Internet Engineering Task Force (IETF)
Request for Comments: 8412
Category: Standards Track
ISSN: 2070-1721

C. Schmidt
D. Haynes
C. Coffin
The MITRE Corporation
D. Waltermire
National Institute of Standards and Technology
J. Fitzgerald-McKay
United States National Security Agency

July 2018

Software Inventory Message and Attributes (SWIMA) for PA-TNC

Abstract

This document extends "PA-TNC: A Posture Attribute (PA) Protocol Compatible with Trusted Network Connect (TNC)" (RFC 5792) by providing specific attributes and message exchanges to allow endpoints to report their installed software inventory information to a NEA Server, as defined in "Network Endpoint Assessment (NEA): Overview and Requirements" (RFC 5209).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Knowing the list of software installed on endpoints is useful to understand and maintain the security state of a network. For example, if an enterprise policy requires the presence of certain software and prohibits the presence of other software, reported software installation information can be used to indicate compliance and non-compliance with these requirements. Endpoint software installation inventory lists (hereinafter "software inventories") can further be used to determine an endpoint's exposure to attack based on comparison of vulnerability or threat alerts against identified software's patch-level data. These are some of the highly useful management use cases supported by software inventory data.

Software Inventory Message and Attributes (SWIMA) for PA-TNC (see "PA-TNC: A Posture Attribute (PA) Protocol Compatible with Trusted Network Connect (TNC)" [RFC5792]) provides a standardized method for exchanging software inventory data that includes a unique Software Identifier associated with a specific version of a software product. SWIMA can also convey metadata about software products beyond this identifier. SWIMA enables software identification, installation, and characterization information to be transported to a central server from any endpoint that supports this specification. Such information can come from multiple sources, including tag files (such as ISO Software Identification (SWID) tags [SWID15]), reports from third-party inventory tools, output from package managers, and other sources. SWIMA does not standardize how software is detected, instead relying on a set of "data sources" to provide information about installed software. SWIMA provides a flexible transport capable of conveying this information, regardless of how it is expressed.

This specification is designed to only report software that is installed on a target endpoint. In particular, it does not monitor or report information about what software is running on the endpoint. Likewise, it is not intended to report individual files, libraries, installation packages, or similar artifacts. While all of this information has its uses, this information requires different metadata and monitoring methods. As a result, this specification focuses solely on software inventory information, leaving the reporting of other classes of endpoint information to other specifications.

Note that while this specification focuses on "software inventory", the mechanisms it describes could also be used to convey information about firmware and operating systems associated with an endpoint. The focus on software throughout this document should not be read as excluding the use of SWIMA for these other purposes.

This specification defines a new set of PA-TNC attributes; these attributes are used to communicate requests for software inventory information and software installation change events. The exchange of these messages allows software inventory information to be sent to a Network Endpoint Assessment (NEA) Server, which can make this information available to other applications.

Part of the motivation for the development of SWIMA was to support the IETF’s Security Automation and Continuous Monitoring (SACM) architecture. More details about SWIMA’s role in SACM appear in Section 7. However, SWIMA has no dependencies on any part of SACM and is usable wherever the NEA architecture is employed.

1.1. Network Endpoint Assessment (NEA)

SWIMA defines extensions to the PA-TNC specification [RFC5792]; these extensions are part of the NEA architecture. The NEA specifications define an open solution architecture that enables network operators to collect and utilize information about endpoint configuration and state. This information can be used to enforce policies and monitor endpoint health, among many other activities. Information about the software present on an endpoint is an important consideration for such activities. The new PA-TNC attributes defined in this document are used to communicate software inventory evidence, collected from a range of possible sources, from the Posture Collector on the endpoint to the Posture Validator on a NEA Server using the PA-TNC interface, as shown in Figure 1 below.

```
       +-------------+                          +--------------+
       |  Posture    |   <--------PA-------->   |   Posture    |
       |  Collectors |                          |   Validators |
       |  (1 .. N)   |                          |   (1 .. N)   |
       +-------------+                          +--------------+
             |                                         |
             |                                         |
             |                                         |
       +-------------+                          +--------------+
       |   Posture   |                          |   Posture    |
       |   Broker    |   <--------PB-------->   |   Broker     |
       |   Client    |                          |   Server     |
       +-------------+                          +--------------+
             |                                         |
             |                                         |
       +-------------+                          +--------------+
       |   Posture   |                          |   Posture    |
       |   Transport |   <--------PT-------->   |   Transport  |
       |   Client    |                          |   Server     |
       |   (1 .. N)  |                          |   (1 .. N)   |
       +-------------+                          +--------------+
          NEA CLIENT                               NEA SERVER
```

Figure 1: NEA Reference Model

To better understand this specification, the reader should review the NEA reference architecture as described in "Network Endpoint Assessment (NEA): Overview and Requirements" [RFC5209]. The reader should also review the PA-TNC interfaces as defined in RFC 5792 [RFC5792].

This document is based on standards published by the Trusted Computing Group's Trusted Network Communications (TNC) workgroup (see <https://trustedcomputinggroup.org/>). The TNC and NEA architectures are interoperable, and many components are equivalent.

1.2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Definitions

This section defines terms that have special meaning within this document.

- SWIMA-PC - A NEA Posture Collector (PC) that interprets SWIMA attributes sent by SWIMA-PVs and that conforms to this specification. Note that such a Posture Collector might also support other PA-TNC exchanges beyond those defined herein.

- SWIMA-PV - A NEA Posture Validator (PV) that interprets SWIMA attributes sent by SWIMA-PCs and that conforms to this specification. Note that such a Posture Validator might also support other PA-TNC exchanges beyond those defined herein.

- SWIMA Attribute - A PA-TNC attribute (as defined in RFC 5792 [RFC5792]) whose structure and behavior is defined in this specification.

- Endpoint's Software Inventory Evidence Collection - The set of information regarding the set of software installed on an endpoint. An endpoint's Software Inventory Evidence Collection might include information created by or derived from multiple sources, including but not limited to SWID tag files deposited on the filesystem during software installation, information generated by software discovery tools, and information dynamically generated by a software or package management system on an endpoint.

- Software Inventory Evidence Record - Part of the endpoint's Software Inventory Evidence Collection (which is composed of "records"). Each record corresponds to one installed instance of a particular software product as reported by some data source. It is possible for a single installed instance to have multiple

Software Inventory Evidence Records in an endpoint's Software Inventory Evidence Collection -- this can happen if multiple sources all report the same software installation instance.

- Software Identifier - A string associated with a specific version of a specific software product. These identifiers are derived from the records used to describe software products. SWIMA does not limit the formats of these records, nor does it enforce that all data sources populate records using the same format. As such, while each Software Identifier uniquely identifies a specific software product, the same software product might be associated with multiple Software Identifiers reflecting differences between different data sources and supported record formats.

2. Background

2.1. Supported Use Cases

This section describes the use cases supported by this specification. The primary use of exchanging software inventory information over the PA-TNC interface is to enable a challenger (e.g., a NEA Server) to obtain inventory evidence about some system in a way that conforms to NEA procedures. Collected software information can support a range of security activities, including determining whether an endpoint is permitted to connect to the enterprise, determining which endpoints contain software that requires patching, and similar activities.

2.1.1. Use Software Inventory as an Access Control Factor

Some enterprises might define security policies that require connected endpoints to have certain pieces of security software installed. By contrast, some security policies might prevent access to resources by endpoints that have certain prohibited pieces of software installed, since such applications might pose a security risk. To support such policies, the NEA Server needs to collect software inventory evidence from a target endpoint that is seeking to initiate or continue connectivity to the enterprise resource.

Based on this specification, the SWIMA-PC can provide a complete or partial inventory to the SWIMA-PV as required to determine policy compliance. The SWIMA-PV can then use this as evidence of compliance or non-compliance to make a policy-based access decision.

2.1.2. Central Stores of Up-to-Date Endpoint Software Inventory Data

Many tools use information about an endpoint’s software inventory to monitor and enforce the security of a network. For example, a software-patching tool needs to determine if there is out-of-date software installed that needs to be updated. A vulnerability management tool needs to identify endpoints with known vulnerable software installed (patched or otherwise) to gauge an endpoint’s relative exposure to attack. A license management tool needs to verify that all installed software within the enterprise is accounted for. A central repository representing an up-to-date understanding of each endpoint’s software inventory facilitates these activities. Multiple tools can share such a repository, ensuring that software inventory information is collected more frequently and efficiently, leading to a more complete and consistent understanding of installed software state as compared to each tool collecting the same inventory information from endpoints individually.

This specification supports these activities through a number of mechanisms. As noted above, a SWIMA-PC can provide a complete list of software present in an endpoint’s Software Inventory Evidence Collection to the SWIMA-PV, which can then pass this information on to a central repository, such as a Configuration Management Database (CMDB) or similar application. In addition, SWIMA-PCs are required to be able to monitor for changes to an endpoint’s Software Inventory Evidence Collection in near real time and can be requested to immediately push reports of detected changes to the SWIMA-PV. Thus, any central repository fed by a SWIMA-PV receiving inventory information can be updated quickly after a change occurs. Keeping a central repository synchronized with current software inventory information in this way allows tools to make efficient decisions based on up-to-date, consistent information.

2.1.3. PA-TNC Use Cases

SWIMA is intended to operate over the PA-TNC interface and, as such, is intended to meet the use cases set out in the PA-TNC specification.

2.2. Use Cases That Are Not Supported

Some use cases not covered by this specification include:

- Addressing how the endpoint’s Software Inventory Evidence Collection is populated. In particular, NEA components are not expected to perform software discovery activities beyond compiling information in an endpoint’s Software Inventory Evidence Collection. This collection might come from multiple sources on the endpoint (e.g., information generated dynamically by package management tools or discovery tools, as well as SWID tag files discovered on the filesystem). While an enterprise might make use of software discovery capabilities to identify installed software, such capabilities are outside the scope of this specification.

- Converting inventory information expressed in a proprietary format into formats used in the attributes described in this specification. Instead, this specification focuses exclusively on defining interfaces for the transportation of software information, expecting that reporting tools will converge around some set of standardized formats for this information.

- Mechanisms for a Posture Validator to request a specific list of software information based on arbitrary software properties. For example, requesting only information about software from a particular vendor is not supported. After the endpoint's Software Inventory Evidence Collection has been copied to some central location, such as the CMDB, processes there can perform queries based on any criteria present in the collected information, but this specification does not address using such queries to constrain the initial collection of this information from the endpoint.

- Use of properties of certain sources of software information that might facilitate local tests (i.e., on the endpoint) of endpoint state. For example, the optional package_footprint field of an ISO SWID tag can contain a list of files and hash values associated with the software indicated by the tag. Tools on the endpoint can use the values in this field to test for the presence of the indicated files. Successful evaluation of such tests leads to greater assurance that the indicated software is present on the endpoint. Currently, most SWID tag creators do not provide values for tag fields that support local testing. For this reason, the added complexity of supporting endpoint testing using these fields is out of scope for this specification, but this topic may be considered in a future version.

2.3. SWIMA Requirements

Below are the requirements that SWIMA must meet in order to successfully play its role in the NEA architecture.

Efficient: The NEA architecture enables delay of network access until the endpoint is determined not to pose a security threat to the network, based on its asserted integrity information. To minimize user frustration, SWIMA ought to minimize overhead delays and make PA-TNC communications as rapid and efficient as possible.

Scalable: SWIMA needs to be usable in enterprises that contain tens of thousands of endpoints or more. As such, it needs to allow security tools to make decisions based on up-to-date information about an endpoint’s software inventory without creating an excessive burden on the enterprise’s network.

Support precise and complete historical reporting: This specification outlines capabilities that support real-time understanding of the state of endpoints in a network in a way that can be used by other tools. One means of facilitating such an outcome is for a CMDB to be able to contain information about all endpoints connected to the enterprise for all points in time between the endpoint’s first connection and the present. In such a scenario, it is necessary that any SWIMA-PC be able to report any changes to its Software Inventory Evidence Collection in near real time while connected and, upon reconnection to the enterprise, be able to update the NEA Server (and, through it, the CMDB) with regard to the state of its Software Inventory Evidence Collection throughout the entire interval when it was not connected.

2.4. Non-SWIMA Requirements

There are certain capabilities that users of SWIMA might require but that are beyond the scope of SWIMA itself and need to be addressed by other standards.

Confidentiality: SWIMA does not define a mechanism for confidentiality, nor is confidentiality automatically provided by using the PA-TNC interface. In the NEA architecture, confidentiality is generally provided by the underlying transport protocols, such as the PT binding to TLS [RFC6876] or PT-EAP (Posture Transport for Tunneled Extensible Authentication Protocol (EAP) Methods) [RFC7171]; see Section 7 for more information on related standards. The information conveyed by SWIMA is often sensitive in nature for both security (Section 8) and privacy (Section 9) reasons. Those who implement SWIMA need to ensure that appropriate NEA transport mechanisms are employed to meet confidentiality requirements.

2.5. Assumptions

The Posture Broker Client and Posture Broker Server are assumed to provide reliable delivery for PA-TNC messages and attributes sent between the SWIMA-PCs and the SWIMA-PVs. "Reliable delivery" means that either a message is delivered or the sender is made aware of the delivery failure. In the event that reliable delivery cannot be provided, some SWIMA features, primarily subscriptions, might not behave as expected.

2.6. Assumptions Not Made

This specification explicitly does not assume that software inventory information exchanges reflect the software installation state of the endpoint. This specification does not attempt to detect when the endpoint is providing false information, either through malice or error, but instead focuses on correctly and reliably providing the reported Software Inventory Evidence Collection to the NEA Server. Tools that employ the SWIMA standard can include methods to help verify the accuracy of reports, but how those tools do so is beyond the scope of this specification.

Similarly, this specification makes no assumption about the completeness of the Software Inventory Evidence Collection’s coverage of the total set of software installed on the endpoint. It is possible, and even likely, that some installed software is not represented by a record in an endpoint’s Software Inventory Evidence Collection. Instead, SWIMA ensures that what does get reported is reported consistently and that the software products that are reported can be reliably tracked.

See Section 8 for more on this security consideration.

3. System Requirements

SWIMA facilitates the exchange of software inventory and event information. Specifically, each application supporting SWIMA includes a component known as the SWIMA-PC that receives SWIMA attributes. The SWIMA-PC is also responsible for sending appropriate SWIMA attributes back to the SWIMA-PV in response. This section outlines what software inventories and events are and the requirements on SWIMA-PCs and SWIMA-PVs in order to support the stated use cases of this specification.

3.1. Data Sources

The records in an endpoint’s Software Inventory Evidence Collection come from one or more "sources". A source represents one collection of software inventory information about the endpoint. Examples of sources include, but are not limited to, ISO SWID tags deposited on the filesystem and collected therefrom, information derived from package managers, and the output of software inventory-scanning tools.

There is no expectation that any one source of inventory information will have either perfect or complete software inventory information. For this reason, this specification supports the simultaneous use of multiple sources of software inventory information. Each source might have its own "sphere of expertise" and report the software within that sphere. For example, a package manager would have an excellent understanding of the software that it managed but would not necessarily have any information about software installed via other means.

A SWIMA-PC is not required to utilize every possible source of software information on its endpoint. Some SWIMA-PCs might be explicitly tied only to one or a handful of software inventory sources, or a given SWIMA-PC could be designed to dynamically accommodate new sources. For all software inventory evidence sources that a particular SWIMA-PC supports, it MUST completely support all requirements of this specification with regard to those sources. A potential source that cannot support some set of required functionality (e.g., it is unable to monitor the software it reports for change events, as discussed in Section 3.6) MUST NOT be used as a source of endpoint software inventory information, even if it could provide some information. In other words, a source either supports full functionality as described in this specification or cannot be used at all. In the event that a previously used source becomes unavailable, this would be treated as a discontinuity in the SWIMA-PC’s reporting. Section 3.7.1 describes how to use changes in the Event Identifier (EID) Epoch value to indicate a reporting discontinuity.

When sending information about installed software, the SWIMA-PC MUST include the complete set of relevant data from all supported sources of software inventory evidence. In other words, sources need to be used consistently. This is because if a particular source is included in an initial inventory but excluded from a later inventory, the SWIMA-PV receiving this information might reasonably conclude that the software reported by that source was no longer installed on the endpoint. As such, it is important that all supported sources be used every time the SWIMA-PC provides information to a SWIMA-PV.

Note that if a SWIMA-PC collects data from multiple sources, it is possible that some software products might be "double counted". This can happen if two or more sources of inventory evidence provide a record for a single installation of a software product. When a SWIMA-PC reports information or records events from multiple inventory evidence sources, it MUST use the information those sources provide, rather than attempt to perform some form of reduction. In other words, if multiple sources report records corresponding to a single installation of a software product, all such records from each source are required to be part of the SWIMA-PC's processing even if this might lead to multiple reporting, and the SWIMA-PC is not to ignore some records to avoid such multiple reporting.

All inventory records reported by a SWIMA-PC include a Source Identifier linking them to a particular source. Source Identifiers are discussed more in Section 3.4.5. As discussed in that section, Source Identifiers can help consumers of SWIMA data identify cases of multiple reporting.

3.2. Data Models

SWIMA conveys records about software presence from a SWIMA-PC to a SWIMA-PV. SWIMA does not manage the actual generation or collection of such records on the endpoint. As a result, information available to SWIMA-PCs might come in a variety of formats, and a SWIMA-PC could have little control over the format of the data made available to it. Because of this, SWIMA places no constraints on the format of these generated records and supports an open set of record formats by which installed software instances can be described. The following terms are used in this document:

- Data model - The format used to structure data within a given record. SWIMA does not constrain the data models it conveys.

- Record - A populated instance of some data model that describes a software product.

Do not confuse the "data model" described here with the structure of the SWIMA messages and attributes used to convey information between SWIMA-PVs and SWIMA-PCs. The SWIMA standard dictates the structure of its messages and attributes. Some attributes, however, have specific fields used to convey inventory records, and those fields support an extensible list of data models for their values. In other words, SWIMA data models provide an extension point within SWIMA attributes that allows the structure of inventory records to evolve.

The data model used to structure software inventory information has very little impact on the behavior of the components defined in this specification. The SWIMA-PV has no dependency on the data model of records conveyed in SWIMA messages. For this reason, it MUST NOT reject a message or respond with a PA-TNC Error due to the data model used to structure records in attributes it receives. Similarly, it MUST NOT reject a message or respond with a PA-TNC Error if a record fails to comply with a stated format, unless that failure prevents correct parsing of the attribute itself. In short, the record bodies are effectively treated as "black boxes" by the SWIMA-PV. (Note that the SWIMA-PV might serve as the "front end" of other functionality that does have a dependency on the data model used to structure software information, but any such dependency is beyond the scope of this specification and needs to be addressed outside the behaviors specified in this document. This specification is only concerned with the collection and delivery of software inventory information; components that consume and use this information are a separate concern.)

The SWIMA-PC does have one functional dependency on the data models used in the software records it delivers, but only insofar as it is required to deterministically create a Software Identifier (described in Section 3.4.1) based on each record it delivers. The SWIMA-PC MUST be able to generate a Software Identifier for each record it delivers, and if the SWIMA-PC cannot do so, it cannot deliver the record. All SWIMA-PCs MUST at least be able to generate Software Identifiers for the data model types specified in Section 6 of this document. A SWIMA-PC MAY include the ability to generate Software Identifiers for other data model types and thus be able to support them as well.

3.3. Basic Attribute Exchange

In the most basic exchange supported by this specification, a SWIMA-PV sends a request to the SWIMA-PC, requesting some type of information about the endpoint's software inventory. This simple exchange is shown in Figure 2.

```
+-------------+                          +--------------+
|  SWIMA-PC   |                          |   SWIMA-PV   |  Time
+-------------+                          +--------------+   |
      |                                         |           |
      |<------------SWIMA Request---------------|           |
      |                                         |           |
      |-------------SWIMA Response------------->|           |
      |                                         |           V
```

Figure 2: Basic SWIMA Attribute Exchange

Upon receiving such a SWIMA Request from the SWIMA-PV, the SWIMA-PC is expected to collect all the relevant software inventory information from the endpoint's Software Inventory Evidence Collection and place it within its response attribute.

SWIMA-PVs MUST discard, without error, any SWIMA Response attributes that they receive for which they do not know the SWIMA Request parameters that led to this SWIMA Response. This is due to the fact that the SWIMA Request includes parameters that control the nature of the response (as will be described in the following sections); without knowing those parameters, the SWIMA Response cannot be reliably interpreted. Each SWIMA Request includes a Request ID, which is echoed in any SWIMA Response to that request and allows matching of responses to requests. See Section 5.5 for more on Request IDs. Receiving an unsolicited SWIMA Response attribute will most often happen when a NEA Server has multiple SWIMA-PVs; one SWIMA-PV sends a SWIMA Request, but unless exclusive delivery [RFC5793] is set by the sender and honored by the recipient, multiple SWIMA-PVs will receive copies of the resulting SWIMA Response. In this case, the SWIMA-PV that didn't send the SWIMA Request would lack the context necessary to correctly interpret the SWIMA Response it received and would simply discard it. Note, however, that proprietary measures might allow a SWIMA-PV to discover the SWIMA Request parameters for a SWIMA Response even if that SWIMA-PV did not send the given SWIMA Request. As such, there is no blanket requirement for a SWIMA-PV to discard all SWIMA Responses to SWIMA Requests that the SWIMA-PV did not generate itself -- only that SWIMA-PVs are required to discard SWIMA Responses for which they cannot get the necessary context to interpret.

In the case that it is possible to do so, the SWIMA-PC SHOULD send its SWIMA Response attribute to the SWIMA-PV that requested it, using exclusive delivery as described in Section 4.5 of "PB-TNC: A Posture Broker (PB) Protocol Compatible with Trusted Network Connect (TNC)" [RFC5793]. Exclusive delivery requests that only the sender of the SWIMA Request be the receiver of the resulting SWIMA Response. Note, however, that PB-TNC does not require the recipient to honor the exclusive delivery flag in messages that it receives, so setting the flag cannot be guaranteed to prevent a SWIMA-PV from receiving unsolicited SWIMA Responses.

Note that, in the case that a single endpoint hosts multiple SWIMA-PCs, a single SWIMA Request could result in multiple SWIMA Responses. SWIMA-PVs need to handle such an occurrence without error.

All numeric values sent in SWIMA messages are sent in network (big endian) byte order.

3.4. Core Software-Reporting Information

Different parameters in the SWIMA Request can influence what information is returned in the SWIMA Response. However, while each SWIMA Response provides different additional information about this installed software, the responses all share a common set of fields that support reliable software identification on an endpoint. These fields include Software Identifiers, Data Model Type, Record Identifiers, Software Locators, and Source Identifiers. These fields are present for each reported piece of software in each type of SWIMA Response. The following sections examine these information types in more detail.

3.4.1. Software Identifiers

A Software Identifier uniquely identifies a specific version of a specific software product. The SWIMA standard does not dictate the structure of a Software Identifier (beyond stating that it is a string) or define how it is created. Instead, each data model described in the "Software Data Model Types" registry (Section 10.5) includes its own rules for how a Software Identifier is created based on a record in the endpoint’s Software Inventory Evidence Collection expressed in that data model. Other data models will have their own procedures for the creation of associated Software Identifiers. Within SWIMA, the Software Identifier is simply an opaque string, and there is never any need to unpack any information that might be part of that identifier.

A Software Identifier is a fraction of the size of the inventory record from which it is derived. For this reason, it is often more efficient to collect full inventories using Software Identifiers and only collect full records when necessary using targeted requests. For some combinations of data models and sources, the full record might never be necessary, as the identifier can be directly correlated to the contents of the full record. This is possible with authoritative SWID tags, since these tags always have the same contents and thus a Software Identifier derived from these tags can be used as a lookup to a local copy of the full tag. For other combinations of source and data model, a server might not be able to determine the specific software product and version associated with the identifier without requesting the delivery of the full record. However, even in those cases, downstream consumers of this information might never need the full record as long as the Software Identifiers they receive can be tracked reliably. A SWIMA-PV can use Software Identifiers to track the presence of specific software products on an endpoint over time in a bandwidth-efficient manner.

There are two important limitations of Software Identifiers to keep in mind:

1. The identifiers do not necessarily change when the associated record changes. In some situations, a record in the endpoint’s Software Inventory Evidence Collection will change due to new information becoming available or in order to correct prior errors in that information. Such changes might or might not result in changes to the Software Identifier, depending on the nature of the changes and the rules governing how Software Identifiers are derived from records of the appropriate data model.

2. It is possible that a single software product is installed on a single endpoint multiple times. If these multiple installation instances are reported by the same source using the same data format, then this can result in identical Software Identifiers for both installation instances. In other words, Software Identifiers might not uniquely identify installation instances; they are just intended to uniquely identify software products (which might have more than one installation instance). Instead, to reliably distinguish between multiple instances of a single software product, one needs to make use of Record Identifiers as described in Section 3.4.3.

3.4.2. Data Model Type

The Data Model Type consists of two fields: Data Model Type PEN and Data Model Type. ("PEN" stands for "Private Enterprise Number".) The combination of these fields is used to identify the type of data model of the associated software inventory record. The data model is significant not only because it informs the recipient of the data model of the associated record but also because the process for generating the Software Identifier for the record depends on the record’s data model. Clearly identifying the type of data model from which the Software Identifier was derived thus provides useful context for that value.

The PEN identifies the organization that assigns meaning to the Data Model Type field value. PENs are managed by IANA in the "Private Enterprise Numbers" registry. PENs allow vendors to designate their own set of data models for software inventory description. IANA reserves the PEN of 0x000000. Data Model Types associated with this PEN are defined in the "Software Data Model Types" registry; see Section 10.5 of this specification. Note that this IANA table reserves all values greater than or equal to 0xC0 (192) for local enterprise use. This means that local enterprises can use custom data formats and indicate them with the IANA PEN and a Data Model Type value between 0xC0 and 0xFF, inclusive. Those enterprises are responsible for configuring their SWIMA-PCs to correctly report those custom data models.

3.4.3. Record Identifiers

A Record Identifier is a 4-byte unsigned integer that is generated by the SWIMA-PC and is uniquely associated with a specific record within the endpoint’s Software Inventory Evidence Collection. The SWIMA-PC MUST assign a unique identifier to each record when it is added to the endpoint’s Software Inventory Evidence Collection. The Record Identifier SHOULD remain unchanged if that record is modified. However, it is recognized that, in some circumstances, record modification might be hard to distinguish from record deletion followed by creation of a new record. For this reason, retaining a constant Record Identifier across a record modification is recommended but not required. Similarly, in the case that the software product associated with a record is moved, ideally the Record Identifier for the record of the moved software will remain unchanged, reflecting that it represents the same software product instance, albeit in a new location. However, this level of tracking could prove difficult to achieve and is not required. The SWIMA-PC might wish to assign Record Identifiers sequentially, but any scheme is acceptable, provided that no two records receive the same identifier.

Servers can use Record Identifiers to distinguish between multiple instances of a single software product installed on an endpoint. Since each installation instance of a software product is associated with a separate record, servers can use the Record Identifier to distinguish between instances. For example, if an event is reported (as described in Section 3.7), the Record Identifier will allow the server to discern which instance of a software product is involved.

3.4.4. Software Locators

In addition to the need to identify what software products are on an endpoint, some processes that use inventory information also need to know where software is located on the endpoint. This information might be needed to direct remediation actions or similar processes. For this reason, every reported software product also includes a Software Locator to identify where the software is installed on the endpoint.

If the location is not provided directly by the data source, the SWIMA-PC is responsible for attempting to determine the location of the software product. The "location" of a product SHOULD be the directory in which the software product’s executables are kept. The SWIMA-PC MUST be consistent in reporting the location of a software product. In other words, assuming that a software product has not moved, the SWIMA-PC cannot use one location in one report and a different location for the same software product in another. (If a software product has moved, the Software Locator needs to reflect the new location.)

The location is expressed as a URI string. The string MUST conform to URI syntax requirements [RFC3986]. The URI scheme describes the context of the described location. For example, in most cases the location of the installed software product will be expressed in terms of its path in the filesystem. For such locations, the location URI scheme MUST be "file", and the URI MUST conform to the "file" URI scheme standard [RFC8089], including the percent-encoding of whitespace and other special characters. It is possible that other schemes could be used to represent other location contexts. Apart from specifying the use of the "file" scheme, this specification does not identify other schemes or define their use. When representing software products in other location contexts, tools MUST be consistent in their use of schemes, but the exact schemes are not normatively defined here. SWIMA implementations are not limited to the IANA list of URI schemes <https://www.iana.org/assignments/uri-schemes/> and can define new schemes to support other types of application locations.

It is possible that a SWIMA-PC is unable to determine the location of a reported software product. In this case, the SWIMA-PC MUST provide a zero-length Software Locator.

3.4.5. Source Identifiers

All SWIMA-PCs MUST track the source of each piece of software information they report. Each time a SWIMA-PC gets information to send to a given SWIMA-PV from a new source (from the perspective of that SWIMA-PV), the SWIMA-PC MUST assign that source a Source Identifier, which is an 8-bit unsigned integer. Each item reported includes the number of the Source Identifier for the source that provided that information. All information that is provided by that source MUST be delivered with this same Source Identifier. This MUST be done for each source used. If the SWIMA-PC is ever unclear as to whether a given source is new or not, it MUST assume that this is a new source and assign it a new Source Identifier. Source Identifier numbers do not need to be assigned sequentially. SWIMA does not support the presence of more than 256 sources, as the chance that a single endpoint will have more than 256 methods of collecting inventory information is vanishingly small. All possible values between 0 and 255 are valid; there are no reserved Source Identifier numbers.

Source Identifiers can help with (although will not completely eliminate) the challenges posed by multiple reporting of a single software instance. If multiple sources each report the same type of software product once, there is most likely a single instance of that product installed on the endpoint, which each source has detected independently. On the other hand, if multiple instances are reported by a single source, this almost certainly means that there are actually multiple instances that are being reported.

The SWIMA-PC is responsible for tracking associations between Source Identifiers and the given data source. This association MUST remain consistent with regard to a given SWIMA-PV while there is an active PB-TNC session with that SWIMA-PV. The SWIMA-PC MAY have a different Source Identifier association for different SWIMA-PVs. Likewise, the SWIMA-PC MAY change the Source Identifier association for a given SWIMA-PV if the PB-TNC session terminates. However, implementers of SWIMA-PCs will probably find it easier to manage associations by maintaining the same association for all SWIMA-PVs and across multiple sessions.

Of special note, event records reported from the SWIMA-PC’s event log (discussed in Section 3.7) also need to be sent with their associated data source. The Source Identifier reported with events MUST be the current (i.e., at the time the event is sent) Source Identifier associated with the data source that produced the event, regardless of how long ago that event occurred. Event logs are likely to persist far longer than a single PB-TNC session. SWIMA-PCs MUST ensure that each event can be linked to the appropriate data source, even if the Source Identifiers used when the event was created have since been reassigned. In other words, when sending an event, it needs to be sent with the Source Identifier currently linked to the data source that produced it, regardless of whether a different Source Identifier would have been associated with the event when the event was first created.

Note that the Source Identifier is primarily used to support recognition, rather than identification, of sources. That is to say, a Source Identifier can tell a recipient that two events were reported by the same source, but the number will not necessarily help that recipient determine which source was used. Moreover, different SWIMA-PCs will not necessarily use the same Source Identifiers for the same sources. SWIMA-PCs MUST track the assignment of Source Identifiers to ensure consistent application thereof. SWIMA-PCs MUST also track which Source Identifiers have been used with each SWIMA-PV with which they communicate.

3.4.6. Using Software and Record Identifiers in SWIMA Attributes

A SWIMA attribute reporting an endpoint’s Software Inventory Evidence Collection always contains the Software Identifiers associated with the identified software products. A SWIMA attribute might or might not also contain copies of Software Inventory Evidence Records. The attribute exchange is identical to the diagram shown in Figure 2, regardless of whether Software Inventory Evidence Records are included. The SWIMA Request attribute indicates whether the response is required to include Software Inventory Evidence Records. Excluding Software Inventory Evidence Records can reduce the attribute size of the response by multiple orders of magnitude when compared to sending the same inventory with full records.

3.5. Targeted Requests

Sometimes a SWIMA-PV does not require information about every piece of software on an endpoint but only needs to receive updates about certain software instances. For example, enterprise endpoints might be required to have certain software products installed and to keep these updated. Instead of requesting a complete inventory just to see if these products are present, the SWIMA-PV can make a "targeted request" for the software in question.

Targeted requests follow the same attribute exchange as the exchange described in Figure 2. The SWIMA-PV targets its request by providing one or more Software Identifiers in its SWIMA Request attribute. The SWIMA-PC MUST then limit its response to contain only records that match the indicated Software Identifier(s). This allows the network exchange to exclude information that is not relevant to a given policy question, thus reducing unnecessary bandwidth consumption. The SWIMA-PC’s response might or might not include Software Inventory Evidence Records, depending on the parameters of the SWIMA Request.

Note that targeted requests identify the software relevant to the request only through Software Identifiers. This specification does not support arbitrary, parameterized querying of records. For example, one cannot request all records from a certain software publisher or all records created by a particular data source. Targeted requests only allow a requester to request specific software (as identified by their Software Identifiers) and receive a response that is limited to the named software.

There is no assumption that a SWIMA-PC will recognize "synonymous records" -- that is, records from different sources for the same software. Recall that different sources and data models may use different Software Identifier strings for the same software product. The SWIMA-PC returns only records that match the Software Identifiers in the SWIMA Request, even if there might be other records in the endpoint’s Software Inventory Evidence Collection for the same software product. This is necessary because SWIMA-PCs might not have the ability to determine that two Software Identifiers refer to the same product.

A targeted SWIMA Request attribute does not specify Record Identifiers or Software Locators. The response to a targeted request MUST include all records associated with the named Software Identifiers, including the case where there are multiple records associated with a single Software Identifier.

SWIMA-PCs MUST accept targeted requests and process them correctly as described above. SWIMA-PVs MUST be capable of making targeted requests and processing the responses thereto.

3.6. Monitoring Changes in an Endpoint’s Software Inventory Evidence Collection

The software collection on an endpoint is not static. As software is installed, uninstalled, patched, or updated, the Software Inventory Evidence Collection is expected to change to reflect the new software state on the endpoint. Different data sources might update the evidence collection at different rates. For example, a package manager might update its records in the Software Inventory Evidence Collection immediately whenever it is used to add or remove a software product. By contrast, sources that perform periodic examination of the endpoint would likely only update their records in the Software Inventory Evidence Collection after each examination.

All SWIMA-PCs MUST be able to detect changes to the Software Inventory Evidence Collection. Specifically, SWIMA-PCs MUST be able to detect:

- The creation of records

- The deletion of records

- The alteration of records

An "alteration" is anything that modifies the contents of a record (or would modify it, if the record is dynamically generated on demand) in any way, regardless of whether the change is functionally meaningful.

SWIMA-PCs MUST detect such changes to the endpoint’s Software Inventory Evidence Collection in close to real time (i.e., within seconds) when the SWIMA-PC is operating. In addition, in the case where there is a period during which the SWIMA-PC is not operating, the SWIMA-PC MUST be able to determine the net change to the endpoint’s Software Inventory Evidence Collection over the period it was not operational. Specifically, the "net change" represents the difference between (1) the state of the endpoint’s Software Inventory Evidence Collection when the SWIMA-PC was last operational and monitoring its state and (2) the state of the endpoint’s Software Inventory Evidence Collection when the SWIMA-PC resumed operation. Note that a net change might not reflect the total number of change events over this interval. For example, if a record was altered three times during a period when the SWIMA-PC was unable to monitor for changes, the net change of this interval might only note that there was an alteration to the record, but not how many individual alteration events occurred. It is sufficient for a SWIMA-PC’s determination of a net change to note that there was a difference between the earlier and current state, rather than to enumerate all the individual events that allowed the current state to be reached.

The SWIMA-PC MUST assign a time to each detected change in the endpoint’s Software Inventory Evidence Collection. These timestamps correspond to the SWIMA-PC’s best understanding as to when the detected change occurred. For changes to the endpoint’s Software Inventory Evidence Collection that occur while the SWIMA-PC is operating, the SWIMA-PC ought to be able to assign a time to the event that is accurate to within a few seconds. For changes to the endpoint's Software Inventory Evidence Collection that occur while the SWIMA-PC is not operational, upon becoming operational the SWIMA-PC needs to make a best guess as to the time of the relevant events (possibly by looking at timestamps on files), but these values might be off. In the case of dynamically generated records, the time of change is the time at which the data from which the records are generated changes, not the time at which a changed record is generated. For example, if records are dynamically generated based on data in an RPM database (<http://rpm.org/>), the time of change would be when the RPM database changed.

With regard to deletions of records, the SWIMA-PC needs to detect the deletion of a given record and MUST retain a copy of the full deleted record along with the associated Record Identifier and Software Locator so that the record and associated information can be provided to the SWIMA-PV upon request. This copy of the record MUST be retained for a reasonable amount of time. Vendors and administrators determine what "reasonable" means, but a copy of the record SHOULD be retained for as long as the event recording the deletion of the record remains in the SWIMA-PC's event log (as described in Section 3.7). This is recommended, because as long as the event is in the SWIMA-PC's event log the SWIMA-PC might send a change event attribute (described in Section 3.7) that references this record, and a copy of the record is needed if the SWIMA-PV wants a full copy of the relevant record. In the case that a SWIMA-PC is called upon to report a deletion event that is still in the event log but where the record itself is no longer available, the SWIMA-PC will still return an entry corresponding to the deletion event, but the field of that entry that would normally contain the full copy of the record SHOULD be zero-length.

With regard to alterations to a record, SWIMA-PCs MUST detect any alterations to the contents of a record. Alterations need to be detected even if they have no functional impact on the record. A good guideline is that any alteration to a record that might change the value of a hash taken on the record's contents needs to be detected by the SWIMA-PC. A SWIMA-PC might be unable to distinguish modifications to the contents of a record from modifications to the metadata that the filesystem associates with the record. For example, a SWIMA-PC might use the "last modification" timestamp as an indication of alteration to a given record, but a record's last modification time can change for reasons other than modifications to the record's contents. A SWIMA-PC is still considered compliant with this specification if it also reports metadata change events that do not change the record itself as alterations to the record. In other words, while SWIMA-PC implementers are encouraged to exclude modifications that do not affect the bytes within the record, discriminating between modifications to file contents and changes to file metadata can be difficult and time consuming on some systems. As such, as long as the alterations detected by a SWIMA-PC always cover all modifications to the contents of a record, the SWIMA-PC is considered compliant even if it also registers alterations that do not modify the contents of a record as well. When recording an alteration to a record, the SWIMA-PC is only required to note that an alteration occurred. The SWIMA-PC is not required to note or record how the record was altered, nor is it possible to include such details in SWIMA attributes reporting the change to a SWIMA-PV. There is no need to retain a copy of the original record prior to the alteration.

When a record changes, it SHOULD retain the same Record Identifier. The Software Locator might or might not change, depending on whether the software changed locations during the changes that led to the record change. A record change MUST retain the same Software Identifier. This means that any action that changes a software product (e.g., application of a patch that results in a change to the product’s version) MUST NOT be reflected by a record change but instead MUST result in the deletion of the old record and the creation of a new record. This reflects the requirement that a record in the endpoint’s Software Inventory Evidence Collection correspond directly with an instance of a specific software product.

3.7. Reporting Change Events

As noted in Section 3.6, SWIMA-PCs are required to detect changes to the endpoint’s Software Inventory Evidence Collection (creation, deletion, and alteration) in near real time while the SWIMA-PC is operational, and a given SWIMA-PC MUST be able to account for any net change to the endpoint’s Software Inventory Evidence Collection that occurs when the SWIMA-PC is not operational. However, to be of use to the enterprise, the NEA Server needs to be able to receive these events and be able to understand how new changes relate to earlier changes. In SWIMA, this is facilitated by reporting change events. All SWIMA-PCs MUST be capable of receiving requests for change events and sending change event attributes. All SWIMA-PVs MUST be capable of requesting and receiving change event attributes.

3.7.1. Event Identifiers

To be useful, change events need to be correctly ordered. The ordering of events is facilitated by two pieces of information: an Event Identifier (EID) value and an EID Epoch value.

An EID is a 4-byte unsigned integer that the SWIMA-PC assigns sequentially to each observed event (whether detected in real time or deduced by looking for net changes over a period of SWIMA-PC inactivity). All EIDs exist within the context of some "EID Epoch", which is also represented as a 4-byte unsigned integer. EID Epochs are used to ensure synchronization between the SWIMA-PC and any SWIMA-PVs with which it communicates. EID Epoch values MUST be generated in such a way as to minimize the chance that an EID Epoch will be reused, even in the case where the SWIMA-PC reverts to an earlier state. For this reason, sequential EID Epochs are discouraged, since loss of state could result in value reuse. There are multiple reasons that a SWIMA-PC might need to deliberately reset its EID counter, including exhaustion of available EID values, the need to purge entries from the event log to recover memory, or corruption of the event log. In all cases where a SWIMA-PC needs to reset its EID counter, a new EID Epoch MUST be selected.

Within an Epoch, EIDs MUST be assigned sequentially, so that if a particular event is assigned an EID of N, the next observed event is given an EID of N+1. In some cases, events might occur simultaneously, or the SWIMA-PC might not otherwise be able to determine an ordering for events. In these cases, the SWIMA-PC creates an arbitrary ordering of the events and assigns EIDs according to this ordering. Two change events MUST NOT ever be assigned the same EID within the same EID Epoch. No meaningful comparison can be made between EID values of different Epochs.

The EID value of 0 is reserved and MUST NOT be associated with any event. Specifically, an EID of 0 in a SWIMA Request attribute indicates that a SWIMA-PV wants an inventory response rather than an event response, while an EID of 0 in a SWIMA Response is used to indicate the initial state of the endpoint’s Software Inventory Evidence Collection prior to the observation of any events. Thus, the very first recorded event in a SWIMA-PC’s records within an EID Epoch MUST be assigned a value of 1. Note that EID and EID Epoch values are assigned by the SWIMA-PC without regard to whether events are being reported to one or more SWIMA-PVs. The SWIMA-PC records events and assigns EIDs during its operation. All SWIMA-PVs that request event information from the SWIMA-PC will have those requests served from the same event records and thus will see the same EIDs and EID Epochs for the same events.

If a SWIMA-PC uses multiple sources, a SWIMA-PC’s assignment of EIDs MUST reflect the presence and order of all events on the endpoint (at least for supported sources), regardless of the source. This means that if source A experiences an event and then source B experiences two events, and then source A experiences another two events, the SWIMA-PC is required to capture five events with consecutive EID values reflecting the order in which the events occurred.

The SWIMA-PC MUST ensure that there is no coverage gap (i.e., change events that are not recorded in the SWIMA-PC’s records) in its change event records. This is necessary because a coverage gap might give a SWIMA-PV a false impression of the endpoint’s state. For example, if a SWIMA-PV saw an event indicating that a particular record had been added to the endpoint’s Software Inventory Evidence Collection but did not see any subsequent events indicating that the record in question had been deleted, it might reasonably assume that this record was still present and thus that the indicated software was still installed (assuming that the Epoch has not changed). If there is a coverage gap in the SWIMA-PC’s event records, however, this assumption could be false. For this reason, the SWIMA-PC’s event records MUST NOT contain gaps. In the case where there are periods where it is possible that changes occurred without the SWIMA-PC detecting or recording them, the SWIMA-PC MUST either (1) compute a net change and update its event records appropriately or (2) pick a new EID Epoch to indicate a discontinuity with previous event records.

Within a given Epoch, once a particular event has been assigned an EID, this association MUST NOT be changed. That is, within an Epoch, once an EID is assigned to an event, that EID cannot be reassigned to a different event, and the event cannot be assigned a different EID. When the SWIMA-PC’s Epoch changes, all of these associations between EIDs and events are cancelled, and EID values once again become free for assignment.

3.7.2. Core Event-Tracking Information

Whether reporting events or full inventories, it is important to know how the reported information fits into the overall timeline of change events. This is why all SWIMA Response attributes include fields to place that response within the sequence of detected events. Specifically, all SWIMA Responses include a Last EID field and an EID Epoch field. The EID Epoch field identifies the EID Epoch in which the SWIMA Response was sent. If the SWIMA Response is reporting events, all reported events occurred within the named EID Epoch. The Last EID (which is also always from the named EID Epoch) indicates the EID of the last recorded change event at the time that the SWIMA

Response was sent. These two fields allow any response to be placed in the context of the complete set of detected change events within a given EID Epoch.

3.7.3. Updating Inventory Knowledge Based on Events

Modern endpoints can have hundreds of software products installed, most of which are unlikely to change from one day to the next. As such, instead of exchanging a complete list of an endpoint’s inventory on a regular basis, one might wish to only identify changes since some earlier known state of this inventory. This is readily facilitated by the use of EIDs to place change events in a context relative to the earlier state.

As noted above, every SWIMA Response sent by a SWIMA-PC to a SWIMA-PV (as described in Sections 3.3 through 3.5) includes the EID Epoch and EID of the last event recorded prior to that response being compiled. This allows the SWIMA-PV to place all subsequently received event records in context relative to this SWIMA Response attribute (since the EIDs represent a total ordering of all changes to the endpoint’s Software Inventory Evidence Collection). Specifically, a SWIMA-PV (or, more likely, a database that collects and records its findings) can record an endpoint’s full inventory and also the EID and Epoch of the most recent event reflected at the time of that inventory. From that point on, if change events are observed, the attribute describing these events indicates the nature of the change, the affected records, and the order in which these events occurred (as indicated by the sequential EIDs). Using this information, any remote record of the endpoint’s Software Inventory Evidence Collection can be updated appropriately.

3.7.4. Using Event Records in SWIMA Attributes

A SWIMA-PV MUST be able to request a list of event records instead of an inventory. The attribute flow in such an exchange looks the same as the basic flow shown in Figure 2. The only difference is that in the SWIMA Request attribute the SWIMA-PV provides an EID other than 0. (An EID value of 0 in a SWIMA Request represents a request for an inventory.) When the SWIMA-PC receives such a request, instead of identifying records from the endpoint’s Software Inventory Evidence Collection, it consults its list of detected changes. The SWIMA-PC MUST add an event record to the SWIMA Response attribute for each recorded change event with an EID greater than or equal to the EID in the SWIMA Request attribute (although the targeting of requests, as described in the next paragraph, might limit this list). A list of event records MUST only contain events with EIDs that all come from the current Epoch.

SWIMA-PVs can target requests for event records by including one or more Software Identifiers, as described in Section 3.5, in the SWIMA Request that requests an event record list. A targeted request for event records is used to indicate that only events affecting software that matches one of the provided Software Identifiers are to be returned. Specifically, in response to a targeted request for event records, the SWIMA-PC MUST exclude any event records that are less than the indicated EID (within the current EID Epoch) and exclude any event records where the affected software does not match one of the provided Software Identifiers. This might mean that the resulting list of event records sent in the response attribute does not provide a continuous sequence of EIDs. Both SWIMA-PCs and SWIMA-PVs MUST support targeted requests for event records.

3.7.5. Partial and Complete Lists of Event Records in SWIMA Attributes

Over time, a SWIMA-PC might record a large number of change events. If a SWIMA-PV requests all change events covering a long period of time, the resulting SWIMA Response attribute might be extremely large, especially if the SWIMA-PV requests the inclusion of Software Inventory Evidence Records in the response. In the case that the resulting attribute is too large to send (because it exceeds either (1) the 4 GB attribute size limit imposed by the PA-TNC specification or (2) some smaller size limit imposed on the SWIMA-PC), the SWIMA-PC MAY send a partial list of event records back to the SWIMA-PV.

The generation of a partial list of events in a SWIMA Response attribute requires the SWIMA-PC to identify a "consulted range" of EIDs. A consulted range is the set of event records that are examined for inclusion in the SWIMA Response attribute and that are included in that attribute if applicable. Recall that if a SWIMA Request is targeted, only event records that involve the indicated software would be applicable. (See Section 3.5 for more on targeted requests.) If a request is not targeted, all event records in the consulted range are applicable and are included in the SWIMA Response attribute.

The lower bound of the consulted range MUST be the EID provided in the SWIMA Request. (Recall that a SWIMA-PV indicates a request for event records by providing a non-zero EID value in the SWIMA Request. See Section 3.7.4.) The upper bound of the consulted range is the EID of the latest event record (as ordered by EID values) that is included in the SWIMA Response attribute if it is applicable to the request. The EID of this last event record is called the "Last Consulted EID". The SWIMA-PC chooses this Last Consulted EID based on the size of the event record list it is willing to provide to the SWIMA-PV.

A partial result list MUST include all applicable event records within the consulted range. This means that for any applicable event record (i.e., any event record in a non-targeted request or any event record associated with software matching a requested Software Identifier in a targeted request) whose EID is greater than or equal to the EID provided in the SWIMA Request and whose EID is less than or equal to the Last Consulted EID, that event record MUST be included in the SWIMA Response conveying this partial list of event records. This ensures that every partial list of event records is always complete within its indicated range. Remember that for targeted requests, "complete" doesn’t mean that all EIDs between the range endpoints are present -- only that every matching EID between the range endpoints is included.

In addition to the EID Epoch and Last EID fields that are present in all SWIMA Responses, all SWIMA Response attributes that convey event records include a Last Consulted EID field. Note that if responding to a targeted SWIMA Request, the SWIMA Response attribute might not contain the event record whose EID matches the Last Consulted EID value. For example, that record might have been deemed inapplicable because it did not match the specified list of Software Identifiers in the SWIMA Request.

If a SWIMA-PV receives a SWIMA Response attribute where the Last EID and Last Consulted EID fields are identical, the SWIMA-PV knows that it has received a result list that is complete, given the parameters of the request, up to the present time.

On the other hand, if the Last EID is greater than the Last Consulted EID, the SWIMA-PV has received a partial result list. (The Last Consulted EID MUST NOT exceed the Last EID.) In this case, if the SWIMA-PV wishes to try to collect the rest of the partially delivered result list, it then sends a new SWIMA Request whose EID is one greater than the Last Consulted EID in the preceding response. Doing this causes the SWIMA-PC to generate another SWIMA Response attribute containing event records where the earliest reported event record is the one immediately after the event record with the Last Consulted EID (since EIDs are assigned sequentially). By repeating this process until it receives a SWIMA Response where the Last EID and Last Consulted EID are equal, the SWIMA-PV is able to collect all event records over a given range, even if the complete set of event records would be too large to deliver via a single attribute.

Implementers need to be aware that a SWIMA Request might specify an EID that is greater than the EID of the last event recorded by a SWIMA-PC. In accordance with the behaviors described in Section 3.7.4, a SWIMA-PC MUST respond to such a request with a SWIMA Response attribute that contains zero event records. This is because the SWIMA-PC has recorded no event records with EIDs greater than or equal to the EID in the SWIMA Request. In such a case, the Last Consulted EID field MUST be set to the same value as the Last EID field in this SWIMA Response attribute. This case is called out because the consulted range on a SWIMA-PC in such a situation is a negative range, where the "first" EID in the range (provided in the SWIMA Request) is greater than the "last" EID in the range (this being the EID of the last recorded event on the SWIMA-PC). Implementers need to ensure that SWIMA-PCs do not experience problems in such a circumstance.

Note that this specification only supports the returning of partial results when returning event records. There is no way to return a partial inventory list under this specification.

3.7.6. Synchronizing Event Identifiers and Epochs

Since EIDs are sequential within an Epoch, if a SWIMA-PV’s list of event records contains gaps in the EID values within a single Epoch, the SWIMA-PV knows that there are events that it has not accounted for. The SWIMA-PV can request either (1) a new event list to collect the missing events or (2) a full inventory to resync its understanding of the state of the endpoint’s Software Inventory Evidence Collection. In either case, after the SWIMA-PV’s record of the endpoint’s Software Inventory Evidence Collection has been updated, the SWIMA-PV can record the new latest EID value and track events normally from that point on.

If the SWIMA-PV receives any attribute from a SWIMA-PC where the EID Epoch differs from the EID Epoch that was used previously, then the SWIMA-PV or any entity using this information to track the endpoint’s Software Inventory Evidence Collection knows that there is a discontinuity in its understanding of the endpoint’s state. To move past this discontinuity and reestablish a current understanding of the state of the endpoint’s Software Inventory Evidence Collection, the SWIMA-PV needs to receive a full inventory from the endpoint. The SWIMA-PV cannot be brought in sync with the endpoint’s state through the collection of any set of event records in this situation. This is because it is not possible to account for all events on the SWIMA-PC since the previous Epoch was used: there is no way to query for EIDs from a previous Epoch. Once the SWIMA-PV has received a full inventory for the new Epoch, the SWIMA-PV records the latest EID reported in this new Epoch and can track further events normally.

A SWIMA-PC MUST NOT report events with EIDs from any Epoch other than the current EID Epoch. The SWIMA-PC MAY choose to purge all event records from a previous Epoch from memory after an Epoch change. Alternately, the SWIMA-PC MAY choose to retain some event records from a previous EID Epoch and assign them new EIDs in the current Epoch. However, in the case where a SWIMA-PC chooses the latter option it MUST ensure that the order of events according to their EIDs is unchanged and that there is no coverage gap between the first retained event recorded during the previous Epoch (now reassigned with an EID in the current Epoch) and the first event recorded during the current Epoch. In particular, the SWIMA-PC MUST ensure that all change events that occurred after the last recorded event from the previous Epoch are known and recorded. (This might not be possible if the Epoch change is due to state corruption on the SWIMA-PC.) A SWIMA-PC might choose to reassign EIDs to records from a preceding Epoch to create a "sliding window" of events, where each Epoch change represents a shift in the window of available events.

In the case where a SWIMA-PC suffers a crash and loses track of its current EID Epoch or current EID, then it MUST generate a new EID Epoch value and begin assigning EIDs within that Epoch. In this case, the SWIMA-PC MUST purge all event records from before the crash, as it cannot ensure that there is not a gap between the last of those records and the next detected event. The process for generating a new EID Epoch MUST minimize the possibility that the newly generated EID Epoch is the same as a previously used EID Epoch.

The SWIMA-PV will normally never receive an attribute indicating that the latest EID is less than the latest EID reported in a previous attribute within the same EID Epoch. If this occurs, the SWIMA-PC has suffered an error of some kind, possibly indicative of at least partial corruption of its event log. In this case, the SWIMA-PV MUST treat the situation as if there was a change in Epoch and treat any local copy of the endpoint’s Software Inventory Evidence Collection as being out of sync until a full inventory can be reported by the SWIMA-PC. The SWIMA-PV SHOULD log the occurrence so the SWIMA-PC can be examined to ensure that it is now operating properly.

3.8. Subscriptions

Thus far, all attribute exchanges discussed assume that a SWIMA-PV sent a SWIMA Request attribute and the SWIMA-PC is providing a direct response to that request. SWIMA also supports the ability of a SWIMA-PC to send a SWIMA Response to the SWIMA-PV in response to observed changes in the endpoint’s Software Inventory Evidence Collection, instead of in direct response to a SWIMA Request. An agreement by a SWIMA-PC to send content when certain changes to the endpoint’s Software Inventory Evidence Collection are detected is referred to in this specification as a "subscription", and the SWIMA-PV that receives this content is said to be "subscribed to" the given SWIMA-PC. All SWIMA-PCs and SWIMA-PVs MUST support the use of subscriptions.

3.8.1. Establishing Subscriptions

A SWIMA-PV establishes a subscription on a particular SWIMA-PC by sending a SWIMA Request attribute with the Subscribe flag set. The SWIMA Request attribute is otherwise identical to the SWIMA Requests discussed in previous sections. Specifically, such a SWIMA Request might or might not request the inclusion of Software Inventory Evidence Records, might or might not be targeted, and might request change event records or endpoint inventory. Assuming that no error is encountered, a SWIMA-PC MUST send a SWIMA Response attribute in direct response to this SWIMA Request attribute, just as if the Subscribe flag was not set. As such, the attribute exchange that establishes a new subscription in a SWIMA-PC has the same flow as the flow seen in the previous attribute exchanges, as depicted in Figure 2. If the SWIMA-PV does not receive a PA-TNC Error attribute (as described in Sections 3.9 and 5.15) in response to its subscription request, the subscription has been successfully established on the SWIMA-PC. The SWIMA Request attribute that establishes a new subscription is referred to as the "establishing request" for that subscription.

When a subscription is established, it is assigned a Subscription ID value. The Subscription ID is equal to the value of the Request ID of the establishing request. (For more about Request IDs, see Section 5.5.)

A SWIMA-PC MUST have the ability to record and support at least 8 simultaneous subscriptions and SHOULD have the ability to support more than this. These subscriptions might all come from a single SWIMA-PV, might all be from different SWIMA-PVs (residing on the same or different NEA Servers), or might be a mix. In the case that a SWIMA-PC receives a subscription request but is unable to support an additional subscription, it MUST respond to the request with a PA-TNC Error attribute with error code SWIMA_SUBSCRIPTION_DENIED_ERROR.

A SWIMA-PV MUST have the ability to record and support at least 256 simultaneous subscriptions and SHOULD have the ability to support more than this. Any number of these subscriptions might be to the same SWIMA-PC, and any number of these subscriptions might be to different SWIMA-PCs. In the latter case, some of these SWIMA-PCs might share a single endpoint, while others might be on different endpoints.

3.8.2. Managing Subscriptions

The SWIMA-PC MUST record each accepted subscription along with the identity of the party to whom attributes are to be pushed. This identity includes two parts:

- An identifier for the PB-TNC session between the Posture Broker Server on a NEA Server and the Posture Broker Client on the endpoint. This identifier is called the "Connection ID"
- The Posture Validator Identifier for the SWIMA-PV that made the subscription request

The Posture Validator Identifier is provided in the field of the same name in the PB-PA message that encapsulates the subscription request attribute (Section 4.5 of [RFC5793]), and this information is passed along to NEA Posture Collectors (Section 3.3 of [RFC5792]). The Connection ID is a value local to a particular endpoint’s Posture Broker Client that identifies an ongoing session between a specific Posture Broker Client and a specific Posture Broker Server. Posture Broker Clients and Posture Broker Servers need to be capable of supporting multiple simultaneous sessions, so they already need a way to locally distinguish each ongoing session. (See Section 3.1 of [RFC5793].) A Posture Broker Client needs to assign each session at a given time its own Connection ID that lasts for the life of that session. Connection IDs only need to be unique among the Connection IDs of simultaneously occurring sessions on that endpoint. This Connection ID needs to be exposed to the SWIMA-PC, and the SWIMA-PC needs to be informed when the Connection ID is unbound due to the closure of that connection.

Likewise, SWIMA-PVs MUST record each accepted subscription for which they are the subscribing party, including the parameters of the establishing request, along with the associated Subscription ID and the identity of the SWIMA-PC that will be fulfilling the subscription. The SWIMA-PV needs to retain this information in order to correctly interpret pushed SWIMA Response attributes sent in fulfillment of the subscription. The identity of the SWIMA-PC is given in the Posture Collector Identifier [RFC5793] of the PB-PA message header in all messages from that SWIMA-PC. The SWIMA-PV has no need to record the associated connection ID of the subscription as the SWIMA-PV is only receiving, not sending, attributes once a subscription is established.

3.8.3. Terminating Subscriptions

Subscriptions MAY be terminated at any time by the subscribing SWIMA-PV by setting the Clear Subscriptions flag in a SWIMA Request. (See Section 5.6 for more on using this flag.) In the case that a SWIMA Request with the Clear Subscriptions flag set is received, the SWIMA-PC MUST only clear subscriptions that match both the NEA Server’s Connection ID and the SWIMA-PV’s Posture Validator Identifier for this SWIMA Request and MUST clear all such subscriptions.

This specification does not give the SWIMA-PV the ability to terminate subscriptions individually -- all subscriptions to the SWIMA-PV are cleared when the Clear Subscriptions flag is set.

This specification does not give the SWIMA-PC the ability to unilaterally terminate a subscription. However, if the SWIMA-PC experiences a fatal error while fulfilling a subscription, resulting in sending a PA-TNC Error attribute with error code SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR, then the subscription whose fulfillment led to the error MUST be treated as terminated by both the SWIMA-PC and the SWIMA-PV. Only the subscription experiencing the error is cancelled; other subscriptions are unaffected. See Section 3.9 for more on this error condition.

Finally, a subscription is terminated if the connection between the SWIMA-PC and SWIMA-PV is closed. This occurs when the Connection ID used in the messages between the SWIMA-PC and the SWIMA-PV becomes unbound. Loss of this Connection ID would prevent the SWIMA-PC from sending messages in fulfillment of this subscription. As such, loss of the Connection ID necessarily forces subscription termination between the affected parties.

3.8.4. Subscription Status

A SWIMA-PV can request that a SWIMA-PC report the list of active subscriptions for which the SWIMA-PV is the subscriber. A SWIMA-PV can use this capability to recover lost information about active subscriptions. A SWIMA-PV can also use this capability to verify that a SWIMA-PC has not forgotten any of its subscriptions. The latter is especially useful in cases where a SWIMA-PC does not send any attributes in fulfillment of a given subscription for a long period of time. The SWIMA-PV can check the list of active subscriptions on the SWIMA-PC and verify whether the inactivity is due to (1) a lack of reportable events or (2) the SWIMA-PC forgetting its obligations to fulfill a given subscription.

A SWIMA-PV requests a list of its subscriptions on a given SWIMA-PC by sending that SWIMA-PC a Subscription Status Request. The SWIMA-PC MUST then respond with a Subscription Status Response (or a PA-TNC Error if an error condition is experienced). The Subscription Status Response MUST contain one subscription record for each of the active subscriptions for which the SWIMA-PV is the subscribing party.

3.8.5. Fulfilling Subscriptions

As noted in Section 3.6, SWIMA-PCs are required to automatically detect changes to an endpoint’s Software Inventory Evidence Collection in near real time. For every active subscription, the SWIMA-PC MUST send an attribute to the subscribed SWIMA-PV whenever a change to relevant records is detected within the endpoint’s Software Inventory Evidence Collection. Such an attribute is said to be sent "in fulfillment of" the given subscription, and any such attribute MUST include that subscription’s Subscription ID. If the establishing request for that subscription was a targeted request, then only records that match the Software Identifiers provided in that establishing request are considered relevant. Otherwise (i.e., for non-targeted requests), any record is considered relevant for this purpose. Figure 3 shows a sample attribute exchange where a subscription is established and then attributes are sent from the SWIMA-PC in fulfillment of the established subscription.

```
            +-------------+                    +--------------+
            |  SWIMA-PC   |                    |   SWIMA-PV   |  Time
            +-------------+                    +--------------+   |
                  |                                   |           |
                  |<----------SWIMA Request-----------|           |
                  |                                   |           |
                  |-----------SWIMA Response--------->|           |
                  |                                   |           |
                  .                                   .           .
                  .                                   .           .
                  .                                   .           .
    <Change Event>|                                   |           |
                  |----------SWIMA Response---------->|           |
                  |                                   |           |
                  .                                   .           .
                  .                                   .           .
                  .                                   .           .
    <Change Event>|                                   |           |
                  |----------SWIMA Response---------->|           |
                  |                                   |           V
```

Figure 3: Subscription Establishment and Fulfillment

The contents of an attribute sent in fulfillment of a subscription depend on the parameters provided in the establishing request for that subscription. Specifically, the attribute sent in fulfillment of a subscription has the same attribute type as would a direct response to the establishing request. For example, if the establishing request stipulated a response that contained an event record list that included Software Inventory Evidence Records, all attributes sent in fulfillment of this subscription will also consist of event record lists with Software Inventory Evidence Records. As such, all SWIMA Responses displayed in the exchange depicted in Figure 3 are the same attribute type. A SWIMA Response generated in fulfillment of an active subscription MUST be a valid SWIMA Response attribute according to all the rules outlined in the preceding sections. In other words, an attribute constructed in fulfillment of a subscription will look the same as an attribute sent in direct response to an explicit request from a SWIMA-PV that had the same request parameters and that arrived immediately after the given change event. There are a few special rules that expand on this guideline, as discussed in Sections 3.8.5.1 through 3.8.5.5.

3.8.5.1. Subscriptions That Report Inventories

In the case that a SWIMA-PV subscribes to a SWIMA-PC and requests an inventory attribute whenever changes are detected (i.e., the EID in the establishing request is 0), then the SWIMA-PC MUST send the requested inventory whenever a relevant change is detected. (A "relevant change" is any change for non-targeted requests or a change to an indicated record in a targeted request.) Upon detection of a relevant change for an active subscription, the SWIMA-PC sends the appropriate inventory information as if it had just received the establishing request. Inventory attributes sent in fulfillment of this subscription will probably have a large amount of redundancy, as the same records are likely to be present in each of these SWIMA attributes. The role of an inventory subscription is not to report records just for the items that changed -- that is the role of a subscription that reports events (see Section 3.8.5.2). A SWIMA-PC MUST NOT exclude a record from an attribute sent in fulfillment of an inventory subscription simply because that record was not involved in the triggering event (although a record might be excluded for other reasons, such as if the subscription is targeted; see Section 3.8.5.3).

3.8.5.2. Subscriptions That Report Events

A SWIMA-PV indicates that it wishes to establish a subscription requesting event records by providing a non-zero EID in the SWIMA Request establishing the subscription (see Section 3.7.1). However, when the SWIMA-PC constructs an attribute in fulfillment of the subscription (other than the direct response to the establishing request), it MUST only include event records for the detected change(s) that precipitated this response attribute. In other words, it MUST NOT send a complete list of all changes starting with the establishing request’s EID, up through the latest change, every time a new event is detected. In effect, the EID in the establishing request is treated as being updated every time an attribute is sent in fulfillment of this subscription, such that a single event is not reported twice in fulfillment of a single subscription. As such, every SWIMA-PC MUST track the EID of the last event that triggered an attribute for the given subscription. When the next event (or set of events) is detected, the SWIMA-PC MUST only report events with EIDs after the last reported event. In the case that the EID Epoch of the SWIMA-PC changes, the SWIMA-PC MUST reset this EID tracker to zero (if the event log is completely purged) or to the new EID of the last reported retained event (if the event log is partially purged to create a "sliding window"). Doing this ensures that the SWIMA-PC continues to only send events that have not been previously reported.

Note that while a subscription is active, the subscribing SWIMA-PV MAY make other requests for event records that overlap with events that are reported in fulfillment of a subscription. Such requests are not affected by the presence of the subscription, nor is the subscription affected by such requests. In other words, a given request will get the same results back whether or not there was a subscription. Likewise, an attribute sent in fulfillment of a subscription will contain the same information whether or not other requests had been received from the SWIMA-PV.

A SWIMA-PV needs to pay attention to the EID Epoch in these attributes, as changes in the Epoch might create discontinuities in the SWIMA-PV’s understanding of the endpoint’s Software Inventory Evidence Collection state, as discussed in Section 3.7.6. In particular, once the EID Epoch changes, a SWIMA-PV is unable to have confidence that it has a correct understanding of the state of an endpoint’s Software Inventory Evidence Collection until after the SWIMA-PV collects a complete inventory.

SWIMA-PCs MAY send partial lists of event records in fulfillment of a subscription. (See Section 3.7.5 for more on partial lists of event records.) In the case that a SWIMA-PC sends a partial list of event records in fulfillment of a subscription, it MUST immediately send the next consecutive partial list and continue doing so until it has sent the equivalent of the complete list of event records. In other words, if the SWIMA-PC sends a partial list, it does not wait for another change event to send another SWIMA Response; rather, it continues sending SWIMA Responses until it has sent all event records that would have been included in a complete fulfillment of the subscription. Note that the direct response to the establishing request is not considered to be sent in fulfillment of a subscription. However, in this case the SWIMA-PC MUST treat the presence of unreported events as a triggering event for pushing additional messages in fulfillment of the newly established subscription. As such, the net effect is that if the direct response to the establishing request (i.e., the Subscription Fulfillment flag is unset) is partial, the SWIMA-PC will immediately follow this with additional attributes (with the Subscription Fulfillment flag set) until the complete set of events has been sent to the SWIMA-PV.

3.8.5.3. Targeted Subscriptions

Subscriptions MAY be targeted to only apply to records that match a given set of Software Identifiers. In the case where changes that affect multiple records are detected -- some matching the establishing request’s Software Identifiers and some not -- the attribute sent in fulfillment of the subscription MUST only include inventory or events (as appropriate) for records that match the establishing request’s Software Identifiers. The SWIMA-PC MUST NOT include non-matching records in the attribute, even if those non-matching records experienced change events that were simultaneous with change events on the matching records.

In addition, a SWIMA-PC MUST send an attribute in fulfillment of a targeted subscription only when changes to the endpoint’s Software Inventory Evidence Collection impact one or more records matching the subscription’s establishing request’s Software Identifiers. A SWIMA-PC MUST NOT send any attribute in fulfillment of a targeted subscription based on detected changes to the endpoint’s Software Inventory Evidence Collection that did not involve any of the records targeted by that subscription.

3.8.5.4. No Subscription Consolidation

A SWIMA-PV MAY establish multiple subscriptions to a given SWIMA-PC. If this is the case, it is possible that a single change event on the endpoint might require fulfillment by multiple subscriptions and that the information included in attributes that fulfill each of these subscriptions might overlap. The SWIMA-PC MUST send separate attributes for each established subscription that requires a response due to the given event. Each of these attributes MUST contain all information required to fulfill that individual subscription, even if that information is also sent in other attributes sent in fulfillment of other subscriptions at the same time. In other words, SWIMA-PCs MUST NOT attempt to combine information when fulfilling multiple subscriptions simultaneously, even if this results in some redundancy in the attributes sent to the SWIMA-PV.

3.8.5.5. Delayed Subscription Fulfillment

A SWIMA-PC MAY delay the fulfillment of a subscription following a change event in the interest of waiting to see if additional change events are forthcoming and, if so, conveying the relevant records back to the SWIMA-PV in a single SWIMA Response attribute. This can help reduce network bandwidth consumption between the SWIMA-PC and the SWIMA-PV. For example, consider a situation where 10 changes occur a tenth of a second apart. If the SWIMA-PC does not delay in assembling and sending SWIMA Response attributes, the SWIMA-PV will receive 10 separate SWIMA Response attributes over a period of 1 second. However, if the SWIMA-PC waits half a second after the initial event before assembling a SWIMA Response, the SWIMA-PV only receives two SWIMA Response attributes over the same period of time.

Note that the ability to consolidate events for a single subscription over a given period of time does not contradict the rules in Section 3.8.5.4 prohibiting consolidation across multiple subscriptions. When delaying fulfillment of subscriptions, SWIMA-PCs are still required to fulfill each individual subscription separately. Moreover, in the case that change events within the delay window cancel each other out (e.g., a record is deleted and then re-added), the SWIMA-PC MUST still report each change event, rather than just report the net effect of changes over the delay period. In other words, delayed fulfillment can decrease the number of attributes sent by the SWIMA-PC, but it does not reduce the total number of change events reported.

SWIMA-PCs are not required to support delayed fulfillment of subscriptions. However, in the case that the SWIMA-PC does support delayed subscription fulfillment, it MUST be possible to configure the SWIMA-PC to disable delayed fulfillment. In other words, parties deploying SWIMA-PCs need to be allowed to disable delayed subscription fulfillment in their SWIMA-PCs. The manner in which such configuration occurs is left to the discretion of implementers, although implementers MUST protect the configuration procedure from unauthorized tampering. In other words, there needs to be some assurance that unauthorized individuals are not able to introduce long delays in subscription fulfillment.

3.9. Error Handling

In the case where the SWIMA-PC detects an error in a SWIMA Request attribute that it receives, it MUST respond with a PA-TNC Error attribute with an error code appropriate to the nature of the error. (See Section 4.2.8 of PA-TNC [RFC5792] for more details about PA-TNC Error attributes and error codes, and see Section 5.15 in this specification for error codes specific to SWIMA attributes.) In the case that an error is detected in a SWIMA Request, the SWIMA-PC MUST NOT take any action requested by this SWIMA Request, even if partial completion of the request is possible. In other words, a SWIMA Request that contains an error will be completely ignored by the SWIMA-PC (beyond sending a PA-TNC Error attribute and possibly logging the error locally); no attempt at partial completion of the request will be made.

In the case where the SWIMA-PC receives a valid SWIMA Request attribute but experiences an error during the process of responding to that attribute’s instructions where that error prevents the SWIMA-PC from properly or completely fulfilling that request, the SWIMA-PC MUST send a PA-TNC Error attribute with an error code appropriate to the nature of the error. In the case where a PA-TNC Error attribute is sent, the SWIMA-PC MUST NOT take any of the actions requested by the SWIMA Request attribute that led to the detected error. This is the case even if some actions could have been completed successfully and might even require the SWIMA-PC to reverse some successful actions already taken before the error condition was detected. In other words, either (1) all aspects of a SWIMA Request complete fully and successfully (in which case the SWIMA-PC sends a SWIMA Response attribute) or (2) no aspects of the SWIMA Request occur (in which case the SWIMA-PC sends a PA-TNC Error attribute). In the case that a SWIMA-PC sends a PA-TNC Error attribute in response to a SWIMA Request, then the SWIMA-PC MUST NOT also send any SWIMA Response attribute in response to the same SWIMA Request. For this reason, the sending of a SWIMA Response attribute MUST be the last action taken by a SWIMA-PC in response to a SWIMA Request, to avoid the possibility of a processing error occurring after that SWIMA Response attribute is sent.

In the case that the SWIMA-PC detects an error that prevents it from properly or completely fulfilling its obligations under an active subscription, the SWIMA-PC MUST send a PA-TNC Error attribute with error code SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR to the SWIMA-PV that established this subscription. This type of PA-TNC Error attribute identifies the specific subscription that cannot be adequately honored due to the error condition and also identifies an error "subtype". The error subtype indicates the error code of the error condition the SWIMA-PC experienced that prevented it from honoring the given subscription. In the case that the error condition cannot be identified or does not align with any of the defined error codes, the SWIMA_ERROR error code SHOULD be used in the subtype. In the case that a SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR is sent, the associated subscription MUST be treated as cancelled by both the SWIMA-PC and the SWIMA-PV.

The SWIMA-PV MUST NOT send any PA-TNC Error attributes to SWIMA-PCs. In the case that a SWIMA-PV detects an error condition, it SHOULD log this error, but the SWIMA-PV does not inform any SWIMA-PCs of this event. Errors might include, but are not limited to, the detection of malformed SWIMA Response attributes sent from a given SWIMA-PC, as well as the detection of error conditions when the SWIMA-PV processes SWIMA Responses.

Both SWIMA-PCs and SWIMA-PVs SHOULD log errors so that administrators can trace the causes of errors. Log entries SHOULD include the code of the error, the time it was detected, and additional descriptive information to aid in understanding the nature and cause of the error. Logs are an important debugging tool, and implementers are strongly advised to include comprehensive logging capabilities in their products.

4. Protocol

The SWIMA protocol supports two different types of message exchanges for conveying software inventory information. These message exchanges are described in the following subsections, along with implementation requirements for supporting them.

The SWIMA protocol also supports two simple status exchanges: a Subscription Status exchange for conveying information about active subscriptions, and a Source Metadata exchange for conveying information about a SWIMA-PC’s data sources. In both cases, a SWIMA-PV sends a request attribute (Subscription Status Request or Source Metadata Request, respectively) and a SWIMA-PC responds with a matching response attribute (Subscription Status Response or Source Metadata Response, respectively). As these exchanges are straightforward, no additional information on the exchanges is provided.

4.1. Direct Response to a SWIMA Request

The first type of software information exchange is used to provide the SWIMA-PV with a software inventory or event collection from the queried endpoint.

```
       +-------------+                      +--------------+
       |  SWIMA-PC   |                      |   SWIMA-PV   |  Time
       +-------------+                      +--------------+   |
             |                                     |           |
             |<-----------SWIMA Request------------|           |
             |                                     |           |
             |           SWIMA Response*           |           |
             |-----------------or----------------->|           |
             |             PA-TNC Error            |           |
             |                                     |           V
```

*SWIMA Response is one of the following: Software Identifier Inventory, Software Identifier Events, Software Inventory, or Software Events.

Figure 4: SWIMA Attribute Exchange (Direct Response to SWIMA Request)

In this exchange, the SWIMA-PV indicates to the SWIMA-PC, via a SWIMA Request, the nature of the information it wishes to receive (inventory vs. events, full or targeted) and how it wishes the returned inventory to be expressed (with or without Software Inventory Evidence Records). The SWIMA-PC responds with the requested information using the appropriate attribute type. A single SWIMA Request MUST only lead to a single SWIMA Response or PA-TNC Error that is in direct response to that request.

4.2. Subscription-Based Response

The second type of software information exchange allows change-event-based reporting based on a subscription. If there is an active subscription on the endpoint, the SWIMA-PC sends a SWIMA Response to the SWIMA-PV following a change event on the endpoint in fulfillment of that subscription. Such an exchange is shown in Figure 5.

```
            +-------------+                +--------------+
            |  SWIMA-PC   |                |   SWIMA-PV   |  Time
            +-------------+                +--------------+   |
                  |                               |           |
    <Change Event>|                               |           |
                  |------SWIMA Response(s)*------>|           |
                  |                               |           |
                  |                               |           V
```

*SWIMA Response is one of the following: Software Identifier Inventory, Software Identifier Events, Software Inventory, or Software Events.

Figure 5: SWIMA Attribute Exchange (in Fulfillment of an Active Subscription)

Note that unlike direct responses to a SWIMA Request, a single change event can precipitate multiple SWIMA Responses for a single subscription, but only if all but the last of those SWIMA Responses convey partial lists of event records. When providing multiple SWIMA Responses in this way, the initial responses contain partial lists of event records and the last of those SWIMA Responses conveys the remainder of the relevant event records, completing the delivery of all relevant events in response to the change event. A single change event MUST NOT otherwise be followed by multiple SWIMA Responses or PA-TNC Error attributes in any combination.

4.3. Required Exchanges

All SWIMA-PVs and SWIMA-PCs MUST support both types of software information exchanges. In particular, SWIMA-PCs MUST be capable of pushing a SWIMA Response to a SWIMA-PV immediately upon detection of a change to the endpoint’s Software Inventory Evidence Collection in fulfillment of established SWIMA-PV subscriptions, as described in Section 3.8.

All SWIMA-PCs MUST support both status exchanges (Subscription Status and Source Metadata); SWIMA-PVs are recommended to support these status exchanges, but doing so is not required.

5. Software Inventory Messages and Attributes

This section describes the format and semantics of the SWIMA protocol. This protocol uses the PA-TNC message header format [RFC5792].

5.1. PA Subtype (aka PA-TNC Component Type)

The NEA PB-TNC [RFC5793] interface provides a general message-batching protocol capable of carrying one or more PA-TNC messages between the Posture Broker Client and Posture Broker Server. When PB-TNC is carrying a PA-TNC message, the PB-TNC message headers contain a 32-bit identifier called the "PA Subtype". The PA Subtype field indicates the type of component associated with all of the PA-TNC attributes carried by the PB-TNC message. The core set of PA Subtypes is defined in the PA-TNC specification. This specification defines a new "SWIMA Attributes" PA Subtype, which is registered in Section 10.2 of this document and is used as a namespace for the collection of SWIMA attributes defined in this document.

For more information on PB-TNC messages and PA-TNC messages, as well as their message headers, see the PB-TNC [RFC5793] and PA-TNC [RFC5792] specifications, respectively.

5.2. SWIMA Attribute Overview

Each PA-TNC attribute described in this specification is intended to be sent between the SWIMA-PC and SWIMA-PV and so will be carried in a PB-TNC message indicating a PA Subtype of "SWIMA Attributes". PB-TNC messages MUST always include the SWIMA Attributes Subtype defined in Section 5.1 when carrying SWIMA attributes over PA-TNC. The attributes defined in this specification appear below, along with a short summary of their purposes.

PA-TNC attribute types are identified in the PA-TNC Attribute Header via the PA-TNC Attribute Vendor ID field and the PA-TNC Attribute Type field; see Section 4.1 of [RFC5792] for details. Table 1 identifies the appropriate values for these fields for each attribute type used within the SWIMA protocol. All attributes have a PEN value of 0x000000. Both the hexadecimal and decimal values are provided in the Integer column in the table. Each attribute is described in greater detail in subsequent sections (identified in the table’s Description column).

Attribute Name	Integer	Description
SWIMA Request	0x0000000D (13)	Request sent from a SWIMA-PV to a SWIMA-PC for the SWIMA-PC to provide a software inventory or event list. It might also establish a subscription. (Section 5.6)
Software Identifier Inventory	0x0000000E (14)	An inventory sent without Software Inventory Evidence Records sent from a SWIMA-PC. (Section 5.7)
Software Identifier Events	0x0000000F (15)	A collection of events impacting the endpoint’s Software Inventory Evidence Collection, where events do not include Software Inventory Evidence Records. (Section 5.8)
Software Inventory	0x00000010 (16)	An inventory including Software Inventory Evidence Records sent from a SWIMA-PC. (Section 5.9)
Software Events	0x00000011 (17)	A collection of events impacting the endpoint’s Software Inventory Evidence Collection, where events include Software Inventory Evidence Records. (Section 5.10)
Subscription Status Request	0x00000012 (18)	A request for a list of a SWIMA-PV’s active subscriptions on a SWIMA-PC. (Section 5.11)
Subscription Status Response	0x00000013 (19)	A list of a SWIMA-PV’s active subscriptions on a SWIMA-PC. (Section 5.12)
Source Metadata Request	0x00000014 (20)	A request for information about a SWIMA-PC’s data sources. (Section 5.13)

Source Metadata Response	0x00000015 (21)	Descriptive metadata about a SWIMA-PC’s data sources. (Section 5.14)
PA-TNC Error	0x00000008 (8)	An error attribute as defined in the PA-TNC specification [RFC5792].

Table 1: SWIMA Attribute Enumeration

Because one of the Software Identifier Inventory, Software Identifier Events, Software Inventory, or Software Events attributes is expected to be sent to a SWIMA-PV in direct response to a SWIMA Request attribute or in fulfillment of an active subscription, those four attribute types are referred to collectively in this document as "SWIMA Response attributes".

All SWIMA-PVs MUST be capable of sending SWIMA Request attributes and be capable of receiving and processing all SWIMA Response attributes as well as PA-TNC Error attributes. All SWIMA-PCs MUST be capable of receiving and processing SWIMA Request attributes and be capable of sending all types of SWIMA Response attributes as well as PA-TNC Error attributes. SWIMA-PVs MUST ignore any SWIMA Request attributes that they receive. SWIMA-PCs MUST ignore any SWIMA Response attributes or PA-TNC Error attributes that they receive.

5.3. Message Diagram Syntax

This specification uses diagrams to define the syntax of new PA-TNC messages and attributes. Each diagram depicts the format and size of each field in bits. Implementations MUST send the bits depicted in each diagram as they are shown from left to right for each 32-bit quantity, "traversing" the diagram from top to bottom. Fields representing numeric values MUST be sent in network (big endian) byte order.

Descriptions of bit field (e.g., flag) values refer to the position of the bit within the field. These bit positions are numbered from the most significant bit through the least significant bit. As such, an octet with only bit 0 set would have a value of 0x80 (1000 0000), an octet with only bit 1 set would have a value of 0x40 (0100 0000), and an octet with only bit 7 set would have a value of 0x01 (0000 0001).

5.4. Normalization of Text Encoding

In order to ensure consistency of transmitted attributes, some fields require normalization of their format. When this is necessary, this information is indicated in the field’s description. In such cases, the field contents MUST be normalized to Network Unicode format as defined in RFC 5198 [RFC5198]. Network Unicode format defines a refinement of UTF-8 [RFC3629] that ensures a normalized expression of characters. SWIMA-PCs and SWIMA-PVs MUST NOT perform conversion and normalization on any field values except those specifically identified in the following sections as requiring normalization. Note, however, that some data models require additional normalization before source information is added to an endpoint’s Software Inventory Evidence Collection as a record. The references from the "Software Data Model Types" registry (see Section 10.5) will note where this is necessary.

5.5. Request IDs

All SWIMA Request attributes MUST include a Request ID value. The Request ID field provides a value that identifies a given request relative to other requests between a SWIMA-PV and the receiving SWIMA-PC. Specifically, the SWIMA-PV assigns each SWIMA Request attribute a Request ID value that is intended to be unique within the lifetime of a given network Connection ID.

In the case that a SWIMA Request requests the establishment of a subscription and the receiving SWIMA-PC agrees to that subscription, the Request ID of that SWIMA Request (i.e., the establishing request of the subscription) becomes that subscription’s Subscription ID. All attributes sent in fulfillment of this subscription include a flag indicating that the attribute fulfills a subscription and the subscription’s Subscription ID. A SWIMA-PV MUST NOT reuse a Request ID value in communications with a given SWIMA-PC while that Request ID is also serving as a Subscription ID for an active subscription with that SWIMA-PC. In the case where a SWIMA-PC receives a SWIMA Request from a given SWIMA-PV where that Request ID is also the Subscription ID of an active subscription with that SWIMA-PV, the SWIMA-PC MUST respond with a PA-TNC Error attribute with an error code of SWIMA_SUBSCRIPTION_ID_REUSE_ERROR. Note that this error does not cancel the indicated subscription.

Subscription Status Requests and Subscription Status Responses do not include Request IDs.

In the case where all possible Request ID values have been exhausted within the lifetime of a single network Connection ID, the sender MAY reuse previously used Request IDs within the same network connection if the ID is not being used as a Subscription ID. In the case where reuse is necessary due to exhaustion of possible ID values, the SWIMA-PV SHOULD structure the reuse to maximize the time between original and subsequent use. The Request ID value is included in a SWIMA Response attribute directly responding to this SWIMA Request to indicate which SWIMA Request was received and caused the response. Request IDs can be randomly generated or sequential, as long as values are not repeated per the rules in this paragraph. SWIMA-PCs are not required to check for duplicate Request IDs, except insofar as is necessary to detect Subscription ID reuse.

5.6. SWIMA Request

A SWIMA-PV sends this attribute to a SWIMA-PC to request that the SWIMA-PC send software inventory information to the SWIMA-PV. A SWIMA-PC MUST NOT send this attribute.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Flags        |       Software Identifier Count               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Request ID                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Earliest EID                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   |    SUB-BLOCK (Repeated "Software Identifier Count" times)     |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: SWIMA Request Attribute

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Software Identifier Length  | Software Identifier (var len) |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: SWIMA Request Attribute SUB-BLOCK

Field	Description
Flags: Bit 0 - Clear Subscriptions	If set (1), the SWIMA-PC MUST delete all subscriptions established by the requesting SWIMA-PV (barring any errors).
Flags: Bit 1 - Subscribe	If set (1), in addition to responding to the request as described, the SWIMA-PC MUST establish a subscription with parameters matching those in the SWIMA Request attribute (barring any errors).
Flags: Bit 2 - Result Type	If unset (0), the SWIMA-PC's response MUST include Software Inventory Evidence Records, and thus the response MUST be a Software Inventory, Software Events, or PA-TNC Error attribute. If set (1), the response MUST NOT include Software Inventory Evidence Records, and thus the response MUST be a Software Identifier Inventory, Software Identifier Events, or PA-TNC Error attribute.
Flags: Bits 3-7 - Reserved	Reserved for future use. This field MUST be set to zero on transmission and ignored upon reception.
Software Identifier Count	A 3-byte unsigned integer indicating the number of Software Identifiers that follow. If this value is non-zero, this is a targeted request, as described in Section 3.5. The Software Identifier Length and Software Identifier fields are repeated, in order, the number of times indicated in this field. In the case where Software Identifiers are present, the SWIMA-PC MUST only report software that corresponds to the identifiers the SWIMA-PV provided in this attribute (or respond with a PA-TNC Error attribute). This field value MAY be 0, in which case there are no instances of the Software Identifier Length and Software Identifier fields. In this case, the SWIMA-PV is indicating an interest in all Software Inventory Evidence Records on the endpoint (i.e., this is not a targeted request).
Request ID	A value that uniquely identifies this SWIMA Request from a particular SWIMA-PV.

Earliest EID	In the case where the SWIMA-PV is requesting software events, this field contains the EID value of the earliest event the SWIMA-PV wishes to have reported. (Note: The report will be inclusive of the event with this EID value.) In the case where the SWIMA-PV is requesting an inventory, then this field MUST be 0 (0x00000000). In the case where this field is non-zero, the SWIMA-PV is requesting events, and the SWIMA-PC MUST respond using a Software Events, Software Identifier Events, or PA-TNC Error attribute. In the case where this field is zero, the SWIMA-PV is requesting an inventory, and the SWIMA-PC MUST respond using a Software Inventory, Software Identifier Inventory, or PA-TNC Error attribute.
Software Identifier Length	A 2-byte unsigned integer indicating the length, in bytes, of the Software Identifier field.
Software Identifier	A string containing the Software Identifier value from a Software Inventory Evidence Record. This field value MUST be normalized to Network Unicode format, as described in Section 5.4. This string MUST NOT be null terminated.

Table 2: SWIMA Request Attribute Fields

The SWIMA-PV sends the SWIMA Request attribute to a SWIMA-PC to request the indicated information. Note that between the Result Type flag and the Earliest EID field, the SWIMA-PC is constrained to a single possible SWIMA Response attribute type (or a PA-TNC Error attribute) in its response to the request.

The Subscribe flag and the Clear Subscriptions flag are used to manage subscriptions for the requesting SWIMA-PV on the receiving SWIMA-PC. Specifically, an attribute with the Subscribe flag set seeks to establish a new subscription by the requesting SWIMA-PV to the given SWIMA-PC, while an attribute with the Clear Subscriptions flag set seeks to delete all existing subscriptions by the requesting SWIMA-PV on the given SWIMA-PC. Note that in the latter case, only the subscriptions associated with the Connection ID and the Posture Validator Identifier of the requester are deleted as described in Section 3.8.3. A newly established subscription has the parameters outlined in the SWIMA Request attribute. Specifically, the Result Type flag indicates the type of result to send in fulfillment of the subscription, the value of the Earliest EID field indicates whether the fulfillment attributes list inventories or events, and the fields describing Software Identifiers (if present) indicate if and how a subscription is targeted. In the case that the SWIMA-PC is unable or unwilling to comply with the SWIMA-PV’s request to establish or clear subscriptions, the SWIMA-PC MUST respond with a PA-TNC Error attribute with the SWIMA_SUBSCRIPTION_DENIED_ERROR error code. If the SWIMA-PV requests that subscriptions be cleared but has no existing subscriptions, this is not an error.

An attribute requesting the establishment of a subscription is effectively doing "double duty", as it is a request for an immediate response from the SWIMA-PC in addition to setting up the subscription. Assuming that the SWIMA-PC is willing to comply with the subscription, it MUST send an appropriate response attribute to a request with the Subscribe flag set containing all requested information. The same is true of the Clear Subscriptions flag -- assuming that there is no error, the SWIMA-PC MUST generate a response attribute without regard to the presence of this flag, in addition to clearing its subscription list.

Both the Subscribe flag and the Clear Subscriptions flag MAY be set in a single SWIMA Request attribute. In the case where this request is successful, the end result MUST be equivalent to the SWIMA-PC clearing its subscription list for the given SWIMA-PV first and then creating a new subscription in accordance with the request parameters. In other words, do not first create the new subscription and then clear all the subscriptions (including the one that was just created). In the case that the requested actions are successfully completed, the SWIMA-PC MUST respond with a SWIMA Response attribute. The specific type of SWIMA Response attribute depends on the Result Type flag and the Earliest EID field, as described above. In the case where there is a failure that prevents some part of this request from completing, the SWIMA-PC MUST NOT add a new subscription, MUST NOT clear the old subscriptions, and MUST respond with a PA-TNC Error attribute. In other words, the SWIMA-PC MUST NOT partially succeed at implementing such a request; either all actions succeed or none succeed.

The Earliest EID field is used to indicate if the SWIMA-PV is requesting an inventory or event list from the SWIMA-PC. A value of 0 (0x00000000) represents a request for inventory information. Otherwise, the SWIMA-PV is requesting event information. For Earliest EID values other than 0, the SWIMA-PC MUST respond with event records, as described in Section 3.7. Note that the request does not identify a particular EID Epoch, since responses can only include events in the SWIMA-PC’s current EID Epoch.

The Software Identifier Count indicates the number of Software Identifiers in the attribute. This number might be any value between 0 and 16,777,216, inclusive. A single Software Identifier is represented by the following fields: Software Identifier Length and Software Identifier. These fields are repeated a number of times equal to the Software Identifier Count, which may be 0. The Software Identifier Length field indicates the number of bytes allocated to the Software Identifier field. The Software Identifier field contains a Software Identifier as described in Section 3.4.1. The presence of one or more Software Identifiers is used by the SWIMA-PV to indicate a targeted request, which seeks only inventories of or events affecting software corresponding to the given identifiers. The SWIMA-PC MUST only report software that matched the Software Identifiers provided in the SWIMA-PV’s SWIMA Request attribute.

5.7. Software Identifier Inventory

A SWIMA-PC sends this attribute to a SWIMA-PV to convey the inventory of the endpoint’s Software Inventory Evidence Collection without the inclusion of Software Inventory Evidence Records. This list might represent a complete inventory or a targeted list of records, depending on the parameters in the SWIMA-PV’s request. A SWIMA-PV MUST NOT send this attribute. The SWIMA-PC sends this attribute either (1) in fulfillment of an existing subscription where the establishing request has a Result Type of 1 and the Earliest EID is zero or (2) in direct response to a SWIMA Request attribute where the Result Type is 1 and the Earliest EID is zero.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Flags        |         Software Identifier Count             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Request ID Copy / Subscription ID               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           EID Epoch                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Last EID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|    SUB-BLOCK (Repeated "Software Identifier Count" times)     |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: Software Identifier Inventory Attribute

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Record Identifier                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Data Model Type PEN                 |Data Model Type|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Source Id Num |  Reserved     |   Software Identifier Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Software Identifier (variable length)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Software Locator Length     |Software Locator (variable len)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: Software Identifier Inventory Attribute SUB-BLOCK

Field	Description
Flags: Bit 0 - Subscription Fulfillment	In the case that this attribute is sent in fulfillment of a subscription, this bit MUST be set (1). In the case that this attribute is a direct response to a SWIMA Request, this bit MUST be unset (0).
Flags: Bits 1-7 - Reserved	Reserved for future use. This field MUST be set to zero on transmission and ignored upon reception.
Software Identifier Count	The number of Software Identifiers that follow. This field is an unsigned integer. The Record Identifier, Data Model Type PEN, Data Model Type, Source Identification Number, Reserved, Software Identifier Length, Software Identifier, Software Locator Length, and Software Locator fields are repeated, in order, the number of times indicated in this field. This field value MAY be 0, in which case there are no instances of these fields.

Request ID Copy / Subscription ID	In the case where this attribute is in direct response to a SWIMA Request attribute from a SWIMA-PV, this field MUST contain an exact copy of the Request ID field from that SWIMA Request. In the case where this attribute is sent in fulfillment of an active subscription, this field MUST contain the Subscription ID of the subscription being fulfilled by this attribute.
EID Epoch	The EID Epoch of the Last EID value. This field is a 4-byte unsigned integer.
Last EID	The EID of the last event recorded by the SWIMA-PC, or 0 if the SWIMA-PC has no recorded events. This field is a 4-byte unsigned integer.
Record Identifier	A 4-byte unsigned integer containing the Record Identifier value from a Software Inventory Evidence Record.
Data Model Type PEN	A 3-byte unsigned integer containing the Private Enterprise Number (PEN) of the organization that assigned the meaning of the Data Model Type value.
Data Model Type	A 1-byte unsigned integer containing an identifier number that identifies the data model of the reported record.
Source Identification Number	The Source Identifier number associated with the source from which this software installation inventory instance was reported.
Reserved	Reserved for future use. This field MUST be set to zero on transmission and ignored upon reception.
Software Identifier Length	A 2-byte unsigned integer indicating the length, in bytes, of the Software Identifier field.
Software Identifier	A string containing the Software Identifier value from a Software Inventory Evidence Record. This field value MUST be normalized to Network Unicode format, as described in Section 5.4. This string MUST NOT be null terminated.

Software Locator Length	A 2-byte unsigned integer indicating the length, in bytes, of the Software Locator field.
Software Locator	A string containing the Software Locator value. This field value MUST first be normalized to Network Unicode format, as described in Section 5.4, and then encoded as a URI [RFC3986]. This string MUST NOT be null terminated.

Table 3: Software Identifier Inventory Attribute Fields

In the case that this attribute is sent in fulfillment of a subscription, the Subscription Fulfillment bit MUST be set (1). In the case that this attribute is sent in direct response to a SWIMA Request, the Subscription Fulfillment bit MUST be unset (0). Note that the SWIMA Response attribute sent in direct response to a SWIMA Request that establishes a subscription (i.e., a subscription’s establishing request) MUST be treated as a direct response to that SWIMA Request (and thus the Subscription Fulfillment bit is unset). SWIMA Response attributes are only treated as being in fulfillment of a subscription (i.e., Subscription Fulfillment bit set) if they are sent following a change event, as shown in Figure 3.

The Software Identifier Count field indicates the number of Software Identifiers present in this inventory. Each Software Identifier is represented by the following set of fields: Record Identifier, Data Model Type PEN, Data Model Type, Source Identification Number, Reserved, Software Identifier Length, Software Identifier, Software Locator Length, and Software Locator. These fields will appear once for each reported record.

When responding directly to a SWIMA Request attribute, the Request ID Copy / Subscription ID field MUST contain an exact copy of the Request ID field from that SWIMA Request. When this attribute is sent in fulfillment of an existing subscription on this SWIMA-PC, this field MUST contain the Subscription ID of the fulfilled subscription.

The EID Epoch field indicates the EID Epoch of the Last EID value. The Last EID field MUST contain the EID of the last recorded change event (see Section 3.7 for more about EIDs and recorded events) at the time this inventory was collected. In the case where there are no recorded change events at the time that this inventory was collected, the Last EID field MUST contain 0. These fields can be interpreted to indicate that the provided inventory reflects the state of the endpoint after all changes up to and including this last event have been accounted for.

The Data Model Type PEN and Data Model Type fields are used to identify the data model associated with the given software record. These fields are discussed more in Section 3.4.2.

The Source Identification Number field is used to identify the source that provided the given record, as described in Section 3.1.

5.8. Software Identifier Events

A SWIMA-PC sends this attribute to a SWIMA-PV to convey events where the affected records are reported without Software Inventory Evidence Records. A SWIMA-PV MUST NOT send this attribute. The SWIMA-PC sends this attribute either (1) in fulfillment of an existing subscription where the establishing request has a Result Type of 1 and the Earliest EID is non-zero or (2) in direct response to a SWIMA Request attribute where the Result Type is 1 and the Earliest EID is non-zero.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Flags        |                Event Count                    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Request ID Copy / Subscription ID                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       EID Epoch                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        Last EID                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |               Last Consulted EID                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   |            SUB-BLOCK (Repeated "Event Count" times)           |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: Software Identifier Events Attribute

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            EID                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-                                                             -+
|                                                               |
+-                                                             -+
|                         Timestamp                             |
+-                                                             -+
|                                                               |
+-                                                             -+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Record Identifier                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Data Model Type PEN                 |Data Model Type|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Source Id Num |  Action       |   Software Identifier Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Software Identifier (variable length)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Software Locator Length     |Software Locator (variable len)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Software Identifier Events Attribute SUB-BLOCK

Field	Description
Flags: Bit 0 - Subscription Fulfillment	In the case that this attribute is sent in fulfillment of a subscription, this bit MUST be set (1). In the case that this attribute is a direct response to a SWIMA Request, this bit MUST be unset (0).
Flags: Bits 1-7 - Reserved	Reserved for future use. This field MUST be set to zero on transmission and ignored upon reception.
Event Count	The number of events that are reported in this attribute. This field is a 3-byte unsigned integer. The EID, Timestamp, Record Identifier, Data Model Type PEN, Data Model Type, Source Identification Number, Action, Software Identifier Length, Software Identifier, Software Locator Length, and Software Locator fields are repeated, in order, the number of times indicated in this field. This field value MAY be 0, in which case there are no instances of these fields.
Request ID Copy / Subscription ID	In the case where this attribute is in direct response to a SWIMA Request attribute from a SWIMA-PV, this field MUST contain an exact copy of the Request ID field from that SWIMA Request. In the case where this attribute is sent in fulfillment of an active subscription, this field MUST contain the Subscription ID of the subscription being fulfilled by this attribute.
EID Epoch	The EID Epoch of the Last EID value. This field is a 4-byte unsigned integer.
Last EID	The EID of the last event recorded by the SWIMA-PC, or 0 if the SWIMA-PC has no recorded events. This field contains the EID of the SWIMA-PC’s last recorded change event (which might or might not be included as an event record in this attribute).

Last Consulted EID	The EID of the last event record that was consulted when generating the event record list included in this attribute. This is different from the Last EID field value if and only if this attribute is conveying a partial list of event records. See Section 3.7.5 for more on partial lists of event records.
EID	The EID of the event in this event record.
Timestamp	The timestamp associated with the event in this event record. This timestamp is the SWIMA-PC's best understanding of when the given event occurred. Note that this timestamp might be an estimate. The Timestamp date and time MUST be represented as an ASCII string that is expressed in Coordinated Universal Time (UTC) and is compliant with RFC 3339 [RFC3339], with the additional restrictions that the 'T' delimiter and the 'Z' suffix MUST be capitalized and fractional seconds (time-secfrac) MUST NOT be included. This field conforms to the date-time ABNF production from Section 5.6 of RFC 3339, with the above restrictions. Leap seconds are permitted, and SWIMA-PVs MUST support them. The Timestamp string MUST NOT be null terminated or padded in any way. The length of this field is always 20 octets.
Record Identifier	A 4-byte unsigned integer containing the Record Identifier value from a Software Inventory Evidence Record.
Data Model Type PEN	A 3-byte unsigned integer containing the PEN of the organization that assigned the meaning of the Data Model Type value.
Data Model Type	A 1-byte unsigned integer containing an identifier number that identifies the data model of the reported record.
Source Identification Number	The Source Identifier number associated with the source for the software installation inventory instance that this event record reported.

Action	The type of event that is recorded in this event record. Possible values are as follows: 1 = CREATION - the addition of a record to the endpoint’s Software Inventory Evidence Collection; 2 = DELETION - the removal of a record from the endpoint’s Software Inventory Evidence Collection; 3 = ALTERATION - an alteration that was made to a record within the endpoint’s Software Inventory Evidence Collection. All other values are reserved for future use and MUST NOT be used when sending attributes. In the case where a SWIMA-PV receives an event record that uses an action value other than the ones defined here, it MUST ignore that event record but SHOULD process other event records in this attribute as normal.
Software Identifier Length	A 2-byte unsigned integer indicating the length, in bytes, of the Software Identifier field.
Software Identifier	A string containing the Software Identifier value from a Software Inventory Evidence Record. This field value MUST first be normalized to Network Unicode format, as described in Section 5.4. This string MUST NOT be null terminated.
Software Locator Length	A 2-byte unsigned integer indicating the length, in bytes, of the Software Locator field.
Software Locator	A string containing the Software Locator value. This field value MUST first be normalized to Network Unicode format, as described in Section 5.4, and then encoded as a URI [RFC3986]. This string MUST NOT be null terminated.

Table 4: Software Identifier Events Attribute Fields

The first few fields in the Software Identifier Events attribute mirror those in the Software Identifier Inventory attribute. The primary difference is that instead of conveying an inventory the attribute conveys zero or more event records, consisting of the EID, Timestamp, Record Identifier, Data Model Type PEN, Data Model Type, Source Identification Number, Action, Software Identifier Length, Software Identifier, Software Locator Length, and Software Locator fields of the affected Software Inventory Evidence Record.

With regard to the Timestamp field, it is important to note that clock skew between the SWIMA-PC and SWIMA-PV as well as between different SWIMA-PCs within an enterprise might make correlation of Timestamp values difficult. This specification does not attempt to resolve clock skew issues, although other mechanisms (which are outside the scope of this specification) do exist to reduce the impact of clock skew and make the timestamp more useful for such correlation. Instead, SWIMA uses the Timestamp value primarily as a means to indicate the amount of time between two events on a single endpoint. For example, by taking the difference of the times for when a record was removed and then subsequently re-added, one can get an indication as to how long the system was without the given record (and thus without the associated software). Since this will involve comparison of Timestamp values all originating on the same system, clock skew between the SWIMA-PC and SWIMA-PV is not an issue. However, if the SWIMA-PC’s clock was adjusted between two recorded events, it is possible for such a calculation to lead to misunderstandings regarding the temporal distance between events. Users of this field need to be aware of the possibility for such occurrences. In the case where the Timestamp values of two events appear to contradict the EID ordering of those events (i.e., the later EID has an earlier timestamp), the recipient MUST treat the EID ordering as correct.

All events recorded in a Software Identifier Events attribute are required to be part of the same EID Epoch. Specifically, all such reported events MUST have an EID that is from the same EID Epoch and that is the same as the EID Epoch of the Last EID and Last Consulted EID values. The SWIMA-PC MUST NOT report events with EIDs from different EID Epochs.

The Last Consulted EID field contains the EID of the last event record considered for inclusion in this attribute. If this attribute contains a partial event set (as described in Section 3.7.5), this field value will be less than the Last EID value; if this attribute contains a complete event set, the Last EID and Last Consulted EID values are identical.

If multiple events are sent in a Software Identifier Events attribute, the order in which they appear within the attribute is not significant. The EIDs associated with them are used for ordering the indicated events appropriately. Also note that a single software record might be reported multiple times in an attribute, such as if multiple events involving the associated record were being reported.

5.9. Software Inventory

A SWIMA-PC sends this attribute to a SWIMA-PV to convey a list of inventory records. A SWIMA-PV MUST NOT send this attribute. The SWIMA-PC sends this attribute either (1) in fulfillment of an existing subscription where the establishing request has a Result Type of 0 and the Earliest EID is zero or (2) in direct response to a SWIMA Request attribute where the Result Type is 0 and the Earliest EID is zero.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Flags         |            Record Count                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Request ID Copy / Subscription ID              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           EID Epoch                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Last EID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|          SUB-BLOCK (Repeated "Record Count" times)            |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12: Software Inventory Attribute

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Record Identifier                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Data Model Type PEN                 |Data Model Type|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Source Id Num |  Reserved     |   Software Identifier Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Software Identifier (variable length)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Software Locator Length     |Software Locator (variable len)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Record Length                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Record (variable length)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: Software Inventory Attribute SUB-BLOCK

Field	Description
Flags: Bit 0 - Subscription Fulfillment	In the case that this attribute is sent in fulfillment of a subscription, this bit MUST be set (1). In the case that this attribute is a direct response to a SWIMA Request, this bit MUST be unset (0).
Flags: Bits 1-7 - Reserved	Reserved for future use. This field MUST be set to zero on transmission and ignored upon reception.
Record Count	The number of records that follow. This field is a 3-byte unsigned integer. The Record Identifier, Data Model Type PEN, Data Model Type, Source Identification Number, Reserved, Software Identifier Length, Software Identifier, Software Locator Length, Software Locator, Record Length, and Record fields are repeated, in order, the number of times indicated in this field. This field value MAY be 0, in which case there are no instances of these fields.
Request ID Copy / Subscription ID	In the case where this attribute is in direct response to a SWIMA Request attribute from a SWIMA-PV, this field MUST contain an exact copy of the Request ID field from that SWIMA Request. In the case where this attribute is sent in fulfillment of an active subscription, this field MUST contain the Subscription ID of the subscription being fulfilled by this attribute.
EID Epoch	The EID Epoch of the Last EID value. This field is a 4-byte unsigned integer.
Last EID	The EID of the last event recorded by the SWIMA-PC, or 0 if the SWIMA-PC has no recorded events. This field is a 4-byte unsigned integer.
Record Identifier	A 4-byte unsigned integer containing the Record Identifier value from a Software Inventory Evidence Record.

Data Model Type PEN	A 3-byte unsigned integer containing the PEN of the organization that assigned the meaning of the Data Model Type value.
Data Model Type	A 1-byte unsigned integer containing an identifier number that identifies the data model of the reported record.
Source Identification Number	The Source Identifier number associated with the source from which this software installation inventory instance was reported.
Reserved	Reserved for future use. This field MUST be set to zero on transmission and ignored upon reception.
Software Identifier Length	A 2-byte unsigned integer indicating the length, in bytes, of the Software Identifier field.
Software Identifier	A string containing the Software Identifier value from a Software Inventory Evidence Record. This field value MUST first be normalized to Network Unicode format, as described in Section 5.4. This string MUST NOT be null terminated.
Software Locator Length	A 2-byte unsigned integer indicating the length, in bytes, of the Software Locator field.
Software Locator	A string containing the Software Locator value. This field value MUST first be normalized to Network Unicode format, as described in Section 5.4, and then encoded as a URI [RFC3986]. This string MUST NOT be null terminated.
Record Length	A 4-byte unsigned integer indicating the length, in bytes, of the Record field.
Record	A Software Inventory Evidence Record expressed as a string. The record MUST be converted and normalized to Network Unicode format, as described in Section 5.4. This string MUST NOT be null terminated.

Table 5: Software Inventory Attribute Fields

The Software Inventory attribute contains some number of Software Inventory Evidence Records along with the core response attribute fields. Given that the size of records can vary considerably, the length of this attribute is highly variable and, if transmitting a complete inventory, can be extremely large. To avoid unnecessarily overburdening the network, enterprises might wish to constrain the use of Software Inventory attributes to targeted requests.

When copying a Software Inventory Evidence Record into the Record field, the record MUST be converted and normalized to use Network Unicode format prior to its inclusion in the Record field.

5.10. Software Events

A SWIMA-PC sends this attribute to a SWIMA-PV to convey a list of events that include Software Inventory Evidence Records. A SWIMA-PV MUST NOT send this attribute. The SWIMA-PC sends this attribute either (1) in fulfillment of an existing subscription where the establishing request has a Result Type of 0 and the Earliest EID is non-zero or (2) in direct response to a SWIMA Request attribute where the Result Type is 0 and the Earliest EID is non-zero.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Flags       |                  Event Count                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |               Request ID Copy / Subscription ID               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           EID Epoch                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           Last EID                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Last Consulted EID                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   |             SUB-BLOCK (Repeated "Event Count" times)          |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 14: Software Events Attribute

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                             EID                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-                                                             -+
   |                                                               |
   +-                                                             -+
   |                         Timestamp                             |
   +-                                                             -+
   |                                                               |
   +-                                                             -+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Record Identifier                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           Data Model Type PEN                 |Data Model Type|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Source Id Num |  Action       |   Software Identifier Length  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |              Software Identifier (variable length)            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Software Locator Length     |Software Locator (variable len)|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        Record Length                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                  Record (variable length)                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 15: Software Events Attribute SUB-BLOCK

Field	Description
Flags: Bit 0 - Subscription Fulfillment	In the case that this attribute is sent in fulfillment of a subscription, this bit MUST be set (1). In the case that this attribute is a direct response to a SWIMA Request, this bit MUST be unset (0).
Flags: Bits 1-7 - Reserved	Reserved for future use. This field MUST be set to zero on transmission and ignored upon reception.
Event Count	The number of events being reported in this attribute. This field is a 3-byte unsigned integer. The EID, Timestamp, Record Identifier, Data Model Type PEN, Data Model Type, Source Identification Number, Action, Software Identifier Length, Software Identifier, Software Locator Length, Software Locator, Record Length, and Record fields are repeated, in order, the number of times indicated in this field. This field value MAY be 0, in which case there are no instances of these fields.
Request ID Copy / Subscription ID	In the case where this attribute is in direct response to a SWIMA Request attribute from a SWIMA-PV, this field MUST contain an exact copy of the Request ID field from that SWIMA Request. In the case where this attribute is sent in fulfillment of an active subscription, this field MUST contain the Subscription ID of the subscription being fulfilled by this attribute.
EID Epoch	The EID Epoch of the Last EID value. This field is a 4-byte unsigned integer.
Last EID	The EID of the last event recorded by the SWIMA-PC, or 0 if the SWIMA-PC has no recorded events. This field contains the EID of the SWIMA-PC’s last recorded change event (which might or might not be included as an event record in this attribute).

Last Consulted EID	The EID of the last event record that was consulted when generating the event record list included in this attribute. This is different from the Last EID field value if and only if this attribute is conveying a partial list of event records. See Section 3.7.5 for more on partial lists of event records.
EID	The EID of the event in this event record.
Timestamp	The timestamp associated with the event in this event record. This timestamp is the SWIMA-PC’s best understanding of when the given event occurred. Note that this timestamp might be an estimate. The Timestamp date and time MUST be represented as an ASCII string that is expressed in Coordinated Universal Time (UTC) and is compliant with RFC 3339 [RFC3339], with the additional restrictions that the ’T’ delimiter and the ’Z’ suffix MUST be capitalized and fractional seconds (time-secfrac) MUST NOT be included. This field conforms to the date-time ABNF production from Section 5.6 of RFC 3339, with the above restrictions. Leap seconds are permitted, and SWIMA-PVs MUST support them. The Timestamp string MUST NOT be null terminated or padded in any way. The length of this field is always 20 octets.
Record Identifier	A 4-byte unsigned integer containing the Record Identifier value from a Software Inventory Evidence Record.
Data Model Type PEN	A 3-byte unsigned integer containing the PEN of the organization that assigned the meaning of the Data Model Type value.
Data Model Type	A 1-byte unsigned integer containing an identifier number that identifies the data model of the reported record.
Source Identification Number	The Source Identifier number associated with the source for the software installation inventory instance that this event record reported.

Action	The type of event that is recorded in this event record. Possible values are as follows: 1 = CREATION - the addition of a record to the endpoint’s Software Inventory Evidence Collection; 2 = DELETION - the removal of a record from the endpoint’s Software Inventory Evidence Collection; 3 = ALTERATION - an alteration that was made to a record within the endpoint’s Software Inventory Evidence Collection. All other values are reserved for future use and MUST NOT be used when sending attributes. In the case where a SWIMA-PV receives an event record that uses an action value other than the ones defined here, it MUST ignore that event record but SHOULD process other event records in this attribute as normal.
Software Identifier Length	A 2-byte unsigned integer indicating the length, in bytes, of the Software Identifier field.
Software Identifier	A string containing the Software Identifier value from a Software Inventory Evidence Record. This field value MUST first be normalized to Network Unicode format, as described in Section 5.4. This string MUST NOT be null terminated.
Software Locator Length	A 2-byte unsigned integer indicating the length, in bytes, of the Software Locator field.
Software Locator	A string containing the Software Locator value. This field value MUST first be normalized to Network Unicode format, as described in Section 5.4, and then encoded as a URI [RFC3986]. This string MUST NOT be null terminated.

Record Length	A 4-byte unsigned integer indicating the length, in bytes, of the Record field.
Record	A Software Inventory Evidence Record expressed as a string. The record MUST be converted and normalized to Network Unicode format, as described in Section 5.4. This string MUST NOT be null terminated.

Table 6: Software Events Attribute Fields

The fields of this attribute are used in the same way as the corresponding fields of the previous attributes. As with the Software Inventory attribute, a Software Events attribute can be quite large if many events have occurred following the event indicated by a request’s Earliest EID. As such, it is recommended that the SWIMA Request attributes only request that full records be sent (Result Type set to zero) in a targeted request, thus constraining the response just to records that match a given set of Software Identifiers.

As with the Software Identifier Events attribute, this attribute MUST only contain event records with EIDs coming from the current EID Epoch of the SWIMA-PC.

As with the Software Inventory attribute, the SWIMA-PC MUST perform conversion and normalization of the record.

5.11. Subscription Status Request

A SWIMA-PV sends this attribute to a SWIMA-PC to request a list of active subscriptions for which the requesting SWIMA-PV is the subscriber. A SWIMA-PC MUST NOT send this attribute.

This attribute has no fields.

A SWIMA-PC MUST respond to this attribute by sending a Subscription Status Response attribute (or a PA-TNC Error attribute if it is unable to correctly provide a response).

5.12. Subscription Status Response

A SWIMA-PC sends this attribute to a SWIMA-PV to report the list of active subscriptions for which the receiving SWIMA-PV is the subscriber. A SWIMA-PV MUST NOT send this attribute.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Status Flags  |            Subscription Record Count          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|    SUB-BLOCK (Repeated "Subscription Record Count" times)     |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: Subscription Status Response Attribute

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Flags         |          Software Identifier Count            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Request ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Earliest EID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|  SUB-SUB-BLOCK (Repeated "Software Identifier Count" times)   |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 17: Subscription Status Response Attribute SUB-BLOCK

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Software Identifier Length   | Software Identifier (var len) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 18: Subscription Status Response Attribute SUB-SUB-BLOCK

Field	Description
Status Flags: Bits 0-7 - Reserved	Reserved for future use. This field MUST be set to zero on transmission and ignored upon reception.
Subscription Record Count	The number of subscription records that follow. This field is a 3-byte unsigned integer. The Flags, Software Identifier Count, Request ID, and Earliest EID fields, and zero or more instances of Software Identifier Length and Software Identifier, are repeated, in order, the number of times indicated in this field. (The Software Identifier Length and Software Identifier fields within each of these sets of fields are repeated a number of times equal to the preceding Software Identifier Count value.) The Subscription Record Count field value MAY be 0, in which case there are no instances of these fields.
Flags, Software Identifier Count, Request ID, Earliest EID, Software Identifier Length, and Software Identifier	For each active subscription, these fields contain an exact copy of the fields with the corresponding name provided in the subscription's establishing request.

Table 7: Subscription Status Response Fields

A Subscription Status Response contains zero or more subscription records. Specifically, it MUST contain one subscription record for each active subscription associated with the party that sent the Subscription Status Request to which this attribute is a response. As described in Section 3.8.2, the SWIMA-PC MUST use the requester's Connection ID and its Posture Validator Identifier to determine which subscriptions are associated with the requester.

A SWIMA-PC MUST send a Subscription Status Response attribute in response to a Subscription Status Request attribute, except in cases where the SWIMA-PC experiences an error condition that prevents it from correctly populating the Subscription Status Response attribute (in which case it MUST respond with a PA-TNC Error attribute appropriate to the type of error experienced). If there are no active subscriptions associated with the requesting party, the Subscription Status Response attribute will consist only of its Status Flags field and a Subscription Record Count field with a value of 0, and no additional fields.

Each subscription record included in a Subscription Status Response attribute duplicates the fields of the SWIMA Request attribute that was the establishing request of a subscription. Note that the Request ID field in the record captures the Subscription ID associated with the given subscription record (since the Subscription ID is the same as the Request ID of the establishing request). Note also that if the establishing request is targeted, then its Record Count field will be non-zero and, within that subscription record, the Software Identifier Length and Software Identifier fields are repeated, in order, the number of times indicated in the Record Count field. As such, each subscription record can be different sizes. If the establishing request is not targeted (Record Count field is 0), the subscription record has no Software Identifier Length or Software Identifier fields.

When a SWIMA-PV compares the information received in a Subscription Status Response to its own records of active subscriptions, it should be aware that the SWIMA-PC might be unable to distinguish this SWIMA-PV from other SWIMA-PVs on the same NEA Server. As a result, it is possible that the SWIMA-PC will report more subscription records than the SWIMA-PV recognizes. For this reason, SWIMA-PVs SHOULD NOT automatically assume that extra subscriptions reported in a Subscription Status Response indicate a problem.

5.13. Source Metadata Request

A SWIMA-PV sends this attribute to a SWIMA-PC to request metadata about sources that the SWIMA-PC is using to collect software inventory information. A SWIMA-PC MUST NOT send this attribute.

This attribute has no fields.

A SWIMA-PC MUST respond to this attribute by sending a Source Metadata Response attribute (or a PA-TNC Error attribute if it is unable to correctly provide a response).

5.14. Source Metadata Response

A SWIMA-PC sends this attribute to a SWIMA-PV to provide descriptive metadata about the sources of software inventory information used by the SWIMA-PC. A SWIMA-PV MUST NOT send this attribute.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           Reserved            | Source Count  |               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               |
   |                                                               |
   |           SUB-BLOCK (Repeated "Source Count" times)           |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 19: Source Metadata Response Attribute

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Source Id Num |       Metadata Length         | Metadata (var)|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 20: Source Metadata Response Attribute SUB-BLOCK

Field	Description
Reserved	Reserved for future use. This field MUST be set to zero on transmission and ignored upon reception.
Source Count	The number of source records that follow. The Source Identification Number, Metadata Length, and Metadata fields are repeated, in order, the number of times indicated by this field. This field MAY be 0, in which case no fields follow (but this would only be done to indicate that the SWIMA-PC has no active sources; this would not be a typical situation).
Source Identification Number	The Source Identifier number associated with the described source for any communications with the recipient SWIMA-PV.
Metadata Length	A 2-byte unsigned integer indicating the length, in bytes, of the Metadata field.
Metadata	A string containing descriptive metadata about the indicated data source. This string MUST NOT be null terminated.

Table 8: Source Metadata Response Fields

A Source Metadata Response attribute contains zero or more records, each describing one of the data sources the SWIMA-PC uses to collect software inventory information. It SHOULD contain one metadata record for each source that the SWIMA-PC uses. (There might be reasons not to inform certain SWIMA-PVs of the presence of certain data sources.) The attribute MUST contain a metadata record for each source that has been identified in inventory or event messages to the given SWIMA-PV.

A SWIMA-PC MUST send a Source Metadata Response attribute in response to a Source Metadata Request attribute, except in cases where the SWIMA-PC experiences an error condition that prevents it from correctly populating the Source Metadata Response attribute (in which case it MUST respond with a PA-TNC Error attribute appropriate to the type of error experienced).

The Source Count field indicates how many source metadata records are included in the attribute. Each included record consists of a Source Identification Number field, a Metadata Length field, and a Metadata field.

The Source Identification Number field in the Source Metadata Response attribute corresponds to the Source Identification Number field in inventory and event messages. In the case where (1) the Source Identification Number value in this attribute matches a Source Identification Number field in an inventory or event record and (2) both the Source Metadata Response and the inventory or event record were sent to the same SWIMA-PV, the source described in the Metadata field MUST be the same source that provided the inventory or event record associated with this Source Identifier. Recall that a SWIMA-PC MAY use different Source Identification Number associations with different SWIMA-PVs. As such, the association between a Source Identification Number and the conveyed metadata is also only meaningful for communications between the sending SWIMA-PC and receiving SWIMA-PV. When sending to a given SWIMA-PV, the SWIMA-PC MUST use the recipient SWIMA-PV’s Source Identification Number associations.

The Metadata Length field indicates the length, in bytes, of the Metadata field. The Metadata field contains information about the indicated data source. This specification does not dictate a format for the contents of the Metadata field. This field MAY include machine-readable information. For broadest utility, the Metadata field SHOULD include human-readable, descriptive information about the data source.

5.15. PA-TNC Error as Used by SWIMA

The PA-TNC Error attribute is defined in the PA-TNC specification [RFC5792], and its use here conforms to that specification. A PA-TNC Error can be sent due to any error in the PA-TNC exchange and might also be sent in response to error conditions specific to the SWIMA exchange. The latter case utilizes error codes defined below.

A PA-TNC Error MUST be sent by a SWIMA-PC in response to a SWIMA Request in the case where the SWIMA-PC encounters a fatal error (i.e., an error that prevents further processing of an exchange) relating to the attribute exchange. A SWIMA-PV MUST NOT send this attribute. In the case where the SWIMA-PV experiences a fatal error, it MUST handle the error without sending a PA-TNC Error attribute. The SWIMA-PV MAY take other actions in response to the error, such as logging the cause of the error or even taking actions to isolate the endpoint.

A PA-TNC Error attribute is sent instead of a SWIMA Response attribute when certain issues prevent the reliable creation of a SWIMA Response. As such, a SWIMA-PC MUST NOT send both a PA-TNC Error attribute and a SWIMA Response attribute in response to a single SWIMA Request attribute.

Table 9 lists the error code values for the PA-TNC Error attribute that are specific to the SWIMA exchange. Error codes are shown in both hexadecimal and decimal format. In all of these cases, the Error Code Vendor ID field MUST be set to 0x000000, corresponding to the IETF SMI PEN. The error information structures for each error code are described in the following subsections.

Note that a message with a SWIMA attribute might also result in an error condition covered by the IETF Standard PA-TNC Error Codes defined in Section 4.2.8 of [RFC5792]. For example, a SWIMA attribute might have an invalid parameter, leading to an error code of "Invalid Parameter". In this case, the SWIMA-PC MUST use the appropriate PA-TNC Error Code value as defined in Section 4.2.8 of [RFC5792].

Error Code Value	Description
0x00000004 (4)	SWIMA_ERROR. This indicates a fatal error (i.e., an error that precludes the creation of a suitable response attribute) other than the errors described below but still specific to the processing of SWIMA attributes. The Description field SHOULD contain additional diagnostic information.
0x00000005 (5)	SWIMA_SUBSCRIPTION_DENIED_ERROR. This indicates that the SWIMA-PC denied the SWIMA-PV's request to establish a subscription. The Description field SHOULD contain additional diagnostic information.
0x00000006 (6)	SWIMA_RESPONSE_TOO_LARGE_ERROR. This indicates that the SWIMA-PC's response to the SWIMA-PV's request was too large to be serviced. The error information structure indicates the largest possible size of a response supported by the SWIMA-PC (see Section 5.15.2). The Description field SHOULD contain additional diagnostic information.
0x00000007 (7)	SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR. This indicates that the SWIMA-PC experienced an error while fulfilling a given subscription. The error information includes the Subscription ID of the relevant subscription, as well as a sub-error that describes the nature of the error the SWIMA-PC experienced. The SWIMA-PC and SWIMA-PV MUST treat the identified subscription as cancelled.
0x00000008 (8)	SWIMA_SUBSCRIPTION_ID_REUSE_ERROR. This indicates that the SWIMA-PC received a SWIMA Request from a given SWIMA-PV where the Request ID of that SWIMA Request is currently used as the Subscription ID of an active subscription with that SWIMA-PV. This error does not cancel the identified subscription.

Table 9: PA-TNC Error Codes for SWIMA

The following subsections describe the structures present in the error information fields. Note that all error structures include a variable-length field but do not include any fields indicating the length of those fields. A length field is unnecessary because all other fields in the PA-TNC Error attribute are of fixed length, and thus the length of the variable-length field can be found by subtracting the size of these fixed-length fields from the PA-TNC Attribute Length field in the PA-TNC Attribute Header.

5.15.1. SWIMA_ERROR, SWIMA_SUBSCRIPTION_DENIED_ERROR, and SWIMA_SUBSCRIPTION_ID_REUSE_ERROR Information

The SWIMA_ERROR error code indicates that the sender (the SWIMA-PC) has encountered an error that is related to the processing of a SWIMA Request attribute but that is not covered by SWIMA error codes that are more specific. The SWIMA_SUBSCRIPTION_DENIED_ERROR is used when the SWIMA-PV sends a request to establish a subscription or clear all subscriptions from the given SWIMA-PV but the SWIMA-PC is unable or unwilling to comply with this request. The SWIMA_SUBSCRIPTION_ID_REUSE_ERROR is used when the SWIMA-PC receives a SWIMA Request whose Request ID duplicates a Subscription ID of an active subscription with the request’s sender. All of these error codes use the following error information structure.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Copy of Request ID / Subscription ID               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Description (variable length)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: SWIMA_ERROR, SWIMA_SUBSCRIPTION_DENIED_ERROR, and SWIMA_SUBSCRIPTION_ID_REUSE_ERROR Information

Field	Description
Copy of Request ID / Subscription ID	In the case that this error condition is generated in direct response to a SWIMA Request attribute, this field MUST contain an exact copy of the Request ID field in the SWIMA Request attribute that caused this error. In the case that the attribute in question is generated in fulfillment of an active subscription, this field MUST contain the Subscription ID of the subscription for which the attribute was generated. (This is only possible if the error code is SWIMA_ERROR, as the other errors are not generated by subscription fulfillment.) Note that in the case of failed subscription fulfillment, the indicated error appears as a sub-error for a SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR, as described in Section 5.15.3.
Description	A UTF-8 [RFC3629] string describing the condition that caused this error. This field MAY be zero-length. However, senders SHOULD include some kind of description in all PA-TNC Error attributes with these error codes. This field MUST NOT be null terminated.

Table 10: SWIMA_ERROR, SWIMA_SUBSCRIPTION_DENIED_ERROR, and SWIMA_SUBSCRIPTION_ID_REUSE_ERROR Information Fields

This error information structure is used with SWIMA_ERROR, SWIMA_SUBSCRIPTION_DENIED_ERROR, and SWIMA_SUBSCRIPTION_ID_REUSE_ERROR status codes to identify the SWIMA Request attribute that precipitated the error condition and to describe the error. The Description field contains text describing the error. The SWIMA-PC MAY encode machine-interpretable information in this field but SHOULD also include a human-readable description of the error, since the receiving SWIMA-PV might not recognize the SWIMA-PC’s encoded information.

5.15.2. SWIMA_RESPONSE_TOO_LARGE_ERROR Information

The SWIMA_RESPONSE_TOO_LARGE_ERROR error code indicates that a SWIMA-PC’s response to a SWIMA-PV’s SWIMA Request attribute was too large to send.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Copy of Request ID / Subscription ID             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Maximum Allowed Size                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Description (variable length)                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 22: SWIMA_RESPONSE_TOO_LARGE_ERROR Information

Field	Description
Copy of Request ID / Subscription ID	In the case that the attribute in question is generated in direct response to a SWIMA Request, this field MUST contain an exact copy of the Request ID field in the SWIMA Request attribute that caused this error. In the case that the attribute in question is generated in fulfillment of an active subscription, this field MUST contain the Subscription ID of the subscription for which the attribute was generated. Note that in the latter case, the SWIMA_RESPONSE_TOO_LARGE_ERROR appears as a sub-error for a SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR, as described in Section 5.15.3.
Maximum Allowed Size	This field MUST contain an unsigned integer indicating the largest permissible size, in bytes, of the SWIMA attribute that the SWIMA-PC is currently willing to send in response to a SWIMA Request attribute.
Description	A UTF-8 [RFC3629] string describing the condition that caused this error. This field MAY be zero-length. However, senders SHOULD include some kind of description in all PA-TNC Error attributes with this error code. This field MUST NOT be null terminated.

Table 11: SWIMA_RESPONSE_TOO_LARGE_ERROR Information Fields

This error structure is used with the SWIMA_RESPONSE_TOO_LARGE_ERROR status code to identify the SWIMA Request attribute that precipitated the error condition and to describe the error. The Maximum Allowed Size field indicates the largest attribute the SWIMA-PC is willing to send in response to a SWIMA Request under the current circumstances. Note that under other circumstances, the SWIMA-PC might be willing to return larger or smaller responses than indicated (such as if the endpoint connects to the NEA Server using a different network protocol). The other fields in this error information structure have the same meanings as corresponding fields in the SWIMA_ERROR and SWIMA_SUBSCRIPTION_DENIED_ERROR information structures.

5.15.3. SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR Information

The SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR error code indicates that the SWIMA-PC encountered an error while fulfilling a subscription. The bytes after the first 4 octets duplicate a PA-TNC Error attribute (as described in Section 4.2.8 of PA-TNC [RFC5792]) that is used to identify the nature of the encountered error.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Subscription ID                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Reserved   |           Sub-error Code Vendor ID            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Sub-error Code                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Sub-error Information (variable length)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 23: SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR Information

Field	Description
Subscription ID	This field MUST contain the Subscription ID of the subscription whose fulfillment caused this error.
Reserved	This field MUST contain the value of the Reserved field of a PA-TNC Error attribute that describes the error condition encountered during subscription processing.
Sub-error Code Vendor ID	This field MUST contain the value of the Error Code Vendor ID field of a PA-TNC Error attribute that describes the error condition encountered during subscription processing.
Sub-error Code	This field MUST contain the value of the Error Code field of a PA-TNC Error attribute that describes the error condition encountered during subscription processing.
Sub-error Information	This field MUST contain the value of the Error Information field of a PA-TNC Error attribute that describes the error condition encountered during subscription processing.

Table 12: SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR Information Fields

This error structure is used with the SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR status code. The first 4 octets of this error structure contain the Subscription ID of the subscription that was being fulfilled when the error occurred. The remaining fields of this error structure duplicate the fields of a PA-TNC Error attribute, referred to as the "sub-error". The error code of the sub-error corresponds to the code of the error that the SWIMA-PC encountered while fulfilling the given subscription. The sub-error MUST NOT have an error code of SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR.

The SWIMA-PC sending a PA-TNC Error attribute with this error code, and the SWIMA-PV receiving it, MUST treat the subscription identified by the Subscription ID field as cancelled. All other subscriptions are unaffected.

6. Supported Data Models

SWIMA supports an extensible list of data models for representing and transmitting software inventory information. This list of data models appears in the "Software Data Model Types" registry (see Section 10.5). This document provides guidance for an initial set of data models. Other documents might provide guidance on the use of new data models by SWIMA and will be referenced by extensions to the "Software Data Model Types" registry.

6.1. ISO 2015 SWID Tags Using XML

The International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC) published the specification governing SWID tag construction and use (ISO/IEC 19770-2:2009) in 2009 [SWID09], with a revised version of the specification (ISO/IEC 19770-2:2015) published in 2015 [SWID15]. Since that time, a growing number of vendors have integrated SWID tags into their software products. SWID tags significantly simplify the task of identifying pieces of software: instead of relying on discovery processes that look for clues as to software presence, such as the presence of particular files or registry keys, vendors can use a readily available list of SWID tags that provides simple and immediate evidence as to the presence of the given piece of software.

SWIMA has no reliance on the presence or management of SWID tags on an endpoint as described in the ISO 2015 SWID tag specification. However, as presented in the ISO 2015 SWID tag specification, the data model for describing software provides a robust and comprehensive way of describing software and is adopted here as a means of representing and transmitting software information. It should be emphasized that the use of the ISO SWID tag data model makes no assumption as to whether (1) the source of the recorded information was, in fact, an ISO SWID tag harvested from the endpoint or (2) the information was created using some other source and normalized to the SWID format.

6.1.1. Guidance on Normalizing Source Data to ISO 2015 SWID Tags Using XML

Any record associated with this Software Data Model Type MUST conform to [SWID15].

If generating a new ISO 2015 SWID tag, the software generating the tag MUST use a Tag Creator RegID that is associated with the generating software, unless this is impossible, in which case it MUST use the "http://invalid.unavailable" Tag Creator RegID value. (This conforms to conventions for an unknown tag creator in the ISO 2015

SWID tag specification.) Do not use a RegID associated with any other party. In particular, it is incorrect to use a Tag Creator RegID associated with the software being described by the tag, the enterprise that is using the software, or any other entity except that of the party that built the tool that is generating the SWID tag. This reflects the requirement that the Tag Creator RegID identify the party that created the tag. Moreover, any generated tags SHOULD conform to guidance for tag creators as provided in NISTIR 8060 [NIST8060], which provides additional recommendations to increase interoperable use of SWID tags.

6.1.2. Guidance on Creation of Software Identifiers from ISO 2015 SWID Tags

A Software Identifier generated from an ISO 2015 SWID tag is expressed as a concatenation of the value of the Tag Creator RegID field and the Unique ID field. Specifically, (1) it MUST be of the form TAG_CREATOR_REGID "_" "_" UNIQUE_ID and (2) it consists of the Tag Creator RegID and the Unique ID from the tag connected with a double underscore (_), without any other connecting character or whitespace.

6.2. ISO 2009 SWID Tags Using XML

As noted above, ISO’s SWID tag specification provides a useful data model for representation of software information. As of the writing of this specification, while the ISO 2015 specification is considered more comprehensive and addresses some issues with the ISO 2009 specification, 2009-format SWID tags remain far more common in deployments. For this reason, ISO 2009 SWID tags are included in the "Software Data Model Types" registry.

6.2.1. Guidance on Normalizing Source Data to ISO 2009 SWID Tags Using XML

Any record associated with this Software Data Model Type MUST conform to [SWID09]. Any such tag SHOULD use a UTF-8 encoding [RFC3629] but MUST NOT alter the existing encoding if doing so would invalidate digital signatures included in the tag.

If generating a new ISO 2009 SWID tag, the software generating the tag MUST use a Tag Creator RegID that is associated with the generating software, unless this is impossible, in which case it MUST use "unknown", which indicates that the tag creator is unknown. (This conforms to conventions for an unknown tag creator in the ISO 2009 SWID tag specification.) Do not use a RegID associated with any other party. In particular, it is incorrect to use a Tag Creator RegID associated with the software being described by the tag, the enterprise that is using the software, or any other entity except that of the party that built the tool that is generating the SWID tag. This reflects the requirement that the Tag Creator RegID identify the party that created the tag.

6.2.2. Guidance on Creation of Software Identifiers from ISO 2009 SWID Tags

A Software Identifier generated from an ISO 2009 SWID tag is expressed as a concatenation of the value of the Tag Creator RegID field and the Unique ID field. Specifically, (1) it MUST be of the form TAG_CREATOR_REGID "_" "_" UNIQUE_ID and (2) it consists of the Tag Creator RegID and the Unique ID from the tag connected with a double underscore (_), without any other connecting character or whitespace.

7. Relationship to Other Specifications

This specification is expected to participate in a standard NEA architecture. As such, it is expected to be used in conjunction with the other protocols used in a NEA exchange. In particular, SWIMA attributes are conveyed over PB-TNC [RFC5793], which is in turn conveyed over some variant of PT (either PT-TLS [RFC6876] or PT-EAP [RFC7171]). These protocols have an especially important role, as they are responsible for ensuring that attributes defined under this specification are delivered reliably, securely, and to the appropriate party.

It is important to note that the Product Information, Numeric Version, and String Version attributes defined in the PA-TNC specification [RFC5792] are also meant to convey information about installed applications and the versions thereof. As such, there is some conceptual overlap between those attributes and the intent of this specification. However, PA-TNC was designed to respond to very specific queries about specific classes of products, while SWIMA is able to convey a broader query, resulting in a more comprehensive set of information regarding an endpoint’s installed software. As such, this specification provides important capabilities not present in the PA-TNC specification.

The NEA architecture is intended to support a broad range of activities and, as such, might be employed by other specifications. For example, requirement T-001 in the SACM Requirements document [RFC8248] notes that NEA can support data collection from endpoints within the broader SACM architecture. (Other parts of the NEA architecture, which SWIMA uses, meet the other SACM data transport requirements.) In the SACM architecture, a SWIMA-PV corresponds to a "SACM Collector" and a SWIMA-PC corresponds to a "SACM Internal

Collector". In the SACM architecture, SWIMA can support activities relating to software inventory collection. Specifically, SWIMA supports the SACM "Endpoint Posture Attribute Value Collection" use case (Section 2.1.3 in [RFC7632]) by describing a collection mechanism that enables event-driven, scheduled, and ad hoc data collection of software inventory information. SWIMA's flexibility with regard to the format of inventory data records means that it is compatible with virtually any data format that implements SACM's "Define, Publish, Query, and Retrieve Security Automation Data" use case (Section 2.1.1 in [RFC7632]). This is just one example of how SWIMA can support broader security solution standards. Note that while SWIMA can support these SACM use cases, SWIMA has no dependencies on the SACM architecture or any other context in which NEA might reasonably be applied.

8. Security Considerations

This section discusses some of the security threats facing SWIMA-PCs and SWIMA-PVs. This section primarily notes potential issues for implementers to consider, although it does contain a handful of normative requirements to address certain security issues. The issues identified below focus on capabilities specific to this document. Implementers are advised to consult other relevant NEA specifications, particularly [RFC5209] (the NEA architecture) and [RFC5792] (PA-TNC), for security issues that are applicable to such components in general.

8.1. Evidentiary Value of Software Inventory Evidence Records

The degree to which an endpoint's Software Inventory Evidence Collection accurately reflects the endpoint's actual software load and any changes made to this software load is dependent on the accuracy of the tools used to populate and manage the Software Inventory Evidence Records in this collection. While the SWIMA-PC is required to detect changes to an endpoint's Software Inventory Evidence Collection in near real time, some tools might not be designed to update records in the Software Inventory Evidence Collection in real time. This can result in a collection that is out of sync with actual system state. Moreover, tools might inaccurately characterize software or fail to properly record its removal. Finally, it is likely that there will be software on the endpoint that is not tracked by any source and thus is not reflected in the Software Inventory Evidence Collection. Tools that implement SWIMA ought to be aware of these potential issues and minimize them, but completely eliminating such issues is likely impossible. Users of collected Software Inventory Evidence Records need to understand that the information provided by SWIMA cannot be treated as completely accurate. Nonetheless, having endpoints report this information can still provide useful insights into the state of the endpoint's software load and can alert administrators and policy tools of situations that require remediation.

8.2. Sensitivity of Collected Records

Collected software records can be sensitive in nature. This can include both security sensitivities and privacy sensitivities. Privacy sensitivities are discussed more in Section 9. With regard to security, inventory records represent a wealth of information about the endpoint in question, and for an adversary who does not already have access to the endpoint a collection of the endpoint's inventory records might provide many details that are useful for mounting an attack. A list of the inventory records associated with an endpoint reveals a list of software installed on the endpoint. This list can be very detailed, noting specific versions and even patch levels; an adversary can use this information to identify vulnerable software and design efficacious attacks.

The following information might also be gleaned from a collection of software inventory records:

- An inventory record might include information about where the product was installed on a given endpoint. This can reveal details about the file organization of that endpoint that an attacker can utilize.

- An inventory record might include information about how the software was provided to the endpoint, who in an organization signs off on the package release, and who packaged the product for installation. This information might be used as a starting point for the development of supply chain attacks.

- Events affecting inventory records are reported with timestamps indicating when each given event occurred. This can give the attacker an indication of how quickly an organization distributes patches and updates, helping the attacker determine how long an attack window might remain open.

Any consolidated software inventory is a potential risk, because such an inventory can provide an adversary an insight into the enterprise's configuration and management process. It is recommended that a centralized software inventory record collection be protected against unauthorized access. Mechanisms to accomplish this can include encrypting the data at rest, ensuring that access to the data is limited only to authorized individuals and processes, and other basic security precautions.

8.3. Integrity of Endpoint Records

SWIMA-PCs maintain records of detected changes to the endpoint’s Software Inventory Evidence Collection. These records are used to respond to a SWIMA-PV’s request for change events. The SWIMA-PV might use a list of reported events to update its understanding of the endpoint’s Software Inventory Evidence Collection without needing to receive a full inventory report from the SWIMA-PC. For this reason, preserving the integrity of the SWIMA-PC’s record of events is extremely important. If an attacker modifies the SWIMA-PC’s record of changes to the endpoint’s Software Inventory Evidence Collection, this might cause the SWIMA-PV’s understanding of the endpoint’s Software Inventory Evidence Collection to differ from its actual state. The results of such an attack might include leading the SWIMA-PV to believe that (1) absent software was present or, conversely, that present software was absent or (2) patches have been installed even if this is not the case. Such an attack could also cause the SWIMA-PV to be unaware of other changes to Software Inventory Evidence Records. As such, the SWIMA-PC MUST take steps to protect the integrity of its event records.

In addition, records of established SWIMA-PV subscriptions also require protection against manipulation or corruption. If an attacker is able to modify or delete records of a SWIMA-PV’s established subscription, the SWIMA-PC might fail to correctly fulfill this subscription. The SWIMA-PV would not be aware that its subscription was not being correctly fulfilled unless it received additional information that indicated a discrepancy. For example, the SWIMA-PV might collect a full inventory and realize from this information that certain events had not been correctly reported in accordance with an established subscription. For this reason, the SWIMA-PC MUST protect the integrity of subscription records.

8.4. SWIMA-PC Access Permissions

A SWIMA-PC requires sufficient permissions to collect Software Inventory Evidence Records from all of its supported sources, as well as sufficient permissions to interact with the endpoint’s Posture Broker Client. With regard to the former, this might require permissions to read the contents of directories throughout the filesystem. Depending on the operating environment and other activities undertaken by a SWIMA-PC (or software that incorporates a SWIMA-PC as one of its capabilities), additional permissions might be required by the SWIMA-PC software. The SWIMA-PC SHOULD NOT be granted permissions beyond what it needs to fulfill its duties.

8.5. Sanitization of Record Fields

Not all sources of software inventory evidence are necessarily tightly controlled. For example, consider a source that gathers .swid files from the endpoint’s filesystem. Any party could create a new .swid file that could be collected and turned into a Software Inventory Evidence Record. As a result, it is important that the contents of source information not be automatically trusted. In particular, tools that read source information and the Software Inventory Evidence Records derived therefrom, including SWIMA-PCs, need to be careful to sanitize input to prevent buffer overflow attacks, encoding attacks, and other weaknesses that might be exploited by an adversary who can control the contents of a record.

8.6. PA-TNC Security Threats

In addition to the aforementioned considerations, the SWIMA protocol is subject to the same security threats as other PA-TNC transactions; see Section 5.2 of PA-TNC [RFC5792]. These include, but are not limited to, attribute theft, message fabrication, attribute modification, attribute replay, attribute insertion, and denial of service. Implementers are advised to consult the PA-TNC specification to better understand these security issues.

9. Privacy Considerations

As noted in Section 8.2, if an adversary can gain an understanding of the software installed on an endpoint, they can utilize this to launch attacks and maintain footholds on this endpoint. For this reason, the NEA Server needs to ensure that adequate safeguards are in place to prevent exposure of collected inventory records. For similar reasons, it is advisable that an endpoint only send records to a NEA Server that is authorized to receive this information and that can be trusted to safeguard this information after collection.

In addition, software inventory information can lead to insights about the endpoint’s primary user if that user is able to install software. (Note that users might be "able" to install their own software even if they are not "allowed" to do so.) This is especially true on endpoints that support "apps", as individual apps can be closely tied to specific groups or activities. This could conceivably allow inferences about things such as a user’s hobbies; the banks and other financial institutions that they use; and information about the user’s race, sex, or sexual orientation.

Organizations that collect software inventory information from endpoints ought to make sure the endpoints' users are aware of this collection. In addition, organizations should be aware that a software inventory associated with an individual, such as the inventory of the individual's primary endpoint, could expose sensitive personal information. For this reason, privacy safeguards are necessary for collected inventory information. Such safeguards would require not only protection of the inventory's confidentiality but also appropriate access controls so that only those trained in relevant privacy requirements are able to view the data.

10. IANA Considerations

This section extends multiple existing IANA registries. Specifically, it extends the "PA-TNC Attribute Types" and "PA-TNC Error Codes" registries defined in the PA-TNC specification [RFC5792] and the "PA Subtypes" registry defined in the PB-TNC specification [RFC5793] and extended in PA-TNC. This specification only adds values to these registries and does not alter how these registries work or are maintained. Consult the appropriate specifications for details on the operations and maintenance of these registries.

This section also defines a new IANA registry for "Software Data Model Types". The structure and requirements for this registry are provided, as well as guidelines for reviewers adjudicating the addition of new entries to this registry.

10.1. Guidance for the Designated Experts

For the "Software Data Model Types" registry defined by this specification, new values are added to the registry using the "Specification Required" process defined in RFC 8126 [RFC8126].

This section provides guidance to designated experts so that they may make decisions using a philosophy appropriate for this registry.

Designated experts should focus on the following requirements. All values in this IANA registry MUST be documented in a specification that is permanently and publicly available. Values MUST also be useful, not harmful to the Internet, and defined in a manner that is clear and likely to ensure interoperability.

Designated experts should encourage vendors to avoid defining similar but incompatible values and instead agree on a single value allocated via IETF standards. However, it is beneficial to document existing practice.

There are several ways to ensure that a specification is permanently and publicly available. It may be published as an RFC. Alternatively, it may be published in another manner that makes it freely available to anyone. However, in this latter case, the vendor MUST supply a copy to IANA and authorize IANA to archive this copy and make it freely available to all, if at some point the document becomes no longer freely available to all through other channels.

Sections 10.2, 10.3, and 10.4 define a new PA Subtype, new PA-TNC Attribute Types, and new PA-TNC Error Codes, respectively. Section 10.5 provides guidance to IANA in creating and managing the new "Software Data Model Types" registry defined by this specification.

10.2. PA Subtypes

The following is an extension to the list of PA Subtypes provided in Section 7.2 of [RFC5792] and defined in the "PA Subtypes" registry in Section 6.3 of [RFC5793]. See <https://www.iana.org/assignments/pb-tnc-parameters/>.

PEN	Integer	Name	Defining Specification
0	9	SWIMA Attributes	RFC 8412

10.3. PA-TNC Attribute Types

Section 5.2 of this specification defines several new PA-TNC attributes. The following values have been added to the "PA-TNC Attribute Types" registry defined in the PA-TNC specification. Note that Table 1 in Section 5.2 lists these attributes in both hexadecimal and decimal format. The decimal values given in that table are identical to those provided here. Note also that Table 1 includes an entry for the PA-TNC Error attribute, but the IANA information associated with the PA-TNC Error attribute is already defined in the PA-TNC specification and is not reproduced here.

PEN	Integer	Name	Defining Specification
0	13	SWIMA Request	RFC 8412
0	14	Software Identifier Inventory	RFC 8412
0	15	Software Identifier Events	RFC 8412
0	16	Software Inventory	RFC 8412
0	17	Software Events	RFC 8412
0	18	Subscription Status Request	RFC 8412
0	19	Subscription Status Response	RFC 8412
0	20	Source Metadata Request	RFC 8412
0	21	Source Metadata Response	RFC 8412

10.4. PA-TNC Error Codes

Section 5.15 of this specification defines several new PA-TNC Error Codes. The following values have been added to the "PA-TNC Error Codes" registry defined in the PA-TNC specification. Note that Table 9 in Section 5.15 lists these codes in both hexadecimal and decimal format. The decimal values given in that table are identical to those provided here.

PEN	Integer	Name	Defining Specification
0	4	SWIMA_ERROR	RFC 8412
0	5	SWIMA_SUBSCRIPTION_DENIED_ERROR	RFC 8412
0	6	SWIMA_RESPONSE_TOO_LARGE_ERROR	RFC 8412
0	7	SWIMA_SUBSCRIPTION_FULFILLMENT_ERROR	RFC 8412
0	8	SWIMA_SUBSCRIPTION_ID_REUSE_ERROR	RFC 8412

10.5. Software Data Model Types

For the "Software Data Model Types" registry (<https://www.iana.org/assignments/pa-tnc-parameters/#software-data-model-types>, each entry should include a human-readable name, an SMI PEN, a decimal integer value between 0 and 2^8-1 (inclusive), and a reference to the specification where the use of this data model is defined. This referenced specification needs to provide both a description of the format used by the data model and the procedures by which Software Identifiers are derived from a record expressed using this data model. Note that a specification that just defines the data model structure and its use is generally not sufficient, as it would likely lack the procedures for constructing a Software Identifier. This is why the table below uses the SWIMA standard for ISO SWID tags as the reference for the use of ISO SWID tags and does not reference the ISO SWID tag specification.

The following entries for this registry are defined in this document. They are the initial entries in the "Software Data Model Types" registry. Additional entries to this registry are added following the "Specification Required" policy defined in [RFC8126], following the guidelines in Section 10.1.

PEN	Integer	Name	Defining Specification
0	0	ISO 2015 SWID tags using XML	RFC 8412
0	1	ISO 2009 SWID tags using XML	RFC 8412
0	192-255	Reserved for local enterprise use	N/A

11. References

11.1. Normative References

[NIST8060] Waltermire, D., Cheikes, B., Feldman, L., and G. Witte, "Guidelines for the Creation of Interoperable Software Identification (SWID) Tags", DOI 10.6028/NIST.IR.8060, April 2016, <https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8060.pdf>.

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

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

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

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

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

[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, <https://www.rfc-editor.org/info/rfc5792>.

[RFC8089] Kerwin, M., "The "file" URI Scheme", RFC 8089, DOI 10.17487/RFC8089, February 2017, <https://www.rfc-editor.org/info/rfc8089>.

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

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

[SWID09] The International Organization for Standardization/ International Electrotechnical Commission, "Information technology - Software asset management - Part 2: Software identification tag", ISO/IEC 19770-2:2009, November 2009, <https://www.iso.org/standard/53670.html>.

[SWID15] The International Organization for Standardization/ International Electrotechnical Commission, "Information technology - Software asset management - Part 2: Software identification tag", ISO/IEC 19770-2:2015, October 2015, <https://www.iso.org/standard/65666.html>.

11.2. Informative References

[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, <https://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, <https://www.rfc-editor.org/info/rfc5793>.

[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, <https://www.rfc-editor.org/info/rfc6876>.

[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, <https://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, <https://www.rfc-editor.org/info/rfc7632>.

[RFC8248] Cam-Winget, N. and L. Lorenzin, "Security Automation and Continuous Monitoring (SACM) Requirements", RFC 8248, DOI 10.17487/RFC8248, September 2017, <https://www.rfc-editor.org/info/rfc8248>.

Authors' Addresses

Charles Schmidt
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730
United States of America

Email: cmschmidt@mitre.org

Daniel Haynes
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730
United States of America

Email: dhaynes@mitre.org

Chris Coffin
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730
United States of America

Email: ccoffin@mitre.org

David Waltermire
National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, Maryland
United States of America

Email: david.waltermire@nist.gov

Jessica Fitzgerald-McKay
United States National Security Agency
9800 Savage Road
Ft. Meade, Maryland
United States of America

Email: jmfitz2@radium.ncsc.mil