SACM D. Waltermire, Ed.
Internet-Draft NIST
Intended status: Standards Track K. Watson
Expires: January 9, 2017 DHS
C. Kahn
L. Lorenzin
Pulse Secure, LLC
M. Cokus
D. Haynes
The MITRE Corporation
July 8, 2016

SACM Information Model
draft-ietf-sacm-information-model-06

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

o TE Characterization

o TE Classification

o Collection

o Evaluation

o 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 Figure 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 7. 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 assets in 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:

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

- 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:

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

- 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 conceptual 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 must provide the information contained in these two subjects although it is up to the implementer whether or not the subjects are explicitly defined in a data model.

The notation the SACM IM is defined in is based on a modified version of the IP Information Flow Export (IPFIX) Information Model syntax described in Section 2.1 of [RFC7012]. The customized syntax used by the SACM IM is defined below in Figure 2.

```
    elementId (required):    The numeric identifier of the
                             Information Element. It is used
                             for the compact identification
                             of an Information Element. If
                             this identifier is used without
                             an enterpriseID, then the
                             elementId must be unique, and
                             the description of allowed values
                             is administrated by IANA. The
                             value "TBD" may be used during
                             development of the information
                             model until an elementId is
                             assigned by IANA and filled
                             in at publication time.
```

enterpriseId (optional): Enterprises may wish to define Information Elements without registering them with IANA, for example, for enterprise-internal purposes. For such Information Elements, the elementId is not sufficient when used outside the enterprise. If specifications of enterprise-specific Information Elements are made public and/or if enterprise-specific identifiers are used by SACM components outside the enterprise, then the enterprise-specific identifier MUST be made globally unique by combining it with an enterprise identifier. Valid values for the enterpriseId are defined by IANA as Structure of Management Information (SMI) network management private enterprise numbers.

name (required): A unique and meaningful name for the Information Element.

dataType (required): There are two kinds of datatypes: simple and structured. Attributes are defined using simple datatypes and subjects are defined using structured datatypes. The contents of the datatype field will be either a reference to one of the simple datatypes listed in Section 5.1, or the specification of structured datatype as defined in Section 5.2.

status (required): The status of the specification of the Information Element. Allowed values are "current" and "deprecated". All newly defined Information Elements have "current" status. The process for moving Information Elements to the "deprecated" status is TBD.

description (required): Describes the meaning of the

```
                              Information Element, how it is
                              derived, conditions for its use,
                              etc.

                              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          |
                              +-------+----------------------+
                              |       |                      |
                              +-------+----------------------+

       references (optional):  Identifies other RFCs or documents
                              outside the IETF which provide
                              additional information or context
                              about the Information Element.
```

Figure 2: Information Element Specification Template

4.1. 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 3: Example set of IEs associated with a timestamp and a target endpoint label.

4.2. 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 4: 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 5: Example of conflicting information originating from different SACM components.

4.3. 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:

- 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

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

4.4. 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 7: Example of a SACM statement containing an event.

4.5. 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.

4.6. 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. Also, consider inserting table that discusses the various properties of designation attributes and include it in this section to help others determine whether or not a new Information Element should be considered a designation attribute. Lastly, explain how designation attributes can be used. For example, letting a consumer identify an endpoint, for two purposes:

- To tell whether two endpoint attribute assertions concern the same endpoint

- To respond to compliance measurements, for example by reporting, remediating, and quarantining (SACM does not specify these responses, but SACM exists to enable them.)

4.7. 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.

5. Abstract Data Types

This section describes the set of valid abstract data types that can be used for the specification of the SACM Information Elements in Section 7. SACM currently supports two classes of datatypes that can be used to define Information Elements.

- Simple: Datatypes that are atomic and are used to define the type of data represented by an attribute Information Element.

- Structured: Datatypes that can be used to define the type of data represented by a subject Information Element.

Note that further abstract data types may be specified by future extensions of the SACM information model.

5.1. Simple Datatypes

5.1.1. IPFIX Datatypes

To facilitate the use of existing work, SACM supports the following abstract data types defined in Section 3 of [RFC7012].

- unsigned8, unsigned16, unsigned32, unsigned64

- signed8, signed16, signed32, signed64

- float32, float64

- boolean

- macAddress

- octetArray

- string

- dateTimeSeconds, dateTimeMilliseconds, dateTimeMicroseconds, dateTimeNanoSeconds

- ipv4Address, ipv6Address

5.1.2. ciscoTrainSoftwareVersion

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

5.1.3. rpmSoftwareVersion

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

5.1.4. simpleSoftwareVersion

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

5.2. Structured Datatypes

5.2.1. List Datatypes

SACM defines the following abstract list data types that are used to represent the structured data associated with subjects.

- list: indicates that the Information Element order is not significant but MAY be preserved.

- orderedList: indicates that Information Element order is significant and MUST be preserved.

The notation for defining a SACM structured datatype is based on regular expressions, which are composed of the keywords "list" or "orderedList" and an Information Element expression. IE expressions use some of the regular expression syntax and operators, but the terms in the expression are the names of defined Information Elements instead of character classes. The syntax for defining list and orderedList datatypes is described below, using BNF:

```
       <list-def> -> ("list"|"orderedList") "(" <ie-expression> ")"

       <ie-expression> -> <ie-name> <cardinality>?
                          ( ("," | "|") <ie-name> <cardinality>?)*

       <cardinality> -> "*" | "+" | "?" |
                        ( "(" <non-neg-int> ("," <non-neg-int>)? ")" )
```

Figure 8: Syntax for Defining List Datatypes

As seen above, multiple occurences of an Information Element may be present in a structured datatype. The cardinality of an Information Element within a structured Information Element definition is defined by the following operators:

```
       * - zero or more occurrences

       + - one or more occurrences

       ? - zero or one occurrence

      (m,n) - between m and n occurrences
```

Figure 9: Specifying Cardinality for Structured Datatypes

The absence of a cardinality operator implies one mandatory occurrence of the Information Element.

Below is an example of a structured Information Element definition.

```
personInfo = list(firstName, middleNames?, lastName)
firstName = string
middleNames = orderedList(middleName+)
middleName = string
lastName = string
```

As an example, consider the name "John Ronald Reuel Tolkien". Below are instances of this name, structured according to the personInfo definition.

```
personInfo = (firstName="John", middleNames(middleName="Ronald",
              middleName="Reuel"), lastName="Tolkien")

personInfo = (middleNames(middleName="Ronald", middleName=" Reuel"),
              lastName="Tolkien", firstName="John")
```

The instance below is not legal with respect to the definition of personInfo because the order in middleNames is not preserved.

```
personInfo = (firstName="John", middleNames(middleName=" Reuel",
              middleName="Ronald"), lastName="Tolkien")
```

Figure 10: Example of Defining a Structured Datatype

6. Information Model Assets

In order to represent the Information Elements related to the areas listed in Section 3.1, the information model defines the information needs (or metadata about those information needs) related to following types of assets which arse defined in [I-D.ietf-sacm-terminology] (and included below for convenience) which are of interest to SACM. Specifically:

- Endpoint

- Software Component

- Hardware Component

- Identity

- Guidance

- Evaluation Results

The following figure shows the make up of an Endpoint asset which contains zero or more hardware components and zero or more software components each of which may have zero or more instances running an

endpoint at any given time as well as zero or more identities that act on behalf of the endpoint when interfacing with other endpoints, tools, or services. An endpoint may also contain other endpoints in the case of a virtualized environment.

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

Figure 11: Model of an Endpoint

6.1. Asset

As defined in [RFC4949], an asset is 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 Identity. Furthermore, an asset is not necessarily owned by an organization.

6.2. Endpoint

From [RFC5209], an endpoint is any computing device that can be connected to a network. Such devices normally are associated with a particular link layer address before joining the network and

potentially an IP address once on the network. This includes: laptops, desktops, servers, cell phones, or any device that may have an IP address.

To further clarify, an endpoint is any physical or virtual device that may have a network address. Note that, network infrastructure devices (e.g. switches, routers, firewalls), which fit the definition, are also considered to be endpoints within this document.

Physical endpoints are always composites that are composed of hardware components and software components. Virtual endpoints are composed entirely of software components and rely on software components that provide functions equivalent to hardware components.

The SACM architecture differentiates two essential categories of endpoints: Endpoints whose security posture is intended to be assessed (target endpoints) and endpoints that are specifically excluded from endpoint posture assessment (excluded endpoints).

6.3. Hardware Component

Hardware components are the distinguishable physical components that compose an endpoint. The composition of an endpoint can be changed over time by adding or removing hardware components. In essence, every physical endpoint is potentially a composite of multiple hardware components, typically resulting in a hierarchical composition of hardware components. The composition of hardware components is based on interconnects provided by specific hardware types (e.g. mainboard is a hardware type that provides local busses as an interconnect). In general, a hardware component can be distinguished by its serial number.

Examples of a hardware components include: motherboards, network interfaces, graphics cards, hard drives, etc.

6.4. Software Component

A software package installed on an endpoint (including the operating system) as well as a unique serial number if present (e.g. a text editor associated with a unique license key).

It should be noted that this includes both benign and harmful software packages. Examples of benign software components include: applications, patches, operating system kernel, boot loader, firmware, code embedded on a webpage, etc. Examples of malicious software components include: malware, trojans, viruses, etc.

6.4.1. Software Instance

A running instance of the software component (e.g. on a multi-user system, one logged-in user has one instance of a text editor running and another logged-in user has another instance of the same text editor running, or on a single-user system, a user could have multiple independent instances of the same text editor running).

6.5. Identity

TODO: Define an Asset Identity asset. NOTE: Make sure it is clear that this is not identity in the sense of what we have been saying endpoint identity (now designation).

Examples of an identity include: username, user and device certificates, etc.

6.6. Guidance

TODO: Need to resolve the forms of Guidance in the terminology and those listed as sub-sections below.

Guidance is input instructions to processes and tasks, such as collection or evaluation. Guidance influences the behavior of a SACM component and is considered content of the management plane. Guidance can be manually or automatically generated or provided. Typically, the tasks that provide guidance to SACM components have a low-frequency and tend to be be sporadic. A prominent example of guidance are target endpoint profiles, but guidance can have many forms, including:

> Configuration, e.g. a SACM component's name, or a CMDB's IPv6 address.
>
> Profiles, e.g. a set of expected states for network behavior associated with target endpoints employed by specific users.
>
> Policies, e.g. an interval to refresh the registration of a SACM component, or a list of required capabilities for SACM components in a specific location.

6.6.1. Internal Collection Guidance

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

6.6.2. External Collection Guidance

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

6.6.3. Evaluation Guidance

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

6.6.4. 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.

6.7. Evaluation Results

Evaluation results are the resulting values from having evaluated a set of posture attributes.

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 11 does not show this). In the limiting case, Endpoint Attribute Assertions are not retained. When an Endpoint Attribute Assertion arrives, an evaluator produces an Evaluation Result. These mechanics are out of the scope of the Information Model.

7. 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.

7.1. hardwareSerialNumber

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

7.2. interfaceName

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

7.3. interfaceIndex

elementId: TBD
name: interfaceIndex
dataType: unsigned32
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].

7.4. interfaceMacAddress

elementId: TBD
name: interfaceMacAddress
dataType: macAddress
status: current
description: The IEEE 802 MAC address associated with a network
             interface on an endpoint.

7.5. interfaceType

elementId: TBD
name: interfaceType
dataType: unsigned32
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].

7.6. interfaceFlags

elementId: TBD
name: interfaceFlags
dataType: unsigned16
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

7.7. networkInterface

elementId: TBD
name: networkInterface
dataType: orderedList(interfaceName, interfaceIndex, macAddress, ifType, flags)
status: current
description: Information about a network interface installed on an endpoint. The following high-level digram describes the structure of networkInterface information element.

7.8. softwareIdentifier

```
elementId: TBD
name: softwareIdentifier
dataType: string
status: current
description: A globally unique identifier for a particular
             software application.
```

7.9. softwareTitle

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

7.10. softwareCreator

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

7.11. simpleSoftwareVersion

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

7.12. rpmSoftwareVersion

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

7.13. ciscoTrainSoftwareVersion

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

7.14. softwareVersion

```
   elementId: TBD
   name: softwareVerison
   dataType: list(simpleSoftwareVersion | rpmSoftwareVersion |
                  ciscoTrainSoftwareVersion)
   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.
```

7.15. lastUpdated

```
   elementId: TBD
   name: lastUpdated
   dataType: dateTimeSeconds
   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)
```

7.16. softwareInstance

```
   elementId: TBD
   name: softwareInstance
   dataType: orderedList(softwareIdentifier, title, creator,
                         softwareVersion, lastUpdated)
   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.
```

7.17. globallyUniqueIdentifier

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

7.18. dataOrigin

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

7.19. dataSource

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

7.20. creationTimestamp

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

7.21. collectionTimestamp

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

7.22. publicationTimestamp

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

7.23. relayTimestamp

```
elementId: TBD
name: relayTimestamp
dataType: dateTimeSeconds
status: current
metadata: true
description: The date and time when the posture
             information was relayed to another SACM Component.
```

7.24. storageTimestamp

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

7.25. type

elementId: TBD
name: type
dataType: unsigned16
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
...	...

7.26. protocolIdentifier

elementId: TBD
name: protocolIdentifier
dataType: unsigned8
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.

7.27. sourceTransportPort

elementId: TBD
name: sourceTransportPort
dataType: unsigned16
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.

7.28. sourceIPv4PrefixLength

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

7.29. ingressInterface

elementId: TBD
name: ingressInterface
dataType: unsigned32
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].

7.30. destinationTransportPort

elementId: TBD
name: destinationTransportPort
dataType: unsigned16
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.

7.31. sourceIPv6PrefixLength

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

7.32. sourceIPv4Prefix

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

7.33. destinationIPv4Prefix

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

7.34. sourceMacAddress

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

7.35. ipVersion

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

7.36. interfaceDescription

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

7.37. applicationDescription

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

7.38. applicationId

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

7.39. applicationName

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

7.40. exporterIPv4Address

```
   elementId: TBD
   name: exporterIPv4Address
   dataType: ipv4Address
   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.
```

7.41. exporterIPv6Address

```
elementId: TBD
name: exporterIPv6Address
dataType: ipv6Address
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.
```

7.42. portId

```
elementId: TBD
name: portId
dataType: unsigned32
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.
```

7.43. templateId

```
elementId: TBD
name: templateId
dataType: unsigned16
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.
```

7.44. collectorIPv4Address

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

7.45. collectorIPv6Address

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

7.46. informationElementIndex

elementId: TBD
name: informationElementIndex
dataType: unsigned16
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.

7.47. informationElementId

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

7.48. informationElementDataType

elementId: TBD
name: informationElementDataType
dataType: unsigned8
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.

7.49. informationElementDescription

elementId: TBD
name: informationElementDescription
dataType: string
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.

7.50. informationElementName

```
elementId: TBD
name: informationElementName
dataType: string
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.
```

7.51. informationElementRangeBegin

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

7.52. informationElementRangeEnd

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

7.53. informationElementSemantics

```
elementId: TBD
name: informationElementSemantics
dataType: unsigned8
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.
```

7.54. informationElementUnits

```
elementId: TBD
name: informationElementUnits
dataType: unsigned16
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.
```

7.55. userName

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

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

7.56. applicationCategoryName

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

7.57. mibObjectValueInteger

```
elementId: TBD
name: mibObjectValueInteger
dataType: signed64
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.
```

7.58. mibObjectValueOctetString

elementId: TBD
name: mibObjectValueOctetString
dataType: octetArray
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.

7.59. mibObjectValueOID

elementId: TBD
name: mibObjectValueOID
dataType: octetArray
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.

7.60. mibObjectValueBits

```
elementId: TBD
name: mibObjectValueBits
dataType: octetArray
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.
```

7.61. mibObjectValueIPAddress

```
elementId: TBD
name: mibObjectValueIPAddress
dataType: ipv4Address
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.
```

7.62. mibObjectValueCounter

elementId: TBD
name: mibObjectValueCounter
dataType: unsigned64
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.

7.63. mibObjectValueGauge

elementId: TBD
name: mibObjectValueGauge
dataType: unsigned32
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.

7.64. mibObjectValueTimeTicks

elementId: TBD
name: mibObjectValueTimeTicks
dataType: unsigned32
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.

7.65. mibObjectValueUnsigned

elementId: TBD
name: mibObjectValueUnsigned
dataType: unsigned64
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.

7.66. mibObjectValueTable

elementId: TBD
name: mibObjectValueTable
dataType: orderedList
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.

7.67. mibObjectValueRow

```
   elementId: TBD
   name: mibObjectValueRow
   dataType: orderedList
   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.
```

7.68. mibObjectIdentifier

```
   elementId: TBD
   name: mibObjectIdentifier
   dataType: octetArray
   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].
```

7.69. mibSubIdentifier

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

7.70. mibIndexIndicator

```
   elementId: TBD
   name: mibIndexIndicator
   dataType: unsigned64
   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.
```

7.71. mibCaptureTimeSemantics

elementId: TBD
name: mibCaptureTimeSemantics
dataType: unsigned8
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

7.72. mibContextEngineID

elementId: TBD
name: mibContextEngineID
dataType: octetArray
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.

7.73. mibContextName

```
elementId: TBD
name: mibContextName
dataType: string
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.
```

7.74. mibObjectName

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

7.75. mibObjectDescription

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

7.76. mibObjectSyntax

```
elementId: TBD
name: mibObjectSyntax
dataType: string
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].
```

7.77. mibModuleName

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

8. 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 11 illustrates those elements along with their major communication paths.

8.1. 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.

8.1.1. 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:

- 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.

- 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.

- 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.

8.1.2. Identifiers

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

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

- Software asset

- Network Session

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

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

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

- Guidance

8.1.3. Metadata

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

- 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.

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

- 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.

- 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.

8.1.4. 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.

8.2. 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.

9. 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.

9.1. 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

10. IANA Considerations

This memo includes no request to IANA.

11. Operational Considerations

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

12. Privacy Considerations

TODO: Need to include various privacy considerations here.

13. 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.

14. References

14.1. Normative References

[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>.

[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>.

14.2. Informative References

[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.

[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>.

[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>.

[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>.

[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>.

[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>.

[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.

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 Identifying Attributes.

Split the figure into Figure Model of Endpoint and Figure Information Elements.

Added Figure Information Elements Take 2, proposing a triple-store model.

Some editorial cleanup

A.3. Changes in Revision 03

Moved Appendix A.1, Appendix A.2, and Mapping to SACM Use Cases 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 6 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 7 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.

A.6. Changes in Revision 06

Clarified that the SACM statement and content-element subjects are conceptual and that they do not need to be explicitly defined in a data model as long as the necessary information is provided.

Updated the IPFIX syntax used to define Information Elements. There are still a couple of open issues that need to be resolved.

Updated some of the Information Elements contained in Section 7 to use the revised IPFIX syntax. The rest of the Information Elements will be converted in a later revision.

Performed various clean-up and refactoring in Sections 6 and 7. Still need to go through Section 8.

Removed appendices that were not referenced in the body of the draft. The text from them is still available in previous revisions of this document if needed.

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