnerabilities and could be used to address SACM's need for a vulnerability identifier.

G.3.3. Endpoint characterization

Target when policies (collection, evaluated, guidance) apply

Collection can be used to further characterize

Also human input

Information required to characterize an endpoint is used to determine what endpoints are the target of a posture assessment. It is also used to determine the collection, evaluation, and/or reporting policies and the associated guidance that apply to the assessment. Endpoint characterization information may be populated by:

- o A manual input process and entered into records associated with the endpoint, or
- o Using information collected and evaluated by an assessment.

Regardless of the method of collection, it will be necessary to query and exchange endpoint characterization information as part of the assessment planning workflow.

G.3.3.1. Related Work

G.3.3.1.1. Extensible Configuration Checklist Description Format

G.3.3.1.1.1. Background

The Extensible Configuration Checklist Description Format (XCCDF) is a specification that provides an XML-based format for expressing security checklists. The XCCDF 1.2 specification is published by International Organization for Standardization (ISO) [ISO.18180]. XCCDF contains multiple components and capabilities, and various components align with different elements of this information model.

This specification was originally published by NIST [NISTIR-7275]. When contributed to ISO Joint Technical Committee 1 (JTC 1), a comment was introduced indicating an interest in the IETF becoming the maintenance organization for this standard. If the SACM working group is interested in taking on engineering work pertaining to XCCDF, a contribution through a national body can be made to create a ballot resolution for transition of this standard to the IETF for maintenance.

G.3.3.1.1.2. Applicability to Endpoint characterization

The target component of XCCDF provides a mechanism for capturing characteristics about an endpoint including the fully qualified domain name, network address, references to external identification information (e.g. Asset Identification), and is extensible to support other useful information (e.g. MAC address, globally unique identifier, certificate, etc.). XCCDF may serve as a good starting

point for understanding the types of information that should be used to identify an endpoint.

G.3.3.1.2. Asset Reporting Format

G.3.3.1.2.1. Background

The Asset Reporting Format (ARF) [NISTIR-7694] is a data model to express information about assets, and the relationships between assets and reports. It facilitates the reporting, correlating, and fusing of asset information within and between organizations. ARF is vendor and technology neutral, flexible, and suited for a wide variety of reporting applications.

There are four major sub-components of ARF:

- o Asset: The asset component element includes asset identification information for one or more assets. It simply houses assets independent of their relationships to reports. The relationship section can then link the report section to specific assets.

- o Report: The report component element contains one or more asset reports. An asset report is composed of content (or a link to content) about one or more assets.

- o Report-Request: The report-request component element contains the asset report requests, which can give context to asset reports captured in the report section. The report-request section simply houses asset report requests independent of the report which was subsequently generated.

- o Relationship: The relationship component element links assets, reports, and report requests together with well-defined relationships. Each relationship is defined as {subject} {predicate} {object}, where {subject} is the asset, report request, or report of interest, {predicate} is the relationship type being established, and {object} is one or more assets, report requests, or reports.

G.3.3.1.2.2. Relationship to Endpoint Characterization

For Endpoint Characterization, ARF can be used in multiple ways due to its flexibility. ARF supports the use of the Asset Identification specification (more in Appendix G.3.3.1.2.3) to embed the representation of one or more assets as well as relationships between those assets. It also allows the inclusion of report-requests, which can provide details on what data was required for an assessment.

ARF is agnostic to the data formats of the collected posture attributes and therefore can be used within the SACM Architecture to provide Endpoint Characterization without dictating data formats for the encoding of posture attributes. The embedded Asset Identification data model (see Appendix G.3.1.1.1) can be used to characterize one or more endpoints to allow targeting for collection, evaluation, etc. Additionally, the report-request model can dictate the type of reporting that has been requested, thereby providing context as to which endpoints the guidance applies.

G.3.3.1.2.3. Asset Identification

Described earlier

In the context of Endpoint Characterization, the Asset Identification data model could be used to encode information that identifies specific endpoints and/or classes of endpoints to which a particular assessment is relevant. The flexibility in the Asset Identification specification allows usage of various endpoint identifiers as defined by the SACM engineering work.

As stated in Appendix G.3.3.1.2.3, the Asset Identification specification is included within the Asset Reporting Framework (ARF) and therefore can be used in concert with that specification as well.

G.3.3.1.3. The CPE Applicability Language

CPE described earlier

Applicability in CPE is defined as an XML language [NISTIR-7698] for using CPE names to create applicability statements using logical expressions. These expressions can be used to applicability statements that can drive decisions about assets, whether or not to do things like collect data, report data, and execute policy compliance checks.

It is recommended that SACM evolve the CPE Applicability Language through engineering work to allow it to better fit into the security automation vision laid out by the Use Cases and Architecture for SACM. This should include de-coupling the identification part of the language from the logical expressions, making it such that the language is agnostic to the method by which assets are identified.

This will allow use of SWID, CPE Names, or other identifiers to be used, perhaps supported by an IANA registry of identifier types.

The other key aspect that should be evolved is the ability to make use of the Applicability Language against a centralized repository of collected posture attributes. The language should be able to make

applicability statements against previously collected posture attributes, such that an enterprise can quickly query the correct set of applicable endpoints in an automated and scalable manner.

G.3.4. Posture Attribute Expression

Discuss the catalog concept. Listing of things that can be chosen from. Things we can know about. Vendors define catalogs. Ways for users to get vendor-provided catalogs.

To support the collection of posture attributes, there needs to be a way for operators to identify and select from a set of platform-specific attribute(s) to collect. The same identified attributes will also need to be identified post-collection to associate the actual value of that attribute pertaining to an endpoint as it was configured at the time of the collection. To provide for extensibility, the need exists to support a variety of possible identification approaches. It is also necessary to enable vendors of software to provide a listing, or catalog, of the available posture attributes to operators that can be collected. Ideally, a federated approach will be used to allow organizations to identify the location for a repository containing catalogs of posture attributes provided by authoritative primary source vendors. By querying these repositories, operators will be able to acquire the appropriate listings of available posture attributes for their deployed assets. One or more posture attribute expressions are needed to support these exchanges.

G.3.4.1. Related Work

The ATOM Syndication Format [RFC4287] provides an extensible, flexible XML-based expression for organizing a collection of data feeds consisting of entries. This standard can be used to express one or more catalogs of posture attributes represented as data feeds. Groupings of posture attributes would be represented as entries. These entries could be defined using the data models described in the "Related Work" sections below. Additionally, this approach can also be used more generally for guidance repositories allowing other forms of security automation guidance to be exchanged using the same format.

G.3.4.2. Platform Configuration Attributes

A low-level, platform-dependent posture attribute as determined by the authoritative primary source vendor. Collection guidance will be derived from catalogs of platform specific posture attributes.

For example, a primary source vendor would create a platform-specific posture attribute that best models the posture attribute data for their platform.

G.3.4.2.1. Related Work

G.3.4.2.1.1. Open Vulnerability and Assessment Language

A general overview of OVAL was provided previously in Appendix G.3.2.1.1.2.1. The OVAL System Characteristics platform extension models provide a catalog of the posture attributes that can be collected from an endpoint. In OVAL these posture attributes are further grouped into logical constructs called OVAL Items. For example, the passwordpolicy_item that is part of the Windows platform extension groups together all the posture attributes related to passwords for an endpoint running Windows (e.g. maximum password age, minimum password length, password complexity, etc.). The various OVAL Items defined in the OVAL System Characteristics may serve as a good starting for the types of posture attribute data that needs to be collected from endpoints.

OVAL platform extension models may be shared using the ATOM Syndication Format.

G.3.4.2.1.2. Network Configuration Protocol and YANG Data Modeling Language

The Network Configuration Protocol (NETCONF) [RFC6241] defines a mechanism for managing and retrieving posture attribute data from network infrastructure endpoints. The posture attribute data that can be collected from a network infrastructure endpoint is highly extensible and can defined using the YANG modeling language [RFC6020]. Models exist for common datatypes, interfaces, and routing subsystem information among other subjects. The YANG modeling language may be useful in providing an extensible framework for the SACM community to define one or more catalogs of posture attribute data that can be collected from network infrastructure endpoints.

Custom YANG modules may also be shared using the ATOM Syndication Format.

G.3.4.2.1.3. Simple Network Management Protocol and Management Information Base Entry

The Simple Network Protocol (SNMP) [RFC3411] defines a protocol for managing and retrieving posture attribute data from endpoints on a network . The posture attribute data that can be collected of an

endpoint and retrieved by SNMP is defined by the Management Information Base (MIB) [RFC3418] which is hierarchical collection of information that is referenced using Object Identifiers . Given this, MIBs may provide an extensible way for the SACM community to define a catalog of posture attribute data that can be collected off of endpoints using SNMP.

MIBs may be shared using the ATOM Syndication Format.

G.3.5. Actual Value Representation

Discuss instance concept.

The actual value of a posture attribute is collected or published from an endpoint. The identifiers discussed previously provide names for the posture attributes (i.e., software or configuration item) that can be the subject of an assessment. The information items listed below are the actual values collected during the assessment and are all associated with a specific endpoint.

G.3.5.1. Software Inventory

A software inventory is a list of software identifiers (or content) associated with a specific endpoint. Software inventories are maintained in some organized fashion so that entities can interact with it. Just having software publish identifiers onto an endpoint is not enough, there needs to be an organized listing of all those identifiers associated with that endpoint.

G.3.5.1.1. Related Work

G.3.5.1.1.1. Asset Summary Reporting

The Asset Summary Reporting (ASR) specification [NISTIR-7848] provides a format for capturing summary information about one or more assets. Specifically, it provides the ability to express a collection of records from some defined data source and map them to some set of assets. As a result, this specification may be useful for capturing the software installed on an endpoint, its relevant attributes, and associating it with a particular endpoint.

G.3.5.1.1.2. Software Identification Tags

SWID tag were previously introduced in Appendix G.3.1.3.1.2. As stated before, SWID tags are ideally deployed to an endpoint during software installation. In the less ideal case, they may also be generated based on information retrieved from a proprietary software installation data store. At minimum, SWID tag must contain an

identifier for each unit of installed software. Given this, SWID tags may be a viable way for SACM to express detailed information about the software installed on an endpoint.

G.3.5.2. Collected Platform Configuration Posture Attributes

Configurations associated with a software instance associated with an endpoint

A list of the configuration posture attributes associated with the actual values collected from the endpoint during the assessment as required/expressed by any related guidance. Additionally, each configuration posture attribute is associated with the installed software instance it pertains to.

G.3.5.2.1. Related Work

G.3.5.2.1.1. Open Vulnerability and Assessment Language

A general overview of OVAL was provided previously in Appendix G.3.2.1.1.2.1. As mentioned earlier, the OVAL System Characteristics platform extensions provide a catalog of the posture attributes that can be collected and assessed in the form of OVAL Items. These OVAL Items also serve as a model for representing posture attribute data and associated values that are collected off an endpoint. Furthermore, the OVAL System Characteristics model provides a system_info construct that captures information that identifies and characterizes the endpoint from which the posture attribute data was collected. Specifically, it includes operating system name, operating system version, endpoint architecture, hostname, network interfaces, and an extensible construct to support arbitrary additional information that may be useful in identifying the endpoint in an enterprise such as information capture in Asset Identification constructs. The OVAL System Characteristics model could serve as a useful starting point for representing posture attribute data collected from an endpoint although it may need to undergo some changes to satisfy the needs of the SACM community.

G.3.5.2.1.2. NETCONF-Based Collection

Introduced earlier in Appendix G.3.4.2.1.2, NETCONF defines a protocol for managing and retrieving posture attribute data from network infrastructure endpoints. NETCONF provides the <get-config> and <get> operations to retrieve the configuration data, and configuration data and state data respectively from a network infrastructure endpoint. Upon successful completion of these operations, the current posture attribute data of the network infrastructure endpoint will be made available. NETCONF also

provides a variety of filtering mechanisms (XPath, subtree, content matching, etc.) to trim down the posture attribute data that is collected from the endpoint. Given that NETCONF is widely adopted by network infrastructure vendors, it may useful to consider this protocol as a standardized mechanism for collecting posture attribute data from network infrastructure endpoints.

As a side note, members of the OVAL Community have also developed a proposal to extend the OVAL Language to support the assessment of NETCONF configuration data <https://github.com/OVALProject/Sandbox/blob/master/x-netconf-definitions-schema.xsd>. The proposal leverages XPath to extract the posture attribute data of interest from the XML data returned by NETCONF. The collected posture attribute data can then be evaluated using OVAL Definitions and the results of the evaluation can be expressed as OVAL Results. While this proposal is not currently part of the OVAL Language, it may be worth considering.

G.3.5.2.1.3. SNMP-Based Collection

The SNMP, previously introduced in Appendix G.3.4.2.1.3, defines a protocol for managing and retrieving posture attribute data from endpoints on a network [RFC3411]. SNMP provides three protocol operations [RFC3416] (GetRequest, GetNextRequest, and GetBulkRequest) for retrieving posture attribute data defined by MIB objects. Upon successful completion of these operations, the requested posture attribute data of the endpoint will be made available to the requesting application. Given that SNMP is widely adopted by vendors, and the MIBs that define posture attribute data on an endpoint are highly extensible, it may useful to consider this protocol as a standardized mechanism for collecting posture attribute data from endpoints in an enterprise.

G.3.6. Evaluation Guidance

G.3.6.1. Configuration Evaluation Guidance

The evaluation guidance is applied by evaluators during posture assessment of an endpoint. This guidance must be able to reference or be associated with the following previously defined information elements:

o configuration item identifiers,

o platform configuration identifiers, and

o collected Platform Configuration Posture Attributes.

G.3.6.1.1. Related Work

G.3.6.1.1.1. Open Vulnerability and Assessment Language

A general overview of OVAL was provided previously in Appendix G.3.2.1.1.2.1. The OVAL Definitions model provides an extensible framework for making assertions about the state of posture attribute data collected from an endpoint. Guidance written against this model consists of one or more OVAL Tests, which can be logically combined, where each OVAL Test defines what posture attributes should be collected from an endpoint (as OVAL Objects) and optionally defines the expected state of the posture attributes (as OVAL States). While the OVAL Definitions model may be a useful starting point for evaluation guidance, it will likely require some changes to decouple collection and evaluation concepts to satisfy the needs of the SACM community.

G.3.6.1.1.2. XCCDF Rule

A general description of XCCDF was provided in Appendix G.3.3.1.1.1. As noted there, an XCCDF document represents a checklist of items against which a given endpoint's state is compared and evaluated. An XCCDF Rule represents one assessed item in this checklist. A Rule contains both a prose description of the assessed item (either for presentation to the user in a tool's user interface, or for rendering into a prose checklist for human consumption) and can also contain instructions to support automated evaluation of the assessed item, if such automated assessment is possible. Automated assessment instructions can be provided either within the XCCDF Rule itself, or by providing a reference to instructions expressed in other languages, such as OVAL.

In order to support greater flexibility in XCCDF, checklists can be tailored to meet certain needs. One way to do this is to enable or disable certain rules that are appropriate or inappropriate to a given endpoint, respectively. For example, a single XCCDF checklist might contain check items to evaluate the configuration of an endpoint's operating system. An endpoint deployed in an enterprise's DMZ might need to be locked down more than a common internal endpoint, due to the greater exposure to attack. In this case, some operating system configuration requirements for the DMZ endpoint might be unnecessary for the internal endpoint. Nonetheless, most configuration requirements would probably remain applicable to both environments (providing a common baseline for configuration of the given operating system) and thus be common to the checking instructions for both types of endpoints. XCCDF supports this by allowing a single checklist to be defined, but then tailored to the needs of the assessed endpoint. In the previous example, some Rules

that apply only to the DMZ endpoint would be disabled during the assessment of an internal endpoint and would not be exercised during the assessment or count towards the assessment results. To accomplish this, XCCDF uses the CPE Applicability Language. By enhancing this applicability language to support other aspects of endpoint characterization (see Appendix G.3.3.1.3), XCCDF will also benefit from these enhancements.

In addition, XCCDF Rules also support parameterization, allowing customization of the expected value for a given check item. For example, the DMZ endpoint might require a password of at least 12 characters, while an internal endpoint may only need 8 or more characters in its password. By employing parameterization of the XCCDF Rule, the same Rule can be used when assessing either type of endpoint, and simply be provided with a different target parameter each time to reflect the different role-based requirements. Sets of customizations can be stored within the XCCDF document itself: XCCDF Values store parameters values that can be used in tailoring, while XCCDF Profiles store sets of tailoring instructions, including selection of certain Values as parameters and the enabling and disabling of certain Rules. The tailoring capabilities supported by XCCDF allow a single XCCDF document to encapsulate configuration evaluation guidance that applies to a broad range of endpoint roles.

G.3.7. Evaluation Result Reporting

G.3.7.1. Configuration Evaluation Results

The evaluation guidance applied during posture assessment of an endpoint to customize the behavior of evaluators. Guidance can be used to define specific result output formats or to select the level-of-detail for the generated results. This guidance must be able to reference or be associated with the following previously defined information elements:

- configuration item identifiers,
- platform configuration identifiers, and
- collected Platform Configuration Posture Attributes.

G.3.7.1.1. Related Work

G.3.7.1.1.1. XCCDF TestResults

A general description of the eXtensible Configuration Checklist Description Format (XCCDF) was provided in section Appendix G.3.3.1.1.1. The XCCDF TestResult structure captures the outcome of assessing a single endpoint against the assessed items (i.e., XCCDF Rules) contained in an XCCDF instance document. XCCDF TestResults capture a number of important pieces of information about the assessment including:

- o The identity of the assessed endpoint. See Appendix G.3.3.1.1.2 for more about XCCDF structures used for endpoint identification.
- o Any tailoring of the checklist that might have been employed. See Appendix G.3.6.1.1.2 for more on how XCCDF supports tailoring.
- o The individual results of the assessment of each enabled XCCDF Rule in the checklist. See Appendix G.3.6.1.1.2 for more on XCCDF Rules.

The individual results for a given XCCDF Rule capture only whether the rule "passed", "failed", or experienced some exceptional condition, such as if an error was encountered during assessment. XCCDF 1.2 Rule results do not capture the actual state of the endpoint. For example, an XCCDF Rule result might indicate that an endpoint failed to pass requirement that passwords be of a length greater than or equal to 8, but it would not capture that the endpoint was, in fact, only requiring passwords of 4 or more characters. It may, however, be possible for a user to discover this information via other means. For example, if the XCCDF Rule uses an OVAL Definition to effect the Rule's evaluation, then the actual endpoint state may be captured in the corresponding OVAL System Characteristics file.

The XCCDF TestResult structure does provide a useful structure for understanding the overall assessment that was conducted and the results thereof. The ability to quickly determine the Rules that are not complied with on a given endpoint allow administrators to quickly identify where remediation needs to occur.

G.3.7.1.1.2. Open Vulnerability and Assessment Language

A general overview of OVAL was provided previously in Appendix G.3.2.1.1.2.1. OVAL Results provides a model for expressing the results of the assessment of the actual state of the posture attribute values collected of an endpoint (represented as an OVAL System Characteristics document) against the expected posture attribute values (defined in an OVAL Definitions document. Using OVAL Directives, the granularity of OVAL Results can also be specified. The OVAL Results model may be useful in providing a format for capturing the results of an assessment.

G.3.7.1.1.3. Asset Summary Reporting

A general overview of ASR was provided previously in Appendix G.3.5.1.1.1. As ASR provides a way to report summary information about assets, it can be used within the SACM Architecture to provide a way to aggregate asset information for later use. It makes no assertions about the data formats used by the assessment, but rather provides an XML, record-based way to collect aggregated information about assets.

By using ASR to collect this summary information within the SACM Architecture, one can provide a way to encode the details used by various reporting requirements, including user-definable reports.

G.3.7.1.1.4. ARF

A general overview of ARF was provided previously in Appendix G.3.3.1.2.1. Because ARF is data model agnostic, it can provide a flexible format for exchanging collection and evaluation information from endpoints. It additionally provides a way to encode relationships between guidance and assets, and as such, can be used to associate assessment results with guidance. This could be the guidance that directly triggered the assessment, or for guidance that is run against collected posture attributes located in a central repository.

G.3.7.2. Software Inventory Evaluation Results

The results of an evaluation of an endpoint's software inventory against an authorized software list. The authorized software list represents the policy for what software units are allowed, prohibited, and mandatory for an endpoint.

Appendix H. Graph Model

TODO: Write text on how the information model above can be realized in this kind of graph model.

The graph model describes how security information is structured, related, and accessed. Control of operations to supply and/or access the data is architecturally distinct from the structuring of the data in the information model. Authorization may be applied by the Control Plane (as defined in the SACM Architecture [I-D.ietf-sacm-architecture]) to requests for information from a consumer or requests for publication from a provider, and may also be applied by a provider to a direct request from a consumer.

This architecture addresses information structure independently of the access/transport of that information. This separation enables scalability, customizability, and extensibility. Access to provide or consume information is particularly suited to publish/subscribe/ query data transport and data access control models.

This graph model is a framework that:

o Facilitates the definition of extensible data types that support SACM's use cases

o Provides a structure for the defined data types to be exchanged via a variety of data transport models

o Describes components used in information exchange, and the objects exchanged

o Captures and organizes evolving information and information relationships for multiple data publishers

o Provides access to the published information via publish, query, and subscribe operations

o Leverages the knowledge and experience gained from incorporating TNC IF-MAP into many disparate products that have to integrate and share an information model in a scalable, extensible manner

H.1. Background: Graph Models

Knowledge is often represented with graph-based formalisms. A common formalism defines a graph as follows:

o A set of *vertices*

o A set of *edges*, each connecting two vertices (technically, an edge is an ordered pair of vertices)

o A set of zero or more *properties* attached to each vertices and edges. Each property consists of a type and a optionally a value. The type and the value are typically just strings

```
      +------------------+                     +-----------------+
      | Id:          1   |    parent-of        |Id:          2   |
      | Given name:  Sue | ------------------> |Given name:  Ann |
      | Family name: Wong|                     |Family name: Wong|
      +------------------+                     +-----------------+

            A VERTEX            AN EDGE            A VERTEX
```

Figure 18: Knowledge represented by a graph

A pair of vertices connected by an edge is commonly referred to as a triple that comprises subject, predicate and object. For example, subject = Sue Wong, predicate = is-parent-of, object = Ann Wong. A common language that uses this representation is the Resource Description Framework (RDF) [W3C.REC-rdf11-concepts-20140225].

H.2. Graph Model Overview

The proposed model, influenced by IF-MAP, is a labeled graph: each vertex has a label.

A table of synonyms follows.

Graph Theory	Graph Databases	IF-MAP and This Internet Draft
Vertex or Node	Node	-
Label	-	Identifier
Edge	Edge	Link
-	Property	Metadata Item

In this mode, identifiers and metadata have hierarchical structure.

The graphical aspect makes this model well suited to non-hierarchical relationships, such as connectivity in a computer network.

Hierarchical properties allow this model to accommodate structures such as YANG [RFC6020] data models.

H.3. Identifiers

Each identifier is an XML element. For extensibility, schemas use xsd:anyAttribute and such.

Alternately, this model could be changed to use another hierarchical notation, such as JSON.

Identifiers are unique: two different vertices cannot have equivalent identifiers.

An identifier has a type. There is a finite, but extensible, set of identifier types. If the identifier is XML, the type is based on the XML schema.

In IF-MAP, standard identifier types include IP address, MAC address, identity, and overlay network. Additional identifier types will need to be standardized for SACM use cases.

Any number of metadata items can be attached to an identifier.

Some identifiers, especially those relating to identity, address, and location, require the ability to specify an administrative domain (such as AD domain, L2 broadcast domain / L3 routing domain, or geographic domain) in order to differentiate between instances with the same name occurring in different realms.

H.4. Links

A link can be thought of as an ordered pair of identifiers.

Any number of metadata items can be attached to a link.

H.5. Metadata

A metadata item is the basic unit of information, and is attached to an identifier or to a link.

A given metadata item is an XML document. In IF-MAP metadata items are generally small. However, larger ones, such as YANG data models, can also fit. For extensibility, the XML schemas of metadata items use xsd:anyAttribute and such.

Alternately, this model could be changed to use another hierarchical notation, such as JSON.

A metadata item has a type. There is a finite, but extensible, set of metadata types. If the metadata item is XML, the type is based on the XML schema. An example metadata type is http://www.trustedcomputinggroup.org/2010/IFMAP-METADATA/2#device-characteristic.

TNC IF-MAP Metadata for Network Security [TNC-IF-MAP-NETSEC-METADATA] and TNC IF-MAP Metadata for ICS Security [TNC-IF-MAP-ICS-METADATA] define many pertinent metadata types. More will need to be standardized for SACM use cases.

H.6. Use for SACM

Many of the information elements can be represented as vertices, and many of the relationships can be represented as edges.

Identifiers are like database keys. For example, there would be identifiers for addresses, identities, unique endpoint identifiers, software component identifiers, and hardware component identifiers. The inventory of software instances and hardware instances within an endpoint might be expressed using a single YANG description, as a single metadata item in the graph. Where to put Endpoint Attribute Assertions, Evaluation Results, and the like is an open question.

H.7. Provenance

Provenance helps to protect the SACM ecosystem against a misled or malicious provider.

The provenance of a metadata item includes:

- o The time when the item was produced
- o The component that produced the item, including its software and version
- o The policies that governed the producing component, with versions
- o The method used to produce the information (e.g., vulnerability scan)

How provenance should be expressed is an open question. For reference, in IF-MAP provenance of a metadata item is expressed within the metadata item [TNC-IF-MAP-NETSEC-METADATA]. For example, there is a top-level XML attribute called "timestamp".

It is critical that provenance be secure from tampering. How to achieve that security is out of scope of this document.

H.8. Extensibility

Anyone can define an identifier type or a metadata type, by creating an XML schema (or other specification). There is no need for a central authority. Some deployments may exercise administrative control over the permitted identifier types and metadata types; others may leave components free rein.

A community of components can agree on and use new identifier and metadata types, if the administrators allow it. This allows rapid

innovation. Intermediate software that conveys graph changes from one component to another does not need changes. Components that do not understand the new types do not need changes. Accordingly, a consumer normally ignores metadata types and identifier types it does not understand.

As a proof point for this agility, the original use cases for TNC IF-MAP Binding for SOAP [TNC-IF-MAP-SOAP-Binding] were addressed in TNC IF-MAP Metadata for Network Security [TNC-IF-MAP-NETSEC-METADATA]. Some years later an additional, major set of use cases, TNC IF-MAP Metadata for ICS [TNC-IF-MAP-ICS-METADATA], were specified and implemented.

Authors' Addresses

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

Email: david.waltermire@nist.gov

Kim Watson
United States Department of Homeland Security
DHS/CS&C/FNR
245 Murray Ln. SW, Bldg 410
MS0613
Washington, DC 20528
USA

Email: kimberly.watson@hq.dhs.gov

Clifford Kahn
Pulse Secure, LLC
2700 Zanker Road, Suite 200
San Jose, CA 95134
USA

Email: cliffordk@pulsesecure.net

Lisa Lorenzin
Pulse Secure, LLC
2700 Zanker Road, Suite 200
San Jose, CA 95134
USA

Email: llorenzin@pulsesecure.net

Michael Cokus
The MITRE Corporation
903 Enterprise Parkway, Suite 200

Hampton, VA 23666
USA

Email: msc@mitre.org

Daniel Haynes
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730
USA

Email: dhaynes@mitre.org